Technical validation of new developments and new cycles : *Mitraillette* & `checkpack`

Alexandre Mary, Météo-France

Code training days, 2019 Sept. 9th - Toulouse

Outline

Outline

Mitraillette : what, how ?

- ***Mitraillette*** : a collection of jobs, `NCONF` $\in \{1, 401, 501, 601, 901, 923, 927\}$ with various geometries & model options.

Mitraillette : what, how ?

- ***Mitraillette*** : a collection of jobs, `NCONF` $\in \{1, 401, 501, 601, 901, 923, 927\}$ with various geometries & model options.
- How it works :
  - each test job has a *proto-job* (= template of script)
  - namelists for each job are stored for each cycle
  - there is a building script (`mitraillette.x`) that builds the actual scripts from ***proto-jobs***, **cycle** and **binaries** to be used
  - input resources are taken from almost-hardcoded paths in the proto-jobs

Mitraillette : what, how ?

- ***Mitraillette*** : a collection of jobs, `NCONF` $\in \{1, 401, 501, 601, 901, 923, 927\}$ with various geometries & model options.
- How it works :
  - each test job has a *proto-job* (= template of script)
  - namelists for each job are stored for each cycle
  - there is a building script (`mitraillette.x`) that builds the actual scripts from ***proto-jobs***, **cycle** and **binaries** to be used
  - input resources are taken from almost-hardcoded paths in the proto-jobs
- Procedure for the user :
  1. cd to mitraillette directory
  2. define a list of {job $\Leftrightarrow$ binary} to be used, in a file
  3. run `mitraillette.x`, which creates a new incremental directory `<cycle>/mitraille_`*nnnn*, in which are built up the scripts for each job
  4. run the first job ; if not crashed, it triggers the second one, and so on

Just a bit of nomenclature

All jobs are named as a series of underscore-separated abbreviations, which define their content.
The first two are mandatory :

1. discriminates ECMWF, Arpege and LAM :
   - GE : Global-ECMWF = IFS
   - GM : Global-MF = Arpege
   - L1 : LAM 1D model ($\approx$ MUSC)
   - L2 : LAM 2D vertical-plan model
   - L3 : LAM 3D model
2. type of conf :
   - FCST : forecast
   - C923 : clim files conf 923
   - FPOF : fullpos (offline)
   - C601 : singular vectors
   - ...

Just a bit of nomenclature

Following parts of name specify options to be tested, e.g.

- HYD vs. NHE : hydrostatic vs. elastic NH
- SL2/SL3/EUL : semi-lagrangian 2/3 tsteps, eulerian
- ADIAB/ARPPHYISBA/AROPHYSFEX : adiabatic, Arpege physics, Arome physics with Surfex
- AROMALP1300/TL798S : 1.3km Alps domain, stretched T798 gauss
- VFE/VFD : vertical finite elements/differences
- PCC/PCF : cheap/full Predictor-Corrector scheme
- ...

Up to the user

- to launch next jobs when chaining is broken by crashed job(s)
- to compare the outputs of jobs to a reference : assert **bit-reproducibility** (of norms in listing), or **check differences** (in files) and understand where they come from
- to **deactivate chaining** when re-running failed jobs

Up to the user

- to launch next jobs when chaining is broken by crashed job(s)
- to compare the outputs of jobs to a reference : assert **bit-reproducibility** (of norms in listing), or **check differences** (in files) and understand where they come from
- to **deactivate chaining** when re-running failed jobs

$\Rightarrow$ `ciboulette/checkpack` :
towards **more ergonomy** and **automated sanity checks**

Up to the user

- to launch next jobs when chaining is broken by crashed job(s)
- to compare the outputs of jobs to a reference : assert **bit-reproducibility** (of norms in listing), or **check differences** (in files) and understand where they come from
- to **deactivate chaining** when re-running failed jobs

$\Rightarrow$ `ciboulette/checkpack` :
towards **more ergonomy** and **automated sanity checks**

(NB : Mitraillette/checkpack/ciboulette will be obsolete in a few cycles
$\hookrightarrow$ new validation system ***davaï*** — cf. my presentation at ALADIN/HIRLAM Wk Madrid 2019 / AG GMAP 2019)

Outline

checkpack

`checkpack.py` takes :

- a cycle
- a *gmkpack* compiled pack
- a list of jobs (pre-defined lists exist)

checkpack

`checkpack.py` takes :

- a cycle
- a *gmkpack* compiled pack
- a list of jobs (pre-defined lists exist)

and then :

- run Mitraillette (build jobs)
- launch the jobs with a mini-scheduler, more flexible than original chaining

It's only a ***handy wrapper*** around Mitraillette.

checkpack

`checkpack.py` takes :

- a cycle
- a *gmkpack* compiled pack
- a list of jobs (pre-defined lists exist)

and then :

- run Mitraillette (build jobs)
- launch the jobs with a mini-scheduler, more flexible than original chaining

It's only a ***handy wrapper*** around Mitraillette.

If you also give a reference, where to find outputs of Mitraillette execution on the reference cycle, it will trigger automatic comparisons :

$\Rightarrow$ `ciboulette`

ciboulette

`ciboulette` takes

- ***test*** and ***reference*** Mitraillette job(s) ***output listings***

ciboulette

`ciboulette` takes

- ***test*** and ***reference*** Mitraillette job(s) ***output listings***

and then compares ***norms*** found in listings for each job. Norms comparison consists in the ***number of different digits*** : 0 is bit-reproducibility, 15 is totally different fields.

ciboulette

`ciboulette` takes

- ***test*** and ***reference*** Mitraillette job(s) ***output listings***

and then compares ***norms*** found in listings for each job. Norms comparison consists in the ***number of different digits*** : 0 is bit-reproducibility, 15 is totally different fields.
As output, it produces :

- for each job, a norms comparison file, where norms are compared **step by step** and **field by field**
- a ***graphical summary*** of all jobs, giving their worst norms comparison (among steps & fields)

Install helper for Mitraillette

1. add paths to *checkpack/ciboulette* toolbox (and *vortex* if not already in paths), into `$PYTHONPATH` and `$PATH` :
   ⇒ cf. `beaufix:~mary/public/mocuba/_install_bull`
2. execute mitraillette install helper :
   `mitraillette_install.py`
   which will install to `$HOME/mitraillette`
   You can export `MIT_INSTALL_DIR` beforehand if you want to choose a different directory.

- NB : since Karim Yessad left, one should take mitraillette from P.Saez :
  `mitraillette_install.py --from /home/gmap/mrpm/saez/mitraille`
- NB2 : Mitraillette is now maintained by H.Petithomme and P.Saez

Test my pack

- ***run a job*** on the pack I just compiled :

  -
    ```
    cd ∼/pack/planet_object
    checkpack.py -c 46t1 -j mitraillette:L3_FCST_HYD_SL2_VFD_AROPHYSFEX_MAD
    _AROMALP1300
    ```

  or

  - `checkpack.py -c 46t1 -j mit[...] -b ∼/pack/planet_object/bin/MASTERODB`
- to ***list*** the available jobs and job sets :

  `checkpack.py --list_sets`
- run **all jobs**, and **compare to reference** outputs (in P.Saez directory) :

  `checkpack.py -c 46t1 -j mitraillette:all -r ∼saez/cy46t1`
- **help** : `checkpack.py -h`

(Re-)generate ciboulette summary

The `ciboulette` comparison is also useable on a set of jobs already executed, either natively using *Mitraillette* or with `checkpack`.

- any generated job can be modified and re-ran individually with `sbatch`
- **re-generate summary** for the bench `mitraille_`*nnnn* (implies to be in `$MIT_INSTALL_DIR`) :

  `ciboulette.py cy46t1 ~saez/mitraille/cy46 -t mitraille_`*nnnn* `-i`
- **help** : `ciboulette.py -h`

Outline

Hands-in

1. install mitraillette/checkpack/ciboulette
2. make a pack on top of CY46T1
3. modify `coupling/external/gpcou/esrlxt1.F90` :
   replace α by α^2 in computation of `PGT1GMV` relaxation
4. compile
5. check the pack on jobset `mitraillette:dev`, compared to
   `~saez/mitraille/cy46t1`
   $\Rightarrow$ cf. ciboulette output
6. assume α^2 was a bug, get back to α, recompile
7. re-run the job alone
8. re-build the ciboulette graphical output