MITRAILLETTE: ENVIRONNEMENT FILES AND USER'S GUIDE.

Karim YESSAD and Alexandre MARY (CNRM/GMAP).

March 15, 2018

Version v042018.
Valid for cycle 46.
Physics cf. cy42_op2.
Description stands for machine beaufix.

1 Introduction.

MITRAILLETTE is a software designed to do some basic validations on ARPEGE, ALADIN, AROME, and versions used by ALADIN or HIRLAM partners. It is shared between METEO-FRANCE, ALADIN partners and HIRLAM partners. It is currently not used by ECMWF.

History can be summarized as follows:

- 1999: first version of MITRAILLETTE, for ALADIN only.
- 2008: portable version of MITRAILLETTE, for LAM models only.
- 2009: portable version of MITRARP, for ARPEGE only.
- 2014: merge of MITRARP and MITRAILLETTE.
- 2015: archive MITRAILLETTE environnement on GIT library.
- 2017: deep rewritings, to make modifications easier for everybody.

The current design of MITRAILLETTE allows to use it on different machines with minimal adaptations concentrated in a reduced number of environnement files: that allows a minimal amount of work when scripts must be adapted to a new machine. The design has been adapted in order that introduction of new configurations to validate will be easy for everybody. It is desirable that everybody can contribute to MITRAILLETTE evolutions, via GIT branch contributions, as it is already done for code source modifications.

This document explains how to use MITRAILLETTE, and provides information how to modify a configuration or how to introduce a new configuration.

Additional procedures have been added, or are expected to be added soon:

- A procedure to update namelists from a cycle to another one, including namelist normalisation.
- A procedure to do some checkings on namelists.
- A procedure to compare experiment outputs between two cycles.
- A new procedure to launch experiments.

Use of the following procedures assumes that the following variables are defined in .bash_profile of the machine where validation is done:

```
export STATION=[name of the machine]
export AMAHOME=/home/gmap/mrpe/mary
```

and:

```
export MIT_INSTALL_DIR=${HOME}/mitraille
VORTEX_VERSION="olive"
VORTEX_INSTALL_DIR=${VEROLIVE_HOME}/vortex/vortex-$VORTEX_VERSION
export PYTHONPATH=$PYTHONPATH:$VORTEX_INSTALL_DIR
export PYTHONPATH=$PYTHONPATH:$VORTEX_INSTALL_DIR/src
export PYTHONPATH=$PYTHONPATH:$VORTEX_INSTALL_DIR/site
module load python
export PATH=$PATH:$PYTHONPATH:$VORTEX_INSTALL_DIR/bin
# mocuba
MOCUBA_ROOTDIR=${AMAHOME}/public/mocuba-dev
export PYTHONPATH=$MOCUBA_ROOTDIR:$PYTHONPATH
export PATH=$MOCUBA_ROOTDIR/bin:$PATH
```

2 Modifications since the former version.

- Technical modifications, variables renaming, some cleanings.
- Production of a gathered historic file for configurations using IO server.
- NPRINTLEV=1 (at least for the last cycles).

3 File environnement and user's guide for MITRAILLETTE.

3.1 Identifier.

The important thing to know is that each configuration to validate has an unic identifier. Each configuration generally does one task. The identifier appears at several places. In particuliar:

- Protojob script has name: [identifier].pjob . Each protojob is sufficiently auto-documented, so it is useless to repeat in this documentation a long and tedious inventory of existing configurations with all details (contrary to what was done until 2016).
- Main namelist has name: [identifier].nam .
- Identifiers also appear in files mitraillette.x, PRO_FILE..., profil_table.

3.2 Distributed memory environnement.

We introduce some generic names for different freedom degrees currently allowed for distributed environnement:

- Tunable parameters (via the profil_table configuration file):
  - NTASK_IO: the number of processors (tasks) used by the IO server.
  - NTASKS_TOT: the total number of processors (with the IO server processors).
  - NBNODES denotes the number of nodes.
  - NBTHREADS is the total number of OpenMp threads.
- Other quantities:
  - NTASKS=NTASKS_TOT-NTASK_IO: the number of processors (without the IO server processors).
  - MPITASKS_PER_NODE: ratio NTASKS_TOT/NBNODES.
  - MBX_SIZE is the box size.
- Example: NTASKS_TOT=64, NTASKS=60, NTASK_IO=4, NBNODES=16, NBTHREADS=10 means that there are 4 IO server processors, 60 model processors, 16 nodes, and 10 OpenMp threads used per node.

These quantities appear throughout the whole software with different denominations, summarised below (double underscores are omitted).

- Number of processors (tasks) used by the IO server:
  - mitraillette.x: NPROC_IO, nb_proc_io, job_nproc_io
  - code: NPROC_IO in NAMIO_SERV
  - protojobs: NTASK_IO, nb_proc_io
  - profil_table: nproc_io
  - multiheader: not used
  - namelist RHS: NTASK_IO
- Number of model processors:
  - mitraillette.x: NTASKS, ntasks
  - code: NPROC in NAMPAR0
  - protojobs: NTASKS, ntasks
  - profil_table: not used
  - multiheader: not used
  - namelist RHS: NTASKS
- Total number of processors:
  - mitraillette.x: NTASKS_TOT, ntasks_tot, job_ntasks_tot
  - code: no variable describing this quantity
  - protojobs: not used
  - profil_table: nprocs
  - multiheader: ntasks_tot, but also MPI_TASKS and SLURM_NTASKS

  - namelist RHS: not used

- Number of nodes:
  - mitraillette.x: NBNODES, nb_nodes, job_nnode
  - code: no variable describing this quantity
  - protojobs: not used
  - profil_table: nnode
  - multiheader: nb_nodes, but also NNODES and SLURM_JOB_NUM_NODES
  - namelist RHS: not used
- Ratio (total number of processors)/(number of nodes):
  - mitraillette.x: not used
  - code: no variable describing this quantity
  - protojobs: not used
  - profil_table: not used
  - multiheader: MPITASKS_PER_NODE
  - namelist RHS: not used
- Number of threads OpenMp:
  - mitraillette.x: NBTHREADS, nb_threads, job_nthreads
  - code: no variable describing this quantity
  - protojobs: not used
  - profil_table: nthreads
  - multiheader: nb_threads, but also OMP_NUM_THREADS and SLURM_CPUS_PER_TASK
  - namelist RHS: not used
- Box size:
  - mitraillette.x: not used
  - code: MBX_SIZE in NAMPAR0
  - protojobs: MBX_SIZE
  - profil_table: not used
  - multiheader: not used
  - namelist RHS: MBX_SIZE

In the current release of MITRAILLETTE one (and only one) distributed memory profile must be chosen. We do not distinguish any longer between monoproc and multiproc tasks: a task using 1 processor and 1 node is a particuliar case of multiproc task.

3.3 List of files.

On the machine where validation is done, a directory ${HOME}/mitraille must be created. Under this directory one must find the following files and call tree (example is given for cy46 validation):

- cy46 (dir)
- io_serv_tools (dir)
- mitraillette.x
- namelist/cy46/...nam
- PRO_FILE.cy46_arpref
- PRO_FILE.cy46_aldref
- protojobs/...pjob
- protojobs/beaufix/config_CY46
- protojobs/beaufix/jobtrailer
- protojobs/beaufix/multiheader

- protojobs/beaufix/profil_table

More details about these files:

- mitraillette.x is the main procedure launching MITRAILLETTE: it requires two arguments: the cycle (in uppercase characters), and the PRO_FILE file.
- The PRO_FILE... are files containing the list of tasks to be launched and the executable to be used for each task. Each line contains the task identifier, and the actual executable name (complete path required) to be used.
- The directory io_serv_tools contain tools to concatenate output files produced by jobs using IO server. In particuliar it must contain a copy of procedures arpifs/io_serv/io_poll and ifsaux/lfi_alt/lfi_, and some links on ifsaux/lfi_alt/lfi_: lfi_, lfi_copy, lfi_move.
- The directory "namelist/cy46" contains a list of namelists which should be norm-compliant (see documentation (IDARC)).
- The directory "protojobs" contains a list of scripts.
- The directory "protojobs/[computer name]" contains a list of environnement files:
  - files "config_CY....": give information about directories where some files must be found or stored, modified definitions of some commands (doing file copy, task launching for example), and definition of some other environnement variables.
  - file "jobtrailer" contains the right version of "ja" (to have CPU and memory use).
  - file "multiheader" contains the script header.
  - file "profil_table" contains seven columns giving information about number of processors, nodes, threads, memory allocation, elapsed time CPU time required for each job.

  All these files have a content which may depend on the machine.

The content of namelists and "protojobs.." scripts must not be machine dependent.

Additionally to that a directory is required to store executables and some other ones to store input files (initial files, coupling files, climatological files for example).

Example of where to find the scripts and files (beaufix):

- Scripts can be found on yessad/mitraille/protojobs
- Namelists can be found on yessad/mitraille/namelist/[cycle]
- Files PRO_FILE.. can be found for example on yessad/mitraille
- ARPEGE and LAM model files can be found on yessad/anal_mitraille
- Files containing some constant values can be found on yessad/const.
- Files "config..", "jobtrailer", "multiheader" and "profil_table" can be found on yessad/mitraille/protojobs/beaufix.
- IO server tools can be found on yessad/mitraille/io_serv_tools.

3.4 How to launch MITRAILLETTE: classical method.

We consider the following example:

- Validate on machine "beaufix" a oper-type configuration 001 (task "GM_FCTI_HYD_SL2_VFE_ARPPHYSFEX_SLT_IOSV_TL798S"); multiprocessor job only.
- The user is mrpx888.
- Validation on cycle 46.
- Use the executable mrpx888/executable/cy46_master-main.01.IMPI512IFC1601_ref.x.exe

Actions to be done:

- File mrpx888/mitraille/PRO_FILE.cy46_aldref must contain the following line:
  GM_FCTI_HYD_SL2_VFE_ARPPHYSFEX_SLT_IOSV_TL798S
  ${HOME}/executable/cy46_master-main.01.IMPI512IFC1601_ref.x.exe
- Launch MITRAILLETTE as follows (go to directory mrpx888/mitraille):
  `mitraillette.x CY46 PRO_FILE.cy46_aldref`

Some environnement files are created, with a experiment number (for example we consider that this four digit number is 4407):

- On directory mrpx888/mitraille: rank_file.x4407, job_end.x4407, test.x4407, log_file_CY46_4407, rank_last.x4407, mitraillette.o4407, mitra_home_location.
- On directory mrpx888/mitraille/cy46/mitraille_4407:
  GM_FCTI_HYD_SL2_VFE_ARPPHYSFEX_SLT_IOSV_TL798S.cjob
  and its link chainjob_000

Launch procedure test.x4407: it launches chainjob_000 on "beaufix". The output file will be stored on directory mrpx888/mitraille/cy46/mitraille_4407.

This classical method (with job chaining) is expected to disappear in the future.

Additional remarks:

- It is possible to use a different executable for a set of experiments, without changing the content of PRO_FILE.. files; proceed as follows, for experiment number 4407:
  - Execute statement: `export MY_PRIORITY_MIT_EXE=[new executable with complete path]`
  - Launch `test.x4407`: experiments will use the new executable.
  - Execute statement (to recover use of the original executable): `unset MY_PRIORITY_MIT_EXE`
- It is possible to have a temporary removal of the job chaining; proceed as follows:
  - Remove job chaining: `export MIT_UNCHAINED_JOB="YES"`
  - Restore job chaining: `export MIT_UNCHAINED_JOB=""`, or `unset MIT_UNCHAINED_JOB`

3.5 How to launch MITRAILLETTE: a new method with "checkpack".

See below in section "Additional procedures" the paragraph about `checkpack.py`. More details will be given in a future release of this documentation. It is expected in the future that this will be the only way left to launch MITRAILLETTE.

4 Configurations identifiers.

Each configuration has an identifier which has several tens of characters. The old four-letter codes (ex: ah1e, ar1t, mhlj) have been abandoned.

- Identifier starts by a group of 2 characters saying if model is a global or a LAM one.
  - GM: global model, LECMWF=F.
  - GE: global model, LECMWF=T.
  - L3: 3D LAM model.
  - L2: 2D LAM model (vertical-plane).
  - L1: 1D LAM model (column model).
- The following group has generally four characters, giving configuration type (forecast, fullpos, etc).
  - FCST: forecast without initialisation.
  - FCTI: forecast with DFI initialisation.
  - C501: configuration 501 (test of linear tangent code).
  - C401: configuration 401 (test of adjoint code).
  - C601: configuration 601 (make singular vectors).
  - FPOP: off-line FULL-POS.
  - FPIN: in-line FULL-POS.
  - C923: configuration 923 to make climatologies.
  - C901: configuration 901 to make ARPEGE files from MARS files.
  - PGDI: make PGD file for input to configuration 923.
  - PGDS: make PGD file using output of configuration 923.

  - PGDC: make PGD file: conversion from LFI format to FA format.
  - FPMF: make filtering matrices via FULL-POS setup.
  - DILA: make dilatation-contraction matrices.
  - RGRI: make reduced Gaussian grids.

- For configurations having a temporal advance, we need groups of characters describing dynamics features:
  - Dynamical core: HYD (hydrostatic model), NHE (fully compressible non-hydrostatic model), NHQ (quasi elastic non-hydrostatic model).
  - Advection scheme: EUL (Eulerian), SL3 (three-time level semi-Lagrangian), SL2 (two-time level semi-Lagrangian).
  - Vertical discretisation: VFD (vertical finite differences), VFE (vertical finite elements).
  - For NH model: RDBBC2 (RDBBC with ND4SYS=2), GWADV2 (GWADV with ND4SYS=2).
  - Semi-implicit (SI), full predictor-corrector (PCF), cheap predictor-corrector (PCC).
  - Use standard Legendre transforms (SLT), fast Legendre transforms (FLT).
  - Use standard Fourier transforms (FT9), FFTW (FTW).
  - Is there first-order uncentering (VESL), second-order uncentering (XIDT), no uncentering (NDEC).
  - Is there SLHD diffusion (SLHD), static SLHD diffusion (SSLHD), old SLHD diffusion (OSLHD), SLHDKMIN > 0 (MSLHD).
  - COMAD activated (MAD).
  - IO server activated (IOS or IOSV).

- For configurations having a temporal advance, we need a group of characters describing physics package:
  - ADIAB: adiabatic job.
  - ARPPHYISBA: upper-air physics is operational ARPEGE one; surface physics uses ISBA.
  - ARPPHYSFEX: upper-air physics is operational ARPEGE one; surface physics uses SURFEX.
  - AROPHYSFEX: upper-air physics is operational AROME one; surface physics uses SURFEX.
  - ALRPHYISBA: upper-air physics is ALARO one; surface physics uses ISBA.
  - SIM5PHYISBA: upper-air physics is simplified physics adapted to configuration 501; surface physics uses ISBA.
  - SIM4PHYISBA: upper-air physics is simplified physics adapted to configuration 401; surface physics uses ISBA.
  - VSIPHY: very simplified physics (Buizza).
  - AROPHY1D: AROME physics adapted for 1D column model.
  - ARPPHY1D: ARPEGE physics adapted for 1D column model.

- For LAM models, a group of characters ending identifier gives information on forecast domain.
  - PGAL: ALADIN-PORTUGAL.
  - FROC: domain covering South of France.
  - FRAN: ALADIN-FRANCE.
  - GRANLMRT: large domain covering Europe, with tilted rotated geometry.
  - AROMALP1300: domain covering French Alps, resolution 1300m.
  - OC0500: domain covering southern France, resolution 500m.

- For global models, information about resolution and geometry may be given by a group of characters ending identifier.
  - TL031U: truncation TL031, unstretched untilted geometry.
  - TL030S: truncation TL030, stretched tilted geometry.
  - TL798S: truncation TL798, stretched tilted geometry.
  - TL1198S: truncation TL1198, stretched tilted geometry.

- FULL-POS configurations require information about the output domain:
  - SPGAUSS: spectral output on Gaussian grid (formerly denoted 927).
  - SPLELAM: spectral output on LELAM grid (formerly denoted E927 or EE927).
  - GPGAUSS: grid-point output on Gaussian grid.

  - GPLELAM: grid-point output on LELAM grid.
  - GPLALON: grid-point output on LALON (lat-lon) grid.
  - MODEL: input and output horizontal geometries are identical.
  - SURFLELAM: grid-point output on LELAM grid for surface fields.
  - CI, CIE: for outputs on LELAM domain, we give a precision if the output domain covers C+I (CI) or C+I+E (CIE).
- LAM configuration E923 requires information about the output domain (example LELAM_FRANCE, LELAM_LACE, LELAM_REUNION, LALON_FRANX01).
- Technical information:
  - IO server activated (IOS, IOSV).

Example 1: L3_FCST_NHE_SL2_VFD_AROPHYSFEX_GWADV2_PCCMADIOS_AROMALP1300 means:

- L3: this is a 3D LAM model.
- FCST: this is a forecast without DFI initialisation.
- NHE: forecast uses the fully compressible non-hydrostatic model.
- SL2: advection scheme is two-time level semi-Lagrangian.
- VFD: vertical discretisation uses finite differences.
- AROPHYSFEX: upper-air physics is AROME one; surface physics uses SURFEX.
- GWADV2: LGWADV=T, ND4SYS=2.
- PCCMADIOS: cheap predictor-corrector scheme is switched on; COMAD is switched on; IO server is switched on.
- AROMALP1300: forecast domain covers French Alps; horizontal resolution is 1300m.

We finally reproduce operational version of AROME on a smaller domain.

Example 2: GM_FCTI_HYD_SL2_VFE_ARPPHYSFEX_SLT_IOSV_TL798S means:

- GM: this is a global model with LECMWF=F.
- FCTI: this is a forecast with DFI initialisation.
- HYD: forecast uses the hydrostatic model (deep-layer effects are ignored).
- SL2: advection scheme is two-time level semi-Lagrangian.
- VFE: vertical discretisation uses finite elements.
- ARPPHYSFEX: upper-air physics is ARPEGE one; surface physics uses SURFEX.
- SLT: use of standard Legendre transforms.
- IOSV: IO server is switched on.
- TL798S: truncation is TL798, tilted, stretched (high resolution pole on France).

We finally reproduce something close to operational version of ARPEGE, with a smaller resolution.

Example 3: GM_FPOF_HYD_SPLELAM_CIE_LAM2 means:

- GM: this is a global model with LECMWF=F (for departure geometry).
- FPOF: this is an off-line FULL-POS configuration.
- HYD: forecast uses the hydrostatic model (deep-layer effects are ignored).
- SPLELAM_CIE: output geometry is a spectral LELAM one, covering C+I+E.
- LAM2: additional appendix in order to distinguish between close identifiers.

We finally reproduce something close to a E927 FULL-POS configuration to make LAM files from ARPEGE ones.

5 Examples of configurations modifications: a short user's guide.

5.1 How to introduce a new configuration?

- Define a new identifier: the identifier must comply the rules of existing identifiers.
- Reference to this new configuration must be added in the following files: mitraillette.x, PRO_FILE.currentcycle_aldref if LAM configuration, PRO_FILE.currentcycle_arpref if global model configuration, profil_table. Order of configurations must be the same in all these files.
- Appropriate profiles must be given in file profil_table. It is desirable to have an elapsed time limited to 10 mn for validation.
- Namelists used by this new configuration must be provided, and enter directories "namelist" and "namelist_ref". Name of main namelist is [identifier].nam . Additional ".selnam" namelists may be required (for example for configurations using FULLPOS on domain LALON, SURFEX). Namelists must be norm compliant (see documentation (IDARC)).
- Protojob used by this new configuration must be provided, and enter directory "protojob". Name of protojob is [identifier].pjob . Design of this protojob must comply the general design (see z_GM_frame.pjob or z_L3_frame.pjob). Please remember that $ECP can be a copy or a symbolic link, $CP is always a copy. "cp" (with or without antislash) are forbidden; "rm" and "cat" require an antislash. Commands "ftget" and "ftput" are forbidden.
- New input files may be required, and provided in this case.
- Contributor must check that:
  - the new introduced configuration works.
  - existing configurations still work.

5.2 How to remove an obsolete configuration?

- Reference to this configuration must be removed from the following files: mitraillette.x, PRO_FILE.currentcycle_aldref if LAM configuration, PRO_FILE.currentcycle_arpref if global model configuration, profil_table.
- Namelists and protojobs of this configuration must be removed.

5.3 How to modify an existing configuration?

- Need to modify namelist? modify it.
- Need to modify input files? provide new input files; modify protojob.
- Need to modify distributed memory environnement, memory, time: update the appropriate configuration files.
- Files "config.." may require an update when a new version of PGD, ecoclimap or RRTM files is necessary. This is the case for example when a new version of SURFEX is implemented (need to use a new set of ecoclimap files).

5.4 How doing porting on another machine?

- A new machine name must be introduced (new directory under "protojobs").
- A new set of files "config..", "jobtrailer", "multiheader", "profil_table", must be provided.
- Normally, mitraillette.x does not need to be modified.
- Normally, content of namelists and protojobs must not be modified.

6 Additional procedures.

Most of these procedures can be also used on OLIVE/VORTEX namelists and operational environnement namelists. When not precised, these procedures are present on machine beaufix.

6.1 Namelist normalisation (procedures xpnam and alignnamelist).

Procedure `xpnam` allows to put namelist elements in alphabetical order.

```
xpnam [namelist].nam
```

Procedure `alignnamelist` (directory yessad/ykproc) allows to put namelist elements in alphabetical order, and to put the same elements as the reference (empty) namelist "vide".

```
alignnamelist vide [namelist].nam
```

6.2 Namelist update (procedure tnt.py).

This procedure is under VORTEX environnement (`/home/mf/dp/marp/verolive/vortex/vortex/bin/tnt.py`), and uses some directive files stored under GIT. Example, to update namelists from cy45T1 to cy46.

```
tnt.py -d directives_updnam_cy45t1_to_cy46.py [namelist].nam
```

It is possible to update all namelists with appendix `.nam` as follows:

```
tnt.py -d directives_updnam_cy45t1_to_cy46.py *.nam
```

And for help about `tnt.py` usage:

```
tnt.py -h
```

6.3 Namelist checkings (procedure nam_check_consistency.py).

This procedure does additional checkings in namelists (for example it checks that attribute LPT of GFL does not appear in NAMGFL). It is stored under GIT (and also under `mary/public/nam_check_consistency.py`) and uses some VORTEX environnement. Examples of use:

```
nam_check_consistency.py [namelist].nam
nam_check_consistency.py *.nam
```

6.4 Individual output comparison (procedure compare_listings.py).

This procedure is under VORTEX environnement (`/home/mf/dp/marp/verolive/vortex/vortex/site/arpifs_listings/bin/compare_listings.py`).
This procedure does output comparison for one MITRAILLETTE configuration. Examples:

Norms comparison:

```
compare_listings.py -n listing1 listing2
```

Jo-Tables comparison:

```
compare_listings.py -j listing1 listing2
```

Jo-Tables comparison, print only the maximum differences for each step:

```
compare_listings.py -j listing1 listing2 -x
```

And for help about `compare_listings.py` usage:

```
compare_listings.py -h
```

6.5 Global output comparison (procedure ciboulette.py).

This procedure does output comparison for a set of MITRAILLETTE configurations.

For example we want to compare 46 outputs with 45t1 outputs:

- Experiment number for ARPEGE experiments is 3195.
- Experiment number for ARPEGE references is 3193.

Under directory "mitraille", launch command:

```
ciboulette.py cy46 cy45t1 -t 'mitraille_3195' -r 'mitraille_3193' -s _arpref
```

This procedure produces the file 46-45t1_arpref.html

And for help about `ciboulette.py` usage:

```
ciboulette.py -h
```

6.6 Running jobs in parallel (under progress...): procedure `checkpack.py`.

Developpement of this procedure is still under progress. More details will be given in a future release of this documentation.

For help about `checkpack.py` usage:

```
checkpack.py -h
```

7 Content of GIT library:

The MITRAILLETTE environnement is now stored on GIT library, like the ARPEGE code. It is stored under the project "validation", directory "validation/mitraille".

- Directory "doc": contains the present documentation, and file "history_difnam" which summarizes namelist evolutions since cycle 36.
- Directory "namelist": contains the namelists for current cycle.
- Directory "namelist_ref": contains the namelists for a reference cycle. Allows to rerun a reference for comparison.
- Directory "procedure": contains mitraillette.x and some automatic procedures allowing to update namelists.
- Directory "pro_file": contains PRO_FILE files.
- Directory "protojobs": contains protojob files and some configuration files. Directory protojobs/beaufix contains some configuration files.
- It is not necessary to create a directory io_serv_tools under the GIT library because content is already available in files arpifs/io_serv/io_poll and ifsaux/lfi_alt/lfi_.

What is stored under GIT must allow to do a set of runs with the current cycle, and the same runs with a (supposed validated) reference cycle.

All GIT users are welcomed to make evolve MITRAILLETTE by giving GIT branches.

Remark about reference operational cycle which is taken into account for validations:

- For developpement cycles, the reference operational cycle is the one which is operational at the time the current cycle is labelled. For cycle 46, the reference operational cycle is cycle 42_op2.
- For operational cycles, the reference operational cycle is the current operational one.

8 Expected future modifications:

Expected modifications are:

- Update some FULLPOS configurations to make them more consistent with the current operational use.
- Update the LACE-ALARO configuration.
- Validating more configurations without DFI, and less with DFI, could be studied.
- Validating configurations under L_OOPS=T.

A new validation tool is expected to validate bricks of 4DVAR assimilation: it will use Python language, and it will be stored under a new directory in project "validation" (the name is still to be decided). First developpements are expected late 2018, or maybe rather in 2019.

9 References:

- (IDMITR) Vanda Sousa da Costa, A. Deckmyn, A. Dziedzic, N. Bouzouita and G. Bölöni, 2006: MITRAILLETTE: procedure to validate a release of the code Aladin, version 4. Internal note, available on "http://www.cnrm.meteo.fr/gmapdoc/" (topics "Coding, phasing, porting").
- (IDARC) Yessad, K., 2018: Library architecture and history of the technical aspects in ARPEGE/IFS, ALADIN and AROME in the cycle 46 of ARPEGE/IFS. Internal note, available on "http://www.cnrm.meteo.fr/gmapdoc/" (topics "Coding, phasing, porting").