POLARIS Quickstart Guide

Download

Download zip package from the CPC Library or clone the github repository via:

```
git clone -b DustyPlasma --single-branch https://github.com/polaris-MCRT/POLARIS.git
```

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

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

Requirements

The following packages are required for the installation:

- gcc (preferred), icc, or clang++
- cmake (preferred), or ninja
- python3 (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 cfitsio and CCfits libraries. 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 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 the simulation of the scattering of laser light in a dusty plasma can be found in `projects`.

The example simulations use exemplary (binary) grid files `example1.grid` and `example2.grid` of a homogeneous cylindrical dust cloud which can be found in `projects/constantCylinder/`.

To run the scattering simulation of example 1, move into the POLARIS directory and execute `polaris` followed by the command file:

```
cd /YOUR/POLARIS/PATH/
polaris projects/example1.cmd
```

The results are stored at `projects/constantCylinder/example1/data/` as `.fits.gz` files. These files can be opened with, for example, SAOImageDS9, or a python script using astropy.

Please refer to the command list in the `projects` folder or the manual for available options of the command file.

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.

Create a grid

The (binary) grid file can be created with the command `polaris-gen`. There is already a model **constantCylinder** of a cylindrical dust cloud with a constant particle number density available. The default values are a number density $\rho(r, z) = 10^{13}$ m^{-3}, a height $h = 3$ cm, and a radius $r = 3$ cm of the cylinder.

To create a grid file, use

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

where `model_name` is `constantCylinder`. The (binary) grid file will be stored at `projects/model_name/`. It is also possible to modify some parameters of the model. For example, to create a grid with an outer radius of 1 cm, type:

```
polaris-gen model_name grid_filename.dat --outer_radius 0.01m
```

For more information, type:

```
polaris-gen -h
```

Extra parameter

To modify further model specific parameter values, the user can parse a list of parameter values using the option `--extra` followed by a list of values (int, float, or str). By default, the user can parse one value for the `constantCylinder` model: the number density.

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

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

```
./compile.sh -u
```

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

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

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 with:

```
./compile.sh -u
```

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.