POLARIS Quickstart Guide

Download

Download zip package from the homepage or clone the github repository.

Requirements

The following packages are required for the installation:

- gcc (preferred), icc, or clang++
- cmake (preferred), or ninja
- python3 and numpy

Installation

- Extract the zip file
- 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.
For more information, type:

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

Start a simulation

POLARIS is shipped with a prebuild grid and command files to perform example simulations.
The grid (binary) file `grid.dat` of an example disk can be found in `projects/disk/`.
The command files `.cmd` of the temperature, thermal emission, and scattered stellar emission can be found in

- `projects/disk/example/temp/`,
- `projects/disk/example/dust/`, and
- `projects/disk/example/dust_mc/`, respectively.

Before starting the simulation, change `/YOUR/POLARIS/PATH/` in the command file at `<dust_component>`, `<path_grid>`, and `<path_out>` to your POLARIS path.

To start a temperature simulation, type:

```
polaris projects/disk/example/temp/temp.cmd
```

The results are stored at `projects/disk/example/temp/data/` as `.fits` files. These files can be opened with, for example, SAOImageDS9.

Similar, the simulation for thermal emission and scattered stellar emission are performed.
For available options in the command file, please read the manual.

Create a grid

Predefined models

The grid (binary) file will be created with the command `polaris-gen`.
There are already three models available:

- disk: A circumstellar disk with a Shakura & Sunyaev density distribution (Lynden-Bell & Pringle 1974; Hartmann et al. 1998)

$$\rho(r, z) = \rho_0 \left(\frac{r}{r_0}\right)^{-\alpha} \times \exp\left[-\frac{1}{2}\left(\frac{z}{h(r)}\right)^2\right]$$

$$h(r) = h_0 \left(\frac{r}{r_0}\right)^{\beta}$$

  Here, the default values are $r_0 = 100\,\mathrm{AU}$, $h_0 = 10\,\mathrm{AU}$, $\alpha = 0.9$, and $\beta = \frac{2\alpha+3}{6} = 0.8$.
- globule: A Bok globule with a Bonnor-Ebert sphere density distribution (Harvey et al. 2001; Kaminski et al. 2014)

$$\rho(r) = \rho_0 \begin{cases} r_\mathrm{t}^{-2} & \text{if } r \leq r_\mathrm{t} \\ r^{-2} & \text{if } r_\mathrm{t} < r \end{cases}$$

  Here, the default value is $r_\mathrm{t} = 10^3\,\mathrm{AU}$.
- sphere: A sphere with a constant density distribution

$$\rho(r) = \rho_0$$

By default, the density distribution is normalized to the given total mass.
To create a grid file, use

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

where `model_name` is either `disk`, `globule`, or `sphere`.
The grid file will be stored at `projects/model_name/`.
It is also possible to modify some grid parameters with the command `polaris-gen`.
For more information, type:

```
polaris-gen -h
```

Extra parameter

To modify further 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). These additional parameter values can be used in the function `update_parameter` in the file `model.py` to vary the model.
For example, the user can parse

- 4 values for the `disk` model: reference radius r_0, reference scale height h_0, α, and β,
- 1 value for the `globule` model: truncation radius r_t, and
- 1 value for the `sphere` model: the geometry of the magnetic field (toroidal, vertical, or radial).

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

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

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`.

Write a grid file

It is also possible, to write their own grid file.
For the general structure and available options in the grid file, please read the manual.
For this purpose, the command `polaris-gen` has an ascii to binary converter (and vice versa) for the grid files.
To convert an existing ascii grid file to a binary grid file, use

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

The ascii file has to be located in `projects/model_name/` and the new binary grid file will be stored at `projects/model_name/`.