

IM Calculation Refactor

Please read the readme @ https://github.com/ucgmsim/IM_calculation/blob/master/README.md for instructions on how to run the code.

DONE

- IM calculations have been separated from the "old post-processing" repository, extracting the relevant functions and classes.
- IM values validated on Hypocentre and Kupe against "old post-processing" on the same data.
- Two types of workflows: text based (most likely observations) and binary based (simulations once the binary workflow is in place).
 - If binary workflow does not happen soon, the text based option will be used in both cases.
- Outputs match the formats requested in [File Formats Used On GM](#) and should therefore be usable on the upcoming Non-ergodic codes.
- Tested on very simple multi-process on Kupe with good speed-up using 40 and 80 cores. For the sample, 2228 stations were used

Machine	Cores	Time
Hypocentre	1	132m
Hypocentre	8	8.7m
Kupe	40	27m
Kupe	80	

OUTPUT STRUCTURE

With command : python calculate_ims.py ..//BB.bin b -o /home/yzh231/ -i Albury_666_999 -r Albury -t s -v 18p3 -n 112A -m PGV pSA -p 0.02 0.03 -e -geom -np 2

- **input file path:** ..//BB.bin
- **b:** input file type is **binary**
- **-o:** output result csvs location is **/home/yzh231**, default is **/home/\$user**
- **-i:** unique identifier/runname of the simrun and output folder name are **Albury_666_999**, default is **'all_station_ims'**. This attribute will be stored in the meta data file.
- **-r:** rupture name is **Albury**, default is unknown. This attribute will be stored in the meta data file.
- **-t:** type of simrun is **simulated**, default is unknown. This attribute will be stored in the meta data file.
- **-v:** version of simrun is **v18p3, default is XXpY**. This attribute will be stored in the meta data file.
- **-n:** station names used to perform im claculation are **112A**, default is all the stations in the binary file
- **-m:** measures used to perform im calculation are **PGV and pSA**, default is all the measures
- **-p:** period of pSA used to perform im calculation are **0.02 0.03**, default is Karim's 15 periods
- **-e:** In addition to the period specified by -p option, **use extended 100 period of pSA**, default not using
- **-c:** component of waveform acceleration used to perform im calculation is **geom**, default is '090, 000, ver'
- **-np:** number of processors used to perform im calculation is **2**, default is 2

The result is outputted to the following location, where:

- 'Albury_666_999' is the folder that contains all outputs. The folder name 'Albury_666_999' is made of the string specified by the '-i' argument. Default is 'all_station_ims' if not specified.
- 'Albury_666_999.csv' is the summary csv file that contains all stations' im calculations. The summary file name is made of the string specified by the '-i' argument.
- 'Albury_666_999_imcalc.info' is the meta data file. The meta data file name is made of the string specified by the '-i' argument.
- 'station' is the folder that contains all individual station's im_calculations. The folder name is defaulted and cannot be specified by the user.
- '112A_geom.csv' is the individual csv file that contains geom component im calculation for station 112A. Each name of the individual station csv file name is made of station_name + component

```
yzh231@hypocentre ~ % tree Albury_666_999
Albury_666_999
├── Albury_666_999.csv
├── Albury_666_999_imcalc.info
└── stations
    └── 112A_geom.csv
```

TEST FOR CALCUALTE_IMS.PY

All the steps below are to be carried out in hypocentre

1. Generate summary benchmark:

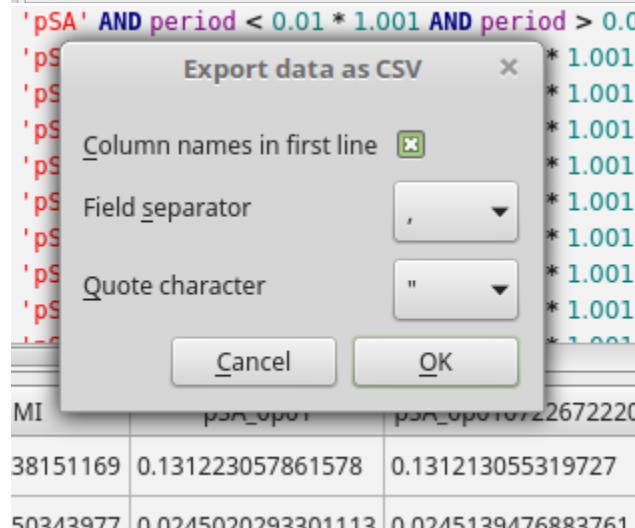
The following steps should only be performed once for each selected binary file

1. Select a source binary file: `/nesi/transit/nesi00213/RunFolder/daniel.lagrava/Kelly_VMSI_Kelly-h0p4_EMODv3p0p4_180531/BB/Cant1D_v2-midQ_leer_hfnp2mm+_rfv0p8_sd50_k0p045/Kelly_HYP01-03_S1244/Acc/BB_with_siteamp.bin`
 2. Identify corresponding database for the selected source binary file: `/home/nesi00213/RunFolder/wdl16/database_old_pp/database.db`
 3. Find the script to extract benchmark im value files from the database in step 2: `/nesi/projects/nesi00213/dev/impp_datasets/extract_ims.sql`
 4. Create a folder to store benchmark files. eg benchmark_im_sims
 5. Execute `extract_ims.sql` in `database.db` 4 times with specified components. eg: 'ver'

```
DB Browser for SQLite - /run/user/1000/gvfs/sftp:host=hypocentre,user=zyh231/home/nesi00213/runFolder/wd16/database_old.ppm/database.old

File Edit View Help
New Database Open Database Write Changes Revert Changes
Database Structure Browse Data Edit Pragmas Execute SQL
SQL1 SQL2
1 select station.name, 'ver' as component,
2 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PGa' AND component = 'ver') AS PgA,
3 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PGv' AND component = 'ver') AS PgV,
4 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'CAV' AND component = 'ver') AS CAV,
5 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'AI' AND component = 'ver') AS AI,
6 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'D575' AND component = 'ver') AS D575,
7 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'D595' AND component = 'ver') AS D595,
8 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PMS' AND component = 'ver') AS PMS,
9 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND component = 'ver') AS PsA,
10 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.01 * 1.001 AND period > 0.001 * 0.990 AND component = 'ver') AS psA_0,
11 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.01726722101032 * 1.001 AND period > 0.01726722101032 * 0.999,
12 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.0123284673944207 * 1.001 AND period > 0.0123284673944207 * 0.999,
13 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.0132194114846603 * 1.001 AND period > 0.0132194114846603 * 0.999,
14 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.0141747416292681 * 1.001 AND period > 0.0141747416292681 * 0.999,
15 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.0159119108295293 * 1.001 AND period > 0.0159119108295293 * 0.999,
16 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.01629575038462064 * 1.001 AND period > 0.01629575038462064 * 0.999,
17 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.017475284006768 * 1.001 AND period > 0.017475284006768 * 0.999,
18 (SELECT value FROM intensity_measure WHERE intensity_measure.station_id = stations.id AND measure.name = 'PSA' AND period < 0.018781742286038 * 1.001 AND period > 0.018781742286038 * 0.999
```

6. Export results to benchmark_im_sims/benchmark_im_sim_ver.csv. Click OK and don't change anything when 'Export data as csv' window prompts



- Repeat step 4 and 5 with different components: '090', '000', 'geom'
 - Now you have 4 summary benchmark files benchmark_im_sm_090/000/ver/geom.csv

2. Generate test input files

- Follow the instruction in [Binary Workflow FAQ](#), we can generate single waveform files. These waveforms are intended for the testing of ascii functionality of calculate_ims.py. Open a python cell

```
from qcore.timeseries import BBSeis
bb = BBSeis('/nesi/transit/nesi00213/RunFolder/daniel.lagrava/Kelly_VMSI_Kelly-h0p4_EMODv3p0p4_180531/BB
/Cant1D_v2-midQ_leer_hfnP2mm+_rvf0p8_sd50_k0p045/Kelly_HYP01-03_S1244/Acc/BB_with_siteamp.bin')
bb.all2txt(self, prefix='/home/$user/benchmark_im_sim_waveforms/', f='acc'):
```

Now we have all the waveforms.

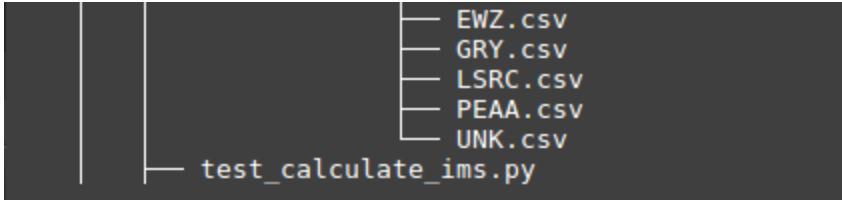
0002014.ver	0002307.000	0002540.090	0002811.ver	0002bce.000	10002b7.090	100086e.ver	1
0002015.000	0002307.090	0002540.ver	0002820.000	0002bce.090	10002b7.ver	1000871.000	1
0002015.090	0002307.ver	0002541.000	0002820.090	0002bce.ver	10002ca.000	1000871.090	1
0002015.ver	0002308.000	0002541.090	0002820.ver	0002bcf.000	10002ca.090	1000871.ver	1
0002016.000	0002308.090	0002541.ver	0002821.000	0002bcf.090	10002ca.ver	1000882.000	1
0002016.090	0002308.ver	0002540.000	0002821.090	0002bcf.ver	10002cc.000	1000882.090	1
0002016.ver	0002309.000	00025e0.090	0002821.ver	0002bd0.000	10002cc.090	1000882.ver	1
0002017.000	0002309.090	00025e0.ver	0002822.000	0002bd0.090	10002cc.ver	10008cc.000	1
0002017.090	0002309.ver	00025e1.000	0002822.090	0002bd0.ver	10002d9.000	10008cc.090	1
0002017.ver	000230a.000	00025e1.090	0002822.ver	0002bd1.000	10002d9.090	10008cc.ver	1
0002018.000	000230a.090	00025e1.ver	0002823.000	0002bd1.090	10002d9.ver	10008cf.000	1
0002018.090	000230a.ver	00025e2.000	0002823.090	0002bd1.ver	10002e3.000	10008cf.090	1
0002018.ver	00023ab.000	00025e2.090	0002823.ver	0002c82.000	10002e3.090	10008cf.ver	1
00020c1.000	00023ab.090	00025e2.ver	0002824.000	0002c82.090	10002e3.ver	10008d3.000	1
00020c1.090	00023ab.ver	00025e3.000	0002824.090	0002c82.ver	10002f2.000	10008d3.090	1
00020c1.ver	00023ac.000	00025e3.090	0002824.ver	0002c83.000	10002f2.090	10008d3.ver	1
00020c2.000	00023ac.090	00025e3.ver	0002825.000	0002c83.090	10002f2.ver	10008de.000	1
00020c2.090	00023ac.ver	00025e4.000	0002825.090	0002c83.ver	10002ff.000	10008de.090	1
00020c2.ver	00023ad.000	00025e4.090	0002825.ver	0002c84.000	10002ff.090	10008de.ver	1
00020c3.000	00023ad.090	00025e4.ver	0002826.000	0002c84.090	10002ff.ver	10008e4.000	1
00020c3.090	00023ad.ver	00025e5.000	0002826.090	0002c84.ver	1000302.000	10008e4.090	1
00020c3.ver	00023ae.000	00025e5.090	0002826.ver	0002c85.000	1000302.090	10008e4.ver	1
00020c4.000	00023ae.090	00025e5.ver	0002827.000	0002c85.090	1000302.ver	10008e8.000	1
00020c4.090	00023ae.ver	00025e6.000	0002827.090	0002c85.ver	100030e.000	10008e8.090	1
00020c4.ver	00023af.000	00025e6.090	0002827.ver	0002c86.000	100030e.090	10008e8.ver	1
00020c5.000	00023af.090	00025e6.ver	0002828.000	0002c86.090	100030e.ver	10008fb.000	1

3. Create Test Folder

- Create The test folder structure follows [Testing Standards for ucgmsim Git repositories](#)
- Select 10 stations you want to test and cp corresponding waveforms files to the singel_files directory as below
- Copy the source binary file 'BB_with_siteamp.bin' to the input folder
- Run 'write_benchmark_csv(sample_bench_path)' function inside test_calculate_ims.py to generate 'new_im_sim_benchmark.csv', where 'sample_bench_path' is the folder we created in 1.4 Generate summary_benchmark: benchmark_im_sims. This function should only be run once for each binary file.

NOW you have all the input files ready

```
└── test
    ├── test_calculate_ims
    │   └── README
    ├── sample1
    │   ├── input
    │   │   ├── BB_with_siteamp.bin
    │   │   ├── new_im_sim_benchmark.csv
    │   │   └── single_files
    │   │       ├── 00020d3.000
    │   │       ├── 00020d3.090
    │   │       ├── 00020d3.ver
    │   │       ├── 2002199.000
    │   │       ├── 2002199.090
    │   │       ├── 2002199.ver
    │   │       ├── CASH.000
    │   │       ├── CASH.090
    │   │       ├── CASH.ver
    │   │       ├── CFW.000
    │   │       ├── CFW.090
    │   │       ├── CFW.ver
    │   │       ├── DLX.000
    │   │       ├── DLX.090
    │   │       ├── DLX.ver
    │   │       ├── EWZ.000
    │   │       ├── EWZ.090
    │   │       ├── EWZ.ver
    │   │       ├── GRY.000
    │   │       ├── GRY.090
    │   │       ├── GRY.ver
    │   │       ├── LSRC.000
    │   │       ├── LSRC.090
    │   │       ├── LSRC.ver
    │   │       ├── PEAA.000
    │   │       ├── PEAA.090
    │   │       ├── PEAA.ver
    │   │       ├── UNK.000
    │   │       ├── UNK.090
    │   │       └── UNK.ver
    │   └── output
    │       ├── ascii_darfield_im_sim
    │       │   ├── ascii_darfield_im_sim.csv
    │       │   └── ascii_darfield_im_sim.info
    │       └── stations
    │           ├── 00020d3.csv
    │           ├── 2002199.csv
    │           ├── CASH.csv
    │           ├── CFW.csv
    │           ├── DLX.csv
    │           ├── EWZ.csv
    │           ├── GRY.csv
    │           ├── LSRC.csv
    │           ├── PEAA.csv
    │           └── UNK.csv
    └── binary_darfield_im_sim
        ├── binary_darfield_im_sim.csv
        └── binary_darfield_im_sim.info
        └── stations
            ├── 00020d3.csv
            ├── 2002199.csv
            ├── CASH.csv
            ├── CFW.csv
            └── DLX.csv
```



4. Run Pytest

Make sure you are currently under the test_calculate_ims folder, run:

```
$ pytest -v -s test_calculate_ims.py
```

```
yzh231@hypocentre ~/IM_calculation/test/test_calculate_ims (git)-[test] % pytest -v -s test_calculate_ims.py
=====
platform linux2 -- Python 2.7.14, pytest-3.2.2, py-1.4.34, pluggy-0.4.0 -- /usr/bin/python2.7
cachedir: ../../.cache
rootdir: /home/yzh231/IM_calculation, inifile:
plugins: cov-2.3.1
collecting 0 items
/home/yzh231/IM_calculation/calculate_ims.py
collected 6 items

test_calculate_ims.py::test_binary_script_calculate_ims python /home/yzh231/IM_calculation/calculate_ims.py /home/yzh231/IM_calculation/sample1/output -i binary_darfield_im_sim -t s -n 2002199 GRY 00020d3 UNK CASH CFW DLX LSRC EWZ PEAA -p 0.01 0.2 0.5 1
/home/yzh231/IM_calculation/test/test_calculate_ims/sample1/output/binary_darfield_im_sim/binary_darfield_im_sim.csv
Calculations are outputted to /home/yzh231/IM_calculation/test/test_calculate_ims/sample1/output/binary_darfield_im_sim

PASSED
test_calculate_ims.py::test_ascii_script_calculate_ims python /home/yzh231/IM_calculation/calculate_ims.py /home/yzh231/IM_calculation/sample1/output -i ascii_darfield_im_sim -t s -n 2002199 GRY 00020d3 UNK CASH CFW DLX LSRC EWZ PEAA -p 0.01 0.2 0.5 1.0 3.0 4.0
```

CHECKPOINTING & SPLITTING A BIG SLURM

Responsible scripts

1. slurm header template: https://github.com/ucgmsim/slurm_gm_workflow/blob/master/templates/slurm_header.cfg
2. im_calc_slurm template: https://github.com/ucgmsim/slurm_gm_workflow/blob/master/templates/im_calc_sl.template
3. submit_hf.py that generates the slurm files: https://github.com/ucgmsim/slurm_gm_workflow/blob/master/scripts/submit_hf.py
4. checkpointing functions: https://github.com/ucgmsim/slurm_gm_workflow/blob/master/scripts/checkpoint.py

Checkpointing

Checkpointing is needed for IM_calculation due to large job size and limited running time on Kupe. Therefore, we implemented checkpointing to track the current progress of an im_calculation job, and carry on from where the job was interrupted by slurm.

Note, the checkpointing code relies on the input/output directory structure specified in the im_calc_sl.template in the checkpoint branch. Failure to match the dir structure will result in runtime error. A quick fix would be modifying the template to suit your own dir structure.

Example:

(1) Simulation

Input/output structure defined in im_calc_sl.template

```
echo __calculating simulations__
{% for sim_dir, sim_name, fault_name in sim_dirs %}
  time python2 $IMPATH/calculate_ims.py {{sim_dir}}/Acc/BB.bin b -o {{sim_dir}}/../../../../IM_calc/ -np 40 -i {{sim_name}} -r {{fault_name}} -t s -s
{% endfor %}
```

Actual input data structure:

```
melody.zhu@kupe01:/nesi/nobackup/nesi00213/RunFolder/Cybershake/v18p6_batched/v18p6_1k_under2p0G_ab/Runs> ls
Aratititia  ArielBank  ArielEast  Astrolabe01  Astrolabe02  Astrolabe03  Astrolabe05  Astrolabe07  Awakeri  Barefell  Bidwill  Billys  BlueLk  BlueMtn  BooBooEAST
```

The input binary file is under:

```
/nesi/nobackup/nesi00213/RunFolder/Cybershake/v18p6_batched/v18p6_1k_under2p0G_ab/Runs/BlueMtn/BB/Cant1D_v3-midQ_OneRay_hfnp2mm+_rvf0p8_sd50_k0p045/BlueMtn_HYP28-31_S1514/Acc/BB.bin
```

The output **IM_calc** folder is under:

```
melody.zhu@kupe01:~/nesi/nobackup/nesi00213/RunFolder/Cybershake/v18p6_batched/v18p6_1k_under200_ab/Runs> ls ./BlueLk/IM_calc
BlueLk_HYP01-29_S1244 BlueLk_HYP03-29_S1264 BlueLk_HYP06-29_S1294 BlueLk_HYP09-29_S1324 BlueLk_HYP11-29_S1344 BlueLk_HYP15-29_S1384 BlueLk_HYP19-29_S1424 BlueLk_HYP22-29_S1454 BlueLk_HYP26-29_S1494 BlueLk_HYP29-29_S1524
BlueLk_HYP02-29_S1254 BlueLk_HYP04-29_S1274 BlueLk_HYP07-29_S1304 BlueLk_HYP10-29_S1334 BlueLk_HYP13-29_S1364 BlueLk_HYP18-29_S1414 BlueLk_HYP20-29_S1434 BlueLk_HYP23-29_S1464 BlueLk_HYP27-29_S1504
```

(2) Observed

Input/output structure defined in im_calc_1l.template

```
echo calculating observed
{% for obs_dir, obs_name, fault_name in obs_dirs %}
    time python2 $IMPATH/calculate_ims.py {{obs_dir}}/*/*/accBB a -o {{obs_dir}}/../IM_calc/ -np {{np}} -i {{obs_name}} -r {{fault_name}} -t o -s
{% endfor %}
```

Actual input data structure:

```
melody.zhu@kupe01:~> ls test_obs/IMCalcExample/
2012p713691 2012p764736 2012p801609 2013p049577 2013p368016 2013p653606 2013p708602 2013p817946 2013p868761 2014p237547 2014p933966 2014p965622 2016p119534 2016p158394 2016p355041 2122842 3631755
```

The output **IM_calc** folder is under:

```
melody.zhu@kupe01:~> ls test_obs/IMCalcExample/IM_calc/
2012p713691 2012p764736 2012p801609 2013p049577 2013p368016 2013p653606 2013p708602 2013p817946 2013p868761
```

Splitting a big slurm

Splitting a big slurm script into several smaller slurms is needed due to the maximum number of lines allowed in a slurm script on Kupe.

Inside **submit_imcalc.py** The **-ml** argument specifies the maximum number of lines of python call to calculate_ims.py/caculate_rrups.py. **Header** and **Footer** like '#SBATCH --time=15:30:00', 'date' etc are **NOT** included.

Say if the max number of lines allowed in a slurm script is 1000, and your (header + footer) is 30 lines, then the number **n** that you pass to **-ml** should be 0 < n <=967. eg. **-ml 967**.

Example:

We have 250 simulation dirs to run, by specifying -ml 100 (100 python calls to calculate_ims.py per slurm script), we expect 3 sim slurm scripts to be outputted.(1-100, 100-200, 200-250)

We have 3 observed dirs to run, by specifying -ml 100 (100 python calls to calculate_ims.py per slurm script), we expect 1 sim slurm scripts to be outputted.

We have 61 rrup files to run, by specifying -ml 100 (100 python calls to calcualte_rrups.py per slurm script), we expect 1 sim slurm scripts to be outputted.

Command to run checkpointing and splitting:

```
python submit_imcalc.py -obs ~/test_obs/IMCalcExample/ -sim runs/Runs -srp /nesi/nobackup/nesi00213/RunFolder
/Cybershake/v18p6_batched/v18p6_exclude_1k_batch_6/Data/Sources -ll /scale_aki_nobackup/filesets/transit
/nesi00213/StationInfo/non_uniform_whole_nz_with_real_stations-hh400_v18p6.ll -o ~/rrup_out -ml 1000 -e -s -i
OtaraWest02_HYP01-21_S1244 Pahiatua_HYP01-26_S1244 -t 24:00:00
```

Output:

```
-rw-r--r-- 1 melody.zhu melody.zhu 48890 Jul  1 03:47 sim_im_calc_0.sl
-rw-r--r-- 1 melody.zhu melody.zhu 48674 Jul  1 03:47 sim_im_calc_100.sl
-rw-r--r-- 1 melody.zhu melody.zhu 31922 Jul  1 03:47 sim_im_calc_200.sl
-rw-r--r-- 1 melody.zhu melody.zhu   1329 Jul  1 03:47 obs_im_calc_0.sl
-rw-r--r-- 1 melody.zhu melody.zhu 23246 Jul  1 03:47 rrup_im_calc_0.sl
```

To submit the slurm script:

```
$cp test.sl /nesi/nobackup/nesi00213/tmp/auto_preproc
$batch test.sl
```

The reason that we have to run 'test.sl' under '/nesi/nobackup/nesi00213/tmp/auto_preproc' is otherwise slurm cannot find machine.env specified by the test.sl script:

```
source machine_env.sh
```

TODO

- Creation of semi-automatic slurm generation that will have all the calls to produce the results as needed.
- Progress printing statements
- Rrup calculation on a smaller station list - currently when generating the slurm script it does the full grid even for stations outside the domain

Notes

- Extensive re-writing of code needs to have smaller deliverables in the future, as this simplifies the integration.