CAMxRunner: A modular environment for efficient CAMx simulations

Goals
- Eliminate human errors in the use of air quality models
- Enhance reproducibility
- Increase modelling speed
- Enable use of different model versions
- Save disk-space

Using CAMxRunner
- The installer guides the user through setup & compilation of source code
- CAMxRunner can be adapted to most environments by configuration
- The system can be adapted to virtually any naming convention by rules:
  Bash Variable
  CXX_PHOTOLYSIS_RATES_FILE_RULE="$(CXX_INPUT_DIR)/tuv $(CXX_RUN_YEAR) $(CXX_RUN_WD).out"

  At runtime is resolved to /my_input_dir/tuv_2006_5.out
- Often-used settings can be stored in a "base-configuration"
- Before an actual run, a so-called dryrun detects configuration issues
- At runtime, the system can notify the user of errors by Email/SMS/Twitter

The system
- is written in the shell language Bash
- stores configuration data centrally
- saves time by parallel execution (see right)
- is modular, so that all pre- and postproc. in our lab could be integrated
- saves space by dynamic compression
- separates models using a clean structure
- helps the user to compile source codes

How does Parallelization work in CAMxRunner?
- Sorting steps according to dependencies
  Example: simple 2-day run (arrows indicate dependencies)

  PreA (Day 1) – creation of emissions → PreA (Day 2)
  PreB (Day 1) – emission conversion → PreB (Day 2)
  CAMx (Day 1) – depends on CAMx (Day 2)
  Post (Day 1) – extraction of data → Post (Day 2)

  Execution of the plan in parallel (by background processes)
  Before starting a step, its dependencies are checked

Features & Structure (excerpt)
- Extensive documentation
- HTML Source-Code documentation
- Modular design
- Each module is a simple script
- Support for different models & versions
- Preprocessors include creation of initial and boundary conditions, AHOMAP, TUV, various converters and more
- Postprocessors include an extractor of model data and various converters

Acknowledgements
- Partly funded by the Swiss Federal Office for the Environment (FOEN)
- CAMx is provided by ENVIRON Inc. (free)
- Icons: px+mx, Double J Design

Compiler flags may influence outcome of a simulation
- Both runs used CAMx 4.51
  A was compiled with PGI
  B with Intel
  Difference diminishes when -fitconsistency flag of Intel compiler is active
  That is why CAMxRunner stores the compiler flags used!

Timeline of sequential approach (Dual-Core Machine)
Core I
  PreA (D. 1) → PreB (D. 1) → CAMx (Day 1) → Post (D. 1) → PreA (D. 2) → PreB (D. 2) → CAMx (Day 2) → Post (D. 2)
Core II
  CAMx (Day 1) → CAMx uses both cores due to OpenMP → CAMx (Day 2)

Timeline of parallel approach (Dual-Core Machine)
Core I
  PreA (D. 1) → PreB (D. 1) → CAMx (Day 1) → CAMx (Day 2) → Post (D. 1)
Core II
  PreA (D. 2) → PreB (D. 2) → CAMx (Day 1) → CAMx (Day 2) → Post (D. 2)

Time saved

Difference A-B of CO after 14 model hours (56 steps) with CAMx 4.51
A: Model compiled with pgI 7.1.1. B: Model compiled with ifort 11.1