

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.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

- 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 (binary) grid 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/POLARIS.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 (binary) grid 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 $r_0 = 100$ AU, $h_0 = 10$ AU, $\alpha = 0.9$, and $\beta = 1.1$.

- globule: A Bok globule with a [Bonnor-Ebert](#) sphere density distribution ([Harvey et al. 2001](#); [Kaminski et al. 2014](#))

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

Here, the default value is $r_t = 10^3$ 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 (binary) 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). By default, 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).

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 --convert ascii2binary model_name grid_filename.txt
```

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

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

The input 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.