

CalRecon Validation Script

General Description

The script CalRecon/src/test/test_CalRecon.py is expected to check the validity of the package. The script is running the executable test_CalRecon with various job options, then compare the output with what is expected. For each job options, there are three files. For example, for the "standard" options :

- jobOptions.txt : the job options themselves.
- jobOptions.log : the output of the last execution.
- jobOptions.ref : what is considered the reference output, i.e. the output of a previous execution which has been recognized as a success.

Worth to notice, the comparison between the log file and the ref file is not a complete blind diff. The output is filtered so to keep only the lines which make sense, and the relevant values are extracted.

When the script detects a change, this is not necessarily due to a bug in the package. This is only a signal that we must check the details of the job output. Either there is a bug in the package and it must be fixed, or there is a bug in the validation script (specifically in the way we filter and process the output), or there has been a valid change, then the new output is valid and must be made the new reference (simple copy of <options>.log into <options>.ref).

Portability

The validation has been developed under Linux. It should work under Windows, but we did not success to automate the compilation step. Thus, under windows, **the user must compile himself the executable test_CalRecon.exe** before running test_CalRecon.py .

Also, and that is not a detail, test_CalRecon.py must be called from a command-line prompt, and will directly issue cmt commands. Thus, **the windows command-line shell must be configured for cmt**. For a developer who usually work with MrVcmt, such a configuration is not done by default. Typically, one lack the definition of %CMTPATH%, which can be done in the windows registries, or interactively at the dos prompt, or within some script myscript.bat which you call at the dos prompt before starting the validation. Example :

```
dos> set CMTCONFIG=VC8debug
dos> call D:\ground\applications\CMT\v1r18p20041201\mgr\setup.bat
dos> set CMTPATH=D:\Users\chamont\MyDevProject:D:\ground\GlastRelease\v6r0
dos> cd D:\Users\chamont\MyDevProject\CalRecon\src\test
dos> python validate.py
```

Input Data

On one hand, we tried to setup standalone jobs which do not depend on numerous external packages, especially Tkr ones. We want here to check the internals of CalRecon, and hope the global integration is checked at a higher level. Also, we want to modify CalRecon and to be able to check the validity of such changes, independantly of parallel Tkr changes.

On the other hand, much code in CalRecon depends on Tkr reconstruction results. That is why, we made a script in the package rootTestData/data/gamma_5_gev_normal, which is running CalRecon, TkrRecon, and store the results in root files. Those files are used as an input to the CalRecon test jobs.

Job Options

jobOptions.txt

This is the main job, and the one which is also executed by the Release Manager. It is simply instanciating CalClustersAlg and CalEventEnergyAlg, thus redoing the clustering, and running all the energy corrections, with the tkr information available from the input file.

simplebOptions.txt

The same as above, with SingleClustering replaced with SimpleClustering, which will generate one cluster for each tower ?

fuzzyOptions.txt

The same as above, with FuzzyClustering. It only exists in the CVS branch CalReconFuzzyClustering.

Output Treatment

TO BE DETAILED MORE.

As said in the general description, the log and ref outputs are processed before being compared. Here are the steps of the process :

- we remove all the lines except the ones starting with "test_CalRecon".
- we extract and sort the energies displayed by the profile tool, the last layer tool, and the display of the clusters.

- we compute a mean value for each kind of energy, and a hashed value from the positions and directions (remember, especially for this last hash value : the idea is not to have meaningful values, but something we could check for change, so a hashed value is fine enough).