POLARIS Quickstart Guide

Download

Download zip package from the homepage or clone the github repository via:

```
git clone https://github.com/polaris-MCRT/POLARIS.git
```

HINT: It is recommended to clone the git repository into the home directory. If downloaded from the homepage, extract the zip file into the home directory via:

```
unzip -q POLARIS-master-basic.zip -d ~/
```

Requirements

The following packages are required for the installation:

- gcc (preferred), icc, or clang++
- cmake (preferred), or ninja
- Python version >= 3.6 (packages: *numpy*, *setuptools*)

Installation (Linux)

Open a terminal/console and move into the POLARIS directory:

```
cd /YOUR/POLARIS/PATH/
```

Run the installation script:

```
./compile.sh -f
```

For the first installation, the option `-f` is required to install the CCfits and cfitsio libraries. Alternatively, these libraries can be installed with a package manager (root permissions are required):

```
sudo apt update
sudo apt install libccfits-dev libcfitsio-dev
```

If these packages are installed on the system, simply install POLARIS via

```
./compile.sh
```

For more information, type:

```
./compile.sh -h
```

POLARIS can now be executed from any newly opened terminal/console. However, to use it in already open terminals/consoles, execute the following command to update the environmental paths:

```
source ~/.bashrc
```

HINT: Please refer to the manual (Sect. 1.2) for installation on **macOS**. An installer to use POLARIS with Windows is not available yet.

Start a simulation

POLARIS simulations are performed by parsing a command file with the simulation parameters. Exemplary `.cmd` command files for various planetary models can be found in `projects/`.

These include a cloud-free Rayleigh-scattering atmosphere (`rayleigh`), a cloudy atmosphere (`cloudy`), an ocean covered cloud-free planet (`ocean`), a ringed planet (`ringed`), a cloud-free Rayleigh-scattering atmosphere with absorbing methane (`methane`), and a Venus-like atmosphere (`venus`) with cloud parameters based on Hansen & Hovenier (1974). The results of the Venus-like atmosphere can be compared with observations by Coffeen & Gehrels (1969). Please also refer to projects/MODELS.md for detailed information about the planetary models.

Parameters of the model such as the grid cell structure or the density of the atmospheric particles are stored in a separate binary grid file (see Create a grid or refer to the manual Sect. 2.3 for detailed information).

To start a simulation, move into the POLARIS directory and execute `polaris` followed by the command file:

```
cd /YOUR/POLARIS/PATH/
polaris projects/rayleigh/POLARIS.cmd
```

The results are stored at `projects/rayleigh/data/` as `.fits.gz` files. These files can be opened with, for example, SAOImageDS9, or a python script using astropy. For this sample simulations, a simple python script is provided, which can be executed with

```
python projects/plot.py rayleigh
```

The output file is stored in the corresping directory, e.g. `projects/rayleigh/rayleigh.png`.

HINT: The previous results will be overwritten, if the same command file is used. Please change <path_out> in the command file to use a new directory for the new results.

HINT: If users write their own command file, before starting the simulation, please check <dust_component>, <path_grid>, and <path_out> in the command file for the correct (absolute) paths.

Default model	Runtime (8 cores)	Comment
Rayleigh	$\sim$ 2 minutes	36 phase angles, 1 wavelength, 10^7 photons
Cloudy	$\sim$ 5 minutes	36 phase angles, 1 wavelength, 10^7 photons
Ocean	$\sim$ 14 minutes	36 phase angles, 1 wavelength, 10^8 photons
Ringed	$\sim$ 16 minutes	1 phase angle, 1 wavelength, 10^9 photons
Methane	$\sim$ 23 minutes	1 phase angle, 61 wavelengths, 10^7 photons per wavelength
Venus	$\sim$ 3 hours	36 phase angles, 10 wavelengths, 10^6 photons per wavelength

Create a grid

POLARIS includes PolarisTools, a Python package to create custom grid files for POLARIS. The binary grid file can be created with the command `polaris-gen`.

Predefined models

There are already various models available (see above):

rayleigh: Cloud-free Rayleigh scattering atmosphere

cloudy: Cloudy atmosphere

ringed: Ringed cloud-free planet

venus: Venus-like cloudy atmosphere

To create a grid file, use

```
polaris-gen model_name grid_filename.dat --num_dens 1 --normalize 0
```

where `model_name` is one of the above models. The parameters of the respective model are defined in `tools/polaris_tools_modules/model.py`.

In addition, the keyword `--num_dens 1` tells PolarisTools that the density in `model.py` (see below) is a number density (instead of a mass density), and `--normalize 0` tells PolarisTools not to normalize the density distribution to a given total dust mass. The (binary) grid file will be stored at `projects/model_name/`.

HINT: The optical depth of the cloudy, ringed and Venus-like model are based on precalculated extinction cross sections at 550 nm for a given size distribution (see respective `.cmd` file and in `tools/polaris_tools_modules/model.py`).

NOTE: The **methane** and the **ocean** model are based on the **rayleigh** model.

Extra parameter

To modify further model specific parameter values, the user can parse a list of parameter values using the option `--extra` followed by the keywords and the corresponding value (int, float, or str). By default, the user can parse the following keywords for the predefined models:

rayleigh:

- `optical_depth`: total optical depth of the atmosphere (default: 1)

cloudy:

- `optical_depth`: total optical depth of the cloud layer (default: 1)

ringed:

- `optical_depth_gas`: total optical depth of the atmosphere (default: 1)
- `optical_depth_ring`: vertical optical depth of the ring (default: 1)

For example, to create a Rayleigh scattering atmosphere with an optical depth of 5, use

```
polaris-gen rayleigh grid_filename.dat --extra optical_depth 5\
    --num_dens 1 --normalize 0
```

Additional parameter values to modify the model can be defined in the function `update_parameter(self, extra_parameter)` in the file `tools/polaris_tools_modules/model.py`.

Hint: For any changes in the files, the user has to recompile PolarisTools with:

```
./compile.sh -t
```

or if compiled without the script:

```
python3 tools/setup.py install --user &>/dev/null
```

Custom model

For a more complex model modification, it is recommended that users define their own models in `tools/polaris_tools_custom/model.py`. Therein, each model is defined as a class with a corresponding entry in the dictionary at the top of `model.py`. General model parameters are defined in `__init__(self)`. The dust density distribution or the temperature are defined in `dust_density_distribution(self)` and `dust_temperature(self)`, respectively. Hereby, `self.position` is a list with three entries defining the position in the grid (x, y, z).

Similar, to create a grid file for a custom model, use

```
polaris-gen model_name grid_filename.dat --num_dens 1 --normalize 0
```

where `model_name` is the name of the model in the dictionary of `model.py`.

Hint: For any changes in the files, the user has to recompile PolarisTools with:

```
./compile.sh -t
```

or if compiled without the script:

```
python3 tools/setup.py install --user &>/dev/null
```

Convert a grid file

Users can also write and edit their own grid file. For this purpose, the command `polaris-gen` has an ascii to binary converter (and vice versa) for converting grid files. To convert an existing ascii grid file to a binary grid file, use

```
polaris-gen model_name grid_filename.txt --convert ascii2binary
```

To convert an existing binary grid file to an ascii grid file, use

```
polaris-gen model_name grid_filename.dat --convert binary2ascii
```

The input grid file has to be located in `projects/model_name/` and the new output grid file will be stored at `projects/model_name/`. For the general structure and available options in the grid file, please read the manual (Sect. 2.3).