ESM Pism Documentation

Release 1.0.0

Paul Gierz

Oct 23, 2020

CONTENTS

This guide will help you to start using the PISM ice sheet model with ESM-Tools framework.

Once everything is set up, you can start PISM runs in the same manner you would also start AWI-ESM, MPI-ESM, OpenIFS and similar models:

```
$ esm_runscripts <experiment_config.yaml> -e <expid>
```

A summary video describing how to set up your experiment_config.yaml file is given below:

https://www.youtu.be/I2PDO1AU0KU

In addition to installing the standard ESM-Tools, you additionally need to install the PISM Plugin:

```
$ pip install git+https://github.com/esm-tools-plugins/esm_pism
```

This gives you several new plugins for your job recipes:

- *esm_pism.plugin.pism_set_couplers*
- *esm_pism.plugin.pism_set_kv_pairs*
- *esm_pism.plugin.pism_set_flags*
- *esm_pism.plugin.pism_override_file*
- *esm_pism.plugin.pism_assemble_command*

The next section shows you how to set up your configuration file.

CHAPTER

ONE

TABLE OF CONTENTS:

1.1 PISM Configuration for ESM Runscripts

The standard tool to run experiments within the `esm-tools` framework is `esm_runscripts`:

```
$ esm_runscripts <experiment_config.yaml> -e <expid>
```

Todo: Update the link to point to the `release` branch later once it is merged.

The `esm_runscripts` tool follows a series of steps to set up and run your simulation, known as a job recipe. There is a recipe for each type of job, e.g. compute, post, couple. Each of these steps is a Python function which recieves the experiment configuration dictionary as an input, and returns the (possibly modified) dictionary for further use. In the case of PISM, the recipe followed by default is can be found here:

```
compute_recipe:
    - "_create_setup_folders"
    - "_create_component_folders"
    - "initialize_experiment_logfile"
    - "pism_set_kv_pairs"
    - "pism_set_flags"
    - "pism_set_couplers"
    - "pism_override_file"
    - "_write_finalized_config"
    - "assemble_filelists"
    - "copy_tools_to_thisrun"
    - "_copy_preliminary_files_from_experiment_to_thisrun"
    - "_show_simulation_info"
    - "copy_files_to_thisrun"
    - "pism_assemble_command"
    - "add_batch_hostfile"
    - "copy_files_to_work"
    - "write_simple_runscript"
    - "report_missing_files"
    - "database_entry"
    - "submit"
```

The `pism_set_kw_pairs`, `pism_set_flags`, `pism_set_couplers` functions set up various parts of the `pismr` command which will be run by the batch system. The `pism_override_file` generates a `pism_overrides.nc` file, and the `pism_assemble_command` puts together the actual call.

Note: In the following, we will sometimes refer to a nested part of a configuration via dot notation, e.g. `a.b.c`.

Here, we mean the following in a YAML config file:

```yaml
a:
  b:
    c: "foo"
```

This would indicate that the value of `a.b.c` is `"foo"`. In Python, you would access this value as a["b"]["c"].

1.1.1 Setting Key-Value Pairs

A "key-value pair" is defined as a `pismr` flag which also requires an argument. For example, a pismr run which uses the shallow ice approximation enhancement factor of 3 would need this if written directly in the command line:

```
$ pismr <...> -sia_e 3 <...>
```

In the YAML configuration file, the same can be done like this:

```yaml
pism:
    kv_pairs:
        sia_e: 3
        "-ssa_e": 5
```

Notice that we here have a nested dictionary. The `pism` main dictionary has a sub-directionary, `kv_pairs`, which in turn consist of dictionaries each with a key and a value. The keys are the names of the `pismr` flags to use, and the values are the arguments. On line 3 and 4 you can see that including the - before the flag name is optional. If omitted, it will be prepended automatically for you. In case you need a double minus for a flag, you can do so with `"--key_name": value`. Setting the key to a string is important in this case!

1.1.2 Setting Flags

A "flag" is defined as a `pismr` command option that does not require an extra argument. For example, the colleagues at PIK have a flag to automatically set several defaults for them quickly. This looks like:

```
$ pismr <...> -pik <...>
```

In the YAML configuration file, the same can be done like this:

```yaml
pism:
    flags:
        - "pik"
        # or:
        - "-pik"
```

In this case, the configuration `pism.flags` should be a list of strings which will be added to the `pismr` command. As with key-value pairs, including the leading - is optional and will be added for you.

1.1.3 Setting Couplers

Todo: Include a link to PISM docs concerning couplers.

The most interesting part of ice sheet modelling is examining interactions between the ice sheet and the remainder of the Earth system. `PISM` accomplishes this by the use of what is termed "couplers". Details are available in the PISM documentation. In a YAML file, you can specify couplers for the atmosphere, ice surface, and ocean interfaces. A generalized example:

```
pism:
    couplers:
        coupler_domain:
            actual_coupler:
                files:
                   file_tag1: file_one
                   file_tag2: file_two
                flags:
                    - "flagA"
                    - "flagB"
                kw_pairs:
                    key_one: value_one
                    key_two: value_two
```

Here, the config `pism.couplers.coupler_domain` should be one of `atmosphere`, `ocean`, or `surface`.

Warning: Using another domain other than atmosphere, ocean, or surface will result in an error!

This should be a dictionary of coupler methods, optionally with files, flags, and kw_pairs that are needed to use that coupler. You may have more than one `coupler` per `coupler_domain`, in which case the other couplers work as modifiers. Files are automatically copied into the experiment tree under the `forcing` directory, and later copied into the work directory, using only their base name. Flags and key-value pairs are added to the PISM call, as above. A more concrete example:

```
pism:
     couplers:
              atmosphere:
                  given:
                      files:
                          atmosphere_given_file: "${forcing_input_dir}/climate_forcing_
↪LIG_16km_monthly.nc"
                      kv_pairs:
                          atmosphere_given_period: 1
                          atmosphere.use_precip_linear_factor_for_temperature: "no"
                  lapse_rate:
                      files:
                          atmosphere_lapse_rate_file: "${forcing_input_dir}/usurf_
↪echam_PI_LIG.nc"
                      kv_pairs:
                          temp_lapse_rate: 7.9
                          precip_lapse_rate: 0
                          smb_lapse_rate: 0
              surface:
                 pdd:
```

(continues on next page)

(continued from previous page)

```
                    files:
                            surface_lapse_rate_file: "${forcing_input_dir}/usurf_
↪echam_PI_LIG.nc"
                    kv_pairs:
                            low_temp: 100
            ocean:
               pico:
                  files:
                     frontal_retreat_file: "${forcing_input_dir}/ocean_kill_
↪topg2000m_orkney.nc"
                     ocean_pico_file: "${forcing_input_dir}/ocean_forcing_8k_fesom_
↪LIG.nc"
                  flags:
                          - "pik"
                          - "kill_icebergs"
                  kv_pairs:
                          sea_level: "constant"
```

The above would translate to:

```
pismr -atmosphere given,lapse_rate -atmosphere_given_file climate_forcing_LIG_16km_
↪monthly.nc -atmosphere_given_period 1 -atmosphere.use_precip_linear_factor_for_
↪temperature no -atmosphere_lapse_rate_file usurf_echam_PI_LIG.nc -temp_lapse_rate 7.
↪9 -precip_lapse_rate 0 -smb_lapse_rate 0 -surface pdd -surface_lapse_rate_file␣
↪usurf_echam_PI_LIG.nc -low_temp 100 -ocean pico -frontal_retreat_file ocean_kill_
↪topg2000m_orkney.nc -ocean_pico_file ocean_forcing_8k_fesom_LIG.nc -pik -kill_
↪icebergs -sea_level constant
```

1.2 esm_pism Package

1.2.1 `plugin` Module

esm_pism.plugin.**pism_assemble_command**(*config*)

Puts together the final PISM command used to launch the model

Parameters **config** (*dict*) – The entire exp config

Returns **config** – The entire exp config

Return type dict

esm_pism.plugin.**pism_override_file**(*config*)

Generates a PISM Overrides file.

Opens the `pism_config.nc` file in your YAML (see below), or uses the default found in the model directory under `share/pism/pism_config.nc`. This is used to determine which override keys are valid. A new file is written which is used during your simulation.

Alternatively, you can provide an override file to use, in which case that one will be used rather than generating a new one.

Warning: It is currently not possible to provide both an override file and extend it!

Example

In your YAML, you can specify:

```
pism:
    # This config file will be used as a template rather than the
    # one in model_dir!
    config_file: "/some/path/to/a/config/file"
    overrides_kv_pairs:
        "frontal_melt.given.period": 3
```

Alternatively:

```
pism:
    overrides_file: "/some/path/to/an/overrides/file.nc"
```

Parameters **config** (*dict*) – The entire exp config

Returns **config** – The entire exp config

Return type dict

esm_pism.plugin.**pism_set_couplers**(*config*)

Parameters **config** (*dict*) – The entire exp config

Returns **config** – The entire exp config

Return type dict

esm_pism.plugin.**pism_set_flags**(*config*)

Parameters **config** (*dict*) – The entire exp config

Returns **config** – The entire exp config

Return type dict

esm_pism.plugin.**pism_set_kv_pairs**(*config*)

Parameters **config** (*dict*) – The entire exp config

Returns **config** – The entire exp config

Return type dict

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

E

M

P