Binary Workflow

Introduction

A binary workflow was created to address issues we ran into with the standard text based workflow.

- Large file size of text (characters) vs binary (4 byte floats).
- Managing small files, some file systems had issues with deleting/moving many small files.
- IO, files are small and fragmented relative to each other. This makes head flying hours high and can even make the computer unresponsive for a non-computationally-intensive task such as zipping or the first stages of post-processing.
- Fragmented nature meant metadata was stored elsewhere and made retrieval difficult, have to keep track of not only the small files but the miscellaneous other small files that contained parameters.

SUMMARY

Old HF workflow

- losing 10% core hours by formatting text (this is for a simple simulation/simple *stoch* file)
- losing 250% storage space by storing text files

New BB workflow

- checks LF/HF compatibility, previous case of different *nt* parameter no longer undetected
- Kupe machine with 80 processes = 55 seconds, Hypocentre machine with 28 processes = 15 seconds (not fully vectorised)
- not fully tested yet
- no longer losing 600% storage space by storing both Vel and Acc as text

HOW TO RUN

HF

```
mpirun -n 80 python2 hf_sim.py source.stoch stations.ll HF.bin -m /home/nesi00213/VelocityModel/Mod-1D/Cant1D_v1-midQ.1d --duration 100.0 --dt 0.02 -i
```

source.stoch is the rupture file
stations.ll is the (lon, lat, name) file of the stations to run HF for
HF.bin is the output file
`-i` runs stations independently through the binary for reproducible results (same seed)
`-m` sets the global 1D velocity model, alternatively set the site specific 1D directory with **`-s`**
`-h` to see a complete list of options

BB

```
mpirun -n 80 python2 bb_sim.py LF/bevan2012_v3_s103252/OutBin /path/to/vm/ HF.bin vsites.vs30 BB.bin
```

The first parameter is the location to the **`OutBin`** folder from Emod3D where *seis* files can be found
The second is where the **`VM`** directory is located and contains *vs3dfile.s* and *params_vel.py*
HF.bin is the HF input binary
vsites.vs30 is the target vs30 file containing station_name vs30
BB.bin is the output file
hf_vs_ref is taken from the HF input (stored as part of HF)
lf_vs_ref is taken from the VM (*vs3dfile.s*)

Changes needed to run on Kupe

- Update qcore to a version that has qconfig.py
- Update EMOD3D so that the Fortran code to write binary files instead of text is present. TODO: remove the stdout for this executable, as it produces huge log files.

FAQ

How are these binary files accessed?

```
from qcore.timeseries import HFSeis
# load file
hf = HFSeis('HF.bin')
# retrieve timeseries (x,y,z)
hf.acc('CACS')
# only x
hf.acc('CACS', comp = <0 or hf.X>)
# store station as txt
hf.acc2txt('CACS')
# store all stations as txt, same as if ran standard txt version
hf.all2txt(prefix = 'Acc/hf_')
```

Same for BB (via **`BBSeis`** class)
LF has a wrapper so the OutBin folder can be accessed in the same way via **`LFSeis`** class.

What if I want LF Acc?

```
from qcore.timeseries import LFSeis
lf = LFSeis('LF/OutBin')
# just like for vel
lf.vel('CACS')
# you can access transparently converted accelerations
lf.acc('CACS')
```