How to Use the Clustering-Based Ground Motion Selection Codes

Last updated on 07_07_2024 by Yiming Jia

Pre-requests:

1. Users need to have access to Python 3.10.14 (Tensorflow 2.13.1 required) and Matlab R2023b (Curve Fitting Toolbox, Econometrics Toolbox, and Sensor Fusion and Tracking Toolbox required).

The authors recommend running Python codes on Jupyter Notebook. Running the codes on different versions of Python, Tensorflow, or Matlab may result in compatibility issues. However, most compatibility issues can be resolved by following the suggestions in the error messages.

2. Users need to have knowledge of earthquake engineering to perform the ground motion (GM) pre-selection.

Step-by-step instruction:

Step 1: Pre-select pulse-type and non-pulse-type GMs based on knowledge of earthquake engineering to obtain candidate GMs.

The candidate GMs can be selected from the NGA-West 2 database (https://ngawest2.berkeley.edu/).

The target response spectrum can be determined from ASCE Hazard Tools (https://ascehazardtool.org/) and saved as a matrix (e.g., 16×2) in MATLAB data ("TargetSa.mat"). The first column is periods, and the second column is response spectra.

The range of magnitudes, which has been found to contribute most significantly to the seismic hazard, can be obtained by the analysis of deaggregated USGS hazard data (https://earthquake.usgs.gov/nshmp/hazard/disagg).

Step 2: Calculate and scale the logarithmic response spectra (lnSa) of candidate GMs and save them as a matrix as MATLAB data.

For example, 67 pulse-type candidate GMs result in MATLAB data ("GMIM_Pulse.mat"), which includes:

a) T (the range of periods defined based on the user's preference as a vector, such as 0.01:0.01:2.00s),

Note the length of T needs to be a multiple of four.

b) LogSa (the lnSa of 67 pre-selected pulse-type GMs as a matrix with a size of 67×200, where 67 is the number of GMs, 200 is the number of periods),

c) Scaled_Period (the period of interest used to scale lnSa as a scalar).

Step 3: Run MATLAB codes "Pretrain_Generation_Pulse.m" and "Pretrain_Generation_NonPulse.m" to generate the pretrain data.

This step may take a couple of hours, depending on the number of pre-trained GMs and the available computational resources.

Note that in "Pretrain_Generation_Pulse.m" and "Pretrain_Generation_NonPulse.m", the parameter – N_Pre (number of pre-trained GMs) on line 18 is user-defined. The author recommends defining N_Pre as a large number (e.g., ≥ 10,000).

The generated data are saved as "GMIM_Pulse_Pretrain.mat" and "GMIM_NonPulse_Pretrain.mat".

Step 4: Run Python codes "Autoencoder_Pulse_Pretrain.ipynb" and "Autoencoder_NonPulse_Pretrain.ipynb" on Jupyter Notebook.

This step may take several hours, depending on the number of candidate GMs and the available computational resources.

Note that in "Autoencoder_Pulse_Pretrain.ipynb" and "Autoencoder_NonPulse_Pretrain.ipynb", the following parameters are user defined:

a) NUM_EPOCH in the 1st block, line 16,

b) BATCH_SIZE in the 1st block, line 20.

The author recommends defining NUM_EPOCH as a large number (e.g., >= 10,000) and BATCH_SIZE as five or ten times the number of candidate GMs.

Each code automatically generates three folders. For the pulse-type GM case, these generated folders are

a) Data_Pulse_Pretrain (includes the latent features, reconstructed lnSa, loss, and running time),

b) Figure_Pulse_Pretrain (includes CAE architecture, quantile-quantile plot of lnSa, and loss plot),

c) Model_Pulse_Pretrain (includes the checkpoint, which is used in the following fine-tuning process).

Step 5: Run Python codes “Autoencoder_Pulse_Finetune.ipynb” and “Autoencoder_NonPulse_Finetune.ipynb” on Jupyter Notebook.

Note that in “Autoencoder_Pulse_Finetune.ipynb” and “Autoencoder_NonPulse_Finetune.ipynb”, the following parameters are user defined:

a) NUM_EPOCH in the 1st block, line 16,

b) BATCH_SIZE in the 1st block, line 20.

The author recommends defining NUM_EPOCH as a large number (e.g., ≥ 30,000) and BATCH_SIZE as the number of candidate GMs.

For each code, three folders are automatically generated (similar to Step 4).

Step 6: Run MATLAB codes “Kmeans_Pulse.m” and “Kmeans_NonPulse.m”.

The results are saved as Matlab data “Cluster_Pulse.mat” and “Cluster_NonPulse.mat”.

Step 7: Calculate the ground motion prediction equations using “GMPEs.xlsm” and knowledge of earthquake engineering, and save them as matrix as MATLAB data “GMPE.mat”.

For example, the GMPE as a 16×3 matrix is saved as MATLAB data (“GMPE.mat”). The first column is the periods, the second column is response spectra, the third column is response spectra + 1 standard deviation. All these data are from cells E23 to G38 in “GMPEs.xlsm”.

Step 8: Run MATLAB code “Generation_GM_Selection.m” to select the pulse-type and non-pulse-type GMs.

Note that this step may take a couple of hours, depending on the number of clusters, the number of realizations, and the available computational resources.

Note that in “Generation_GM_Selection.m”, the following parameters are user-defined

a) N_GM (number of GMs) on line 26,

b) R and epsilon from the analysis of deaggregated USGS hazard data on lines 29 and 30,

c) T_star (scaled period) on line 89,

d) N_Real (number of realizations) on line 121, which is recommended to be larger than 100.

The output file "GM_Selection. mat" includes

a) ID_Pulse_Final and ID_NonPulse_Final (the ID for the selected pulse-type and non-pulse-type GMs as two vectors),

b) LogSa_Final (the lnSa of selected GMs as a matrix),

c) Mean_LogSa_Final (mean of lnSa of selected GMs as a vector),

d) Std_LogSa_Final (standard deviation of lnSa of selected GMs as a vector).

e) CMS (conditional mean spectra as a vector),

f) CMS_sigma (sigma of conditional mean spectra as a vector),

g) T (range of periods as a vector).

Step 9: Run MATLAB code "Plot_Selected_GMs.m" to plot the conditional spectra and the lnSa of selected GMs (for visualization purposes only).