DART-Ray

User guide

Index

Chapter 1

A short introduction to 3D dust radiative transfer

1.1 What dust radiative transfer is

Many astrophysical objects are composed by producers of radiation, such as stars, and a diffuse medium which modifies the specific intensity of the emitted light through four different mechanisms:

- Absorption of radiation;
- Scattering - the deflection of radiation in a different direction;
- Remission of the absorbed radiation at longer wavelengths;
- Polarization of the scattered/re-emitted radiation.

The interstellar medium is composed of gas and dust, which both affect the radiation field intensity through the above mechanisms. However, the dust is usually the main modifying agent of the emission from sources of ultraviolet (UV) to near-infrared (NIR) radiation and the dominant producer of emission in the infrared wavelength range. For this reason, a branch of radiative transfer (RT) modelling of astrophysical objects in the UV to submm ranges, called dust radiative transfer, is dedicated exclusively to the numerical calculation of the effect of dust on the radiation field and the prediction of its infrared emission, ignoring the presence of gas.

In the following, we will consider all the above mechanisms with the exception of polarization which is not included in the DART-Ray code. For a more extended review on 3D dust RT, see Steinacker, Baes, and Gordon 2013.

1.2 The 3D dust RT equation

Given an arbitrary 3D distribution of radiation source luminosity per unit volume (also called volume emissivity) and of dust density, as well as the dust optical parameters, the classical approach to solving the radiation transfer problem is to determine the radiation field specific intensity $I_\lambda(\vec{x},\vec{n})$, which gives the luminosity per unit area, per unit wavelength/frequency interval, per unit solid angle, propagating along the direction $\vec{n}$ at the position $\vec{x}$. By calculating $I_\lambda(\vec{x},\vec{n})$ for all possible $\vec{n}$ and $\vec{x}$, one is able to predict the intensity of the emission detected by an observer at arbitrary positions as well as the radiation field energy density (RFED) distribution $U_\lambda(\vec{x})$ which is related to $I_\lambda(\vec{x},\vec{n})$ by the formula:

$$U_\lambda(\vec{x}) = \frac{\int I_\lambda(\vec{x},\vec{n})d\Omega}{c} \tag{1.1}$$

with the integration on the solid angle Ω carried over the full 4π solid angle and c being the speed of light[1]. The RFED is an essential quantity to derive the dust emission spectra as well

[1] This formula can be seen as $U_\lambda = \frac{F_\lambda}{c}$ with F_λ being the radiation flux (luminosity per unit surface).

as an important physical parameter by itself (e.g. to model the interaction of cosmic rays with the interstellar radiation field).

$I_\lambda(\vec{x},\vec{n})$ depends on six independent variables: three spatial coordinates, two angles defining the direction $\vec{n}$ and one variable for the wavelength/frequency. In principle, the values of $I_\lambda(\vec{x},\vec{n})$ can be found by solving a differential equation, determining the spatial variation of $I_\lambda(\vec{x},\vec{n})$ along an arbitrary direction n, given the appropriate boundary conditions. In 3D dust RT, the differential equation to consider is the following:

$$\vec{n}\nabla_{\vec{x}} I_\lambda(\vec{x},\vec{n}) = -k_\lambda \rho(\vec{x}) I_\lambda(\vec{x},\vec{n}) + j_\lambda(\vec{x}) + j_{\lambda,sca}(\vec{x},\vec{n}) + j_{\lambda,d}(\vec{x}) \tag{1.2}$$

with k_λ the extinction coefficient (taking into account both absorption and scattering), $\rho(\vec{x})$ the dust density and the j_λ functions being three different source terms (in units of luminosity per volume per solid angle per wavelength/frequency interval):

- $j_\lambda(\vec{x})$: a source term representing the emission from sources emitting isotropically in all directions. This term depends only on $\vec{x}$ but neither on n nor on $I_\lambda(\vec{x},\vec{n})$, unlike the following two source terms. It can be used to specify the location and luminosity of extended stellar distributions or point sources.

- $j_{\lambda,sca}(\vec{x},\vec{n})$: a source term representing the scattered luminosity emission distribution. It can be expressed as:

$$j_{\lambda,sca}(\vec{x},\vec{n}) = \omega_\lambda k_\lambda \rho(\vec{x}) \int_\Omega \Phi_\lambda(\vec{n},\vec{n'}) I_\lambda(\vec{x},\vec{n'}) d\Omega' \tag{1.3}$$

  where ω_λ is the albedo, defined such that $\omega_\lambda k_\lambda$ gives the fraction of extinction due to scattering, and $\Phi_\lambda(\vec{n},\vec{n'})$ is the scattering phase function, which gives the probability for radiation coming from direction $\vec{n'}$ to be scattered into direction $\vec{n}$. As it is clear from the formula above, this source term depends on $I_\lambda(\vec{x},\vec{n})$ at the same position $\vec{x}$ and for all possible directions $\vec{n'}$. Because of this dependency, this source term cannot be set as an input-defined distribution but it has to be found while solving equation 1.1.

- $j_{\lambda,d}(\vec{x})$: a source term representing the emission from the interstellar dust. This emission depends on the luminosity absorbed by the dust and the dust optical properties and size distribution. In the case of thermal equilibrium between dust particles and the radiation field heating them, each dust particle a of chemical species i is characterized by a certain equilibrium temperature $T(a,i)$ (see section 1.7). In this case, the source term for a grain mixture can be written as:

$$j_{\lambda,d}(\vec{x}) = \sum_i \int_a \pi a^2 n_d(a,i) Q_{\lambda,abs} B_\lambda(T(a,i)) da \tag{1.4}$$

  where $n_d(a,i)$ is the grain number density and $Q_{\lambda,abs}$ is the grain absorption coefficient (absorption cross section divided by πa^2) and B_λ the Planck function.

  Grains cannot reach equilibrium with the radiation field when they are heated by photons whose energies are comparable with their internal enthalpies. In this regime, grains experience temperature fluctuations through a process called stochastic heating (although they still re-emit the total absorbed luminosity when their emission is averaged on sufficiently long time scales). In this case, one has to derive the temperature probability distribution $P(T(a,i))$ of the grains using a numerical method (see section 1.7). Then the source term can be written as:

$$j_{\lambda,d}(\vec{x}) = \sum_i \int_a \int_T \pi a^2 n_d(a,i) Q_{\lambda,abs} B_\lambda(T(a,i)) P(T(a,i)) da dT \tag{1.5}$$

  Both the calculations of the equilibrium temperature and of the temperature probability distribution require the knowledge of the RFED (see section 1.7) which in turn is dependent on $I_\lambda(\vec{x},\vec{n})$ integrated over all possible directions $\vec{n}$ (see equation 1.1). Therefore, as for the scattering source term, this source term has to be derived while solving equation 1.2.

We note that, since the RT equation is additive in nature, one can calculate the specific intensities of the stellar emission and dust emission separately and then add them together. The stellar emission spectral range usually spans the UV to NIR range ($\sim 0.09 - 5\mu$m), so only these wavelengths need to be considered in the RT calculation. Instead, the dust emission is mainly in the IR range ($\sim 1 - 1000\mu$m), with the NIR range the only region of overlap.

1.3 Graphical explanation of the different terms in the 3D dust RT equation

The 3D dust RT equation can appear rather obscure if described only using the mathematical expressions in the previous section. However, all the terms contained within it correspond to processes that can be described graphically in a relatively easy way. The processes are:

- The production of stellar light;
- the absorption of stellar radiation;
- the scattering of stellar radiation;
- the re-emission of absorbed stellar radiation as dust emission;
- the absorption of dust emission and its re-emission as dust emission (known as "dust self-heating");
- the scattering of dust emission.

The 3D dust RT equation quantifies the effect of these processes in changing the specific intensity of the radiation propagating along a certain direction $\vec{n}$. Let's begin with the simplest case where there are neither stellar emission sources nor dust particles along the direction $\vec{n}$ (see figure 1.1). In this case, all terms on the RHS of the equation 1.2 are zero. Thus, there is no variation of the specific intensity of the radiation propagating along the direction $\vec{n}$ between the starting and final points $\vec{x_o}$ and $\vec{x_1}$. However, note that while the specific intensity I_λ is always conserved in this case, in general the flux F_λ is not (e.g. when the radiation is produced by a point source $F_\lambda \propto r^{-2}$)!

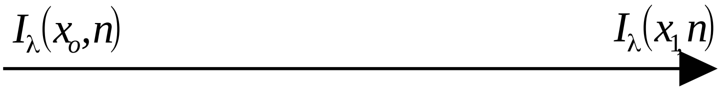

Figure 1.1: No stellar sources and no dust along the direction $\vec{n}$. In this case, the specific intensity along $\vec{n}$ does not vary from point x_o to x_1

If there are only stellar emission sources along n, the only term present on the RHS of equation 1.2 is $j_\lambda(\vec{x})$. This term has the effect of increasing the radiation specific intensity propagating along the direction $\vec{n}$ (see figure 1.2).

If we add dust particles along the same line, this has two effects. The first effect, sometimes called "extinction", is that radiation is partially absorbed and partially scattered in all directions (see figure 1.3). The decrease of specific intensity due to this effect is described by the term $-k_\lambda \rho(\vec{x}) I_\lambda(\vec{x}, \vec{n})$. Note that the relative decrease of intensity over a small segment $ds = \vec{n} d\vec{x}$ is proportional to $d\tau_\lambda = k_\lambda \rho(\vec{x}) ds$, with τ_λ a quantity referred to as "optical depth" which is proportional to the column density of dust along n.

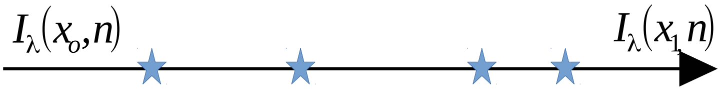

Figure 1.2: Stellar sources but no dust along the direction $\vec{n}$. In this case, the specific intensity along $\vec{n}$ increases from point x_o to x_1. This process is described by the source term $j_\lambda(\vec{x})$.

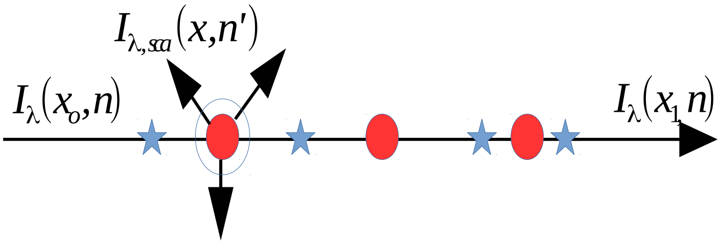

Figure 1.3: Stellar sources and dust along the direction $\vec{n}$. In this case, the specific intensity along $\vec{n}$ tends to be increased by the presence of stars but also decreased because of the absorption and scattering due to dust particles. The latter effect is described by the extinction term $-k_\lambda \rho(\vec{x}) I_\lambda(\vec{x}, \vec{n})$.

The second effect is also due to scattering, but it produces an increase of the specific intensity, as opposed to a decrease. This occurs when incoming radiation is scattered by dust into the direction $\vec{n}$ (see figure 1.4). This effect is described by the source term $j_{\lambda,sca}(\vec{x}, \vec{n})$. As it can be seen from equation 1.3, the amount of radiation scattered into direction $\vec{n}$ depends on the scattering phase function $\Phi_\lambda(\vec{n}, \vec{n'})$ (see section 1.5), as well as the dust density and the radiation specific intensity coming from all directions $\vec{n'}$.

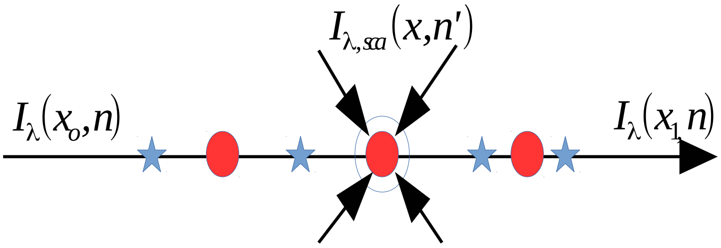

Figure 1.4: Stellar sources and dust along the direction $\vec{n}$. Apart from the extinction, reducing the radiation intensity, the presence of dust can also increase the intensity if radiation coming from other directions $\vec{n'}$ is scattered by dust into the direction $\vec{n}$. This effect is described by the scattering source function term $j_{\lambda,sca}(\vec{x}, \vec{n})$.

Finally, the radiation luminosity absorbed by dust is then re-emitted in the infrared range. As for the stellar emission, this radiation can also be absorbed and scattered by dust particles. This means that, as well as heating due to stellar emission, dust particles can, in principle, experience additional heating from the absorption of photons emitted by dust located elsewhere (see figure 1.5). Therefore, in opposition to the radiation transfer of stellar emission where the source luminosities are unaffected by the radiation field, in the case of dust emission radiation transfer the radiation field affects the production of dust emission in a complex non-linear and non-local way. The production of dust emission is described by the source term $j_{\lambda,d}(\vec{x})$. The extinction and scattering of dust emission are described by the corresponding terms we saw before, which affect both the dust and stellar emission in the same way.

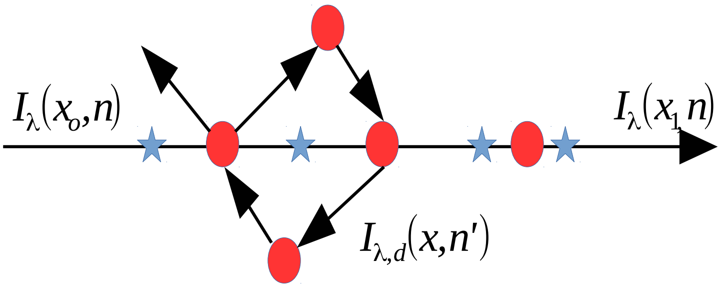

Figure 1.5: Dust re-emits the absorbed luminosity in the infrared range. However, while propagating this radiation can also be absorbed and scattered by dust located at other positions. This in turn affects the dust emission intensity at all positions.

1.4 The stellar emission distribution

In order to perform dust RT calculations, the stellar emission distribution has to be specified to determine the values of $j_\lambda(\vec{x})$ at each position and wavelength. The simplest spectral shape that can be assumed is that of a black body, given a certain effective temperature and bolometric luminosity. This is a common approximation to model single stars. Alternatively, one could use more accurate spectral energy distributions (SED) derived from theoretical models of single stars or measured observationally. However, when stellar populations have to be modelled, one should use integrated spectra of stellar population synthesis models which provide the combined emission of a set of stars. For example, stellar particles in an N-body simulation can be modelled as single age stellar populations (SSP). Complex emission spectra can be obtained by combining the emission from stars with different ages, with several assumptions on the initial mass function and star formation history. Performing radiation transfer calculations of the stellar emission is important both for determining the appearance of an RT model to an observer placed in a particular position and for calculating the RFED at UV to NIR wavelengths, needed to calculate the dust heating. In order to calculate the dust heating with good accuracy, it is typically necessary to specify the emission from stars for at least 15-20 wavelengths spread in the UV-NIR wavelength range.

1.5 The scattering phase function

Dust scatters some of the extincted luminosity in all directions. In the 3D dust RT equation, this effect is taken into account by the source term $j_{\lambda,sca}$ which requires the knowledge of the phase function $\Phi(\vec{n},\vec{n'})$, which gives the probability of angular redistribution of scattered luminosity as a function of the propagation direction n' and the scattered direction n. The vast majority of dust radiation transfer codes uses the Henyey-Greenstein phase function (Henyey and Greenstein 1941):

$$\Phi(\theta) = \frac{1}{4\pi} \frac{1 - g_\lambda^2}{[1 + g_\lambda^2 - 2g_\lambda \cos(\theta)]^{3/2}} \tag{1.6}$$

with θ the angle between the n and n', and g_λ the anisotropy parameter which determines the width and the direction of the scattered light (see figure 1.6).

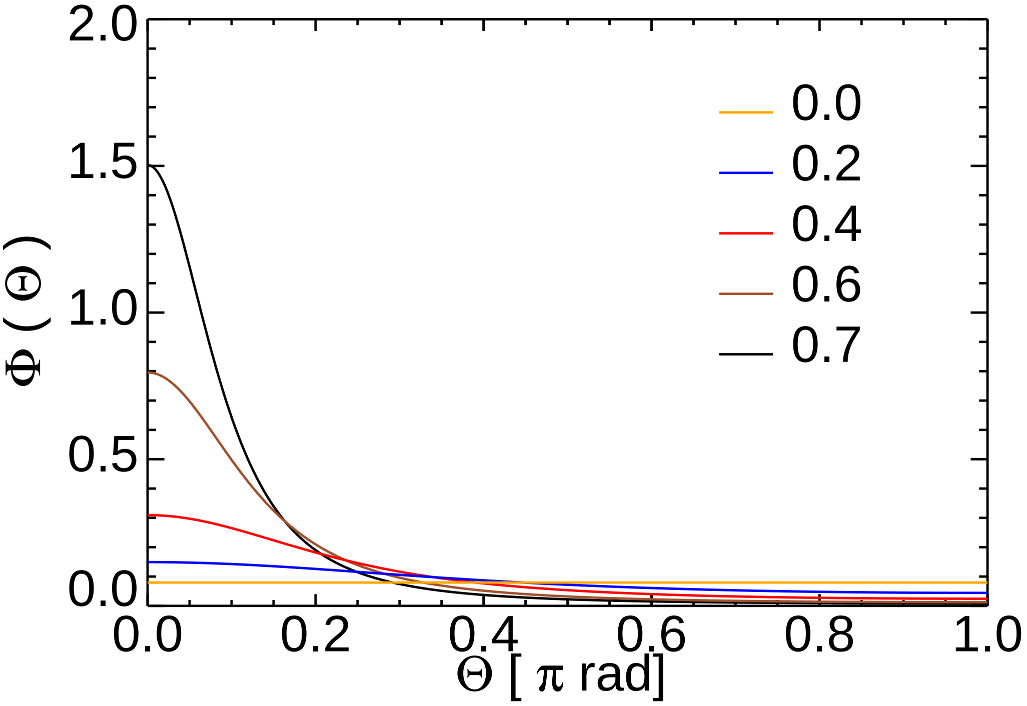

Figure 1.6: Henyey-Greeinstein phase function vs the angle θ between the propagation direction n' and the scattered direction n for several values of the anisotropy parameter g_λ.

The value of the anisotropy parameter g_λ to be used depends on the assumed dust model (see section 1.6). Values in the range [-1,1] are allowed in principle, with $g_\lambda = 1$ meaning all the scattering in the forward direction, $g_\lambda = 0$ isotropic scattering, and $g_\lambda = -1$ all scattering in the backward direction. Typically, for interstellar dust, scattering is mainly forward for short wavelengths and it becomes more and more isotropic going towards long infrared wavelengths.

1.6 The dust density distribution and average opacity coefficients

In equation 1.2 the dust density $\rho(\vec{x})$ is always multiplied by the extinction factor k_λ, and the important quantity that has to be specified at each position to define the dust distribution is their product: $k_\lambda \rho(\vec{x})$. Therefore, the amount of dust per unit volume can be expressed in different ways, depending on the units used for k_λ. In some instances, the dust mass density is specified with k_λ in units of cross section per dust mass. Other times the dust number density is used, with k_λ in units of cross section per grain. However, there are also times when the gas density (instead of the dust density) is specified with k_λ in units per gas mass or per gas atom (e.g. per Hydrogen). A further additional way to specify the dust spatial distribution is by setting the total optical depth $\tau_\lambda = \int k_\lambda \rho(\vec{x}) d\vec{x}$ at a certain wavelength along some particular direction of the RT model and by specifying the functional form of the dust density distribution. In this way, one can derive the distribution of $k_\lambda \rho(\vec{x})$ at all positions, without necessarily calculating $\rho(\vec{x})$.

In summary, there are many different ways to set the dust distribution and its extinction efficiency in a radiation transfer problem, but all boil down to the determination of the product $k_\lambda \rho(\vec{x})$ at a certain reference wavelength for all points in the grid used in the calculation. This is sufficient for performing stellar emission RT calculations. However, when dust emission RT calculation have to be performed, one needs to know also the number density of grains $n_d(a, i)$ at each position in order to calculate the dust emission (see equations 1.4 and 1.5). If not specified directly, the grain number density can be derived from the product $k_\lambda\rho(\vec{x})$, the value of k_λ and the grain size distribution (with appropriate unit transformations when needed).

Once the extinction term $k_\lambda\rho$ has been set for one reference wavelength, the corresponding values at all other wavelengths simply reflect the variation of k_λ from the reference wavelength to another (assuming that the dust optical properties do not vary with position). The wavelength dependence of k_λ is determined by the assumed "dust model", that is, a certain mix of dust particle species of different chemical compositions, each with a certain grain size distribution. Several dust models have been derived in the literature which allow to reproduce observational constraints such as the extinction curve of the Milky Way (see e.g. Zubko, Dwek, and Arendt 2004). The same models also provide the albedo ω_λ as well as an average scattering anisotropy parameter $< g_\lambda >$.

The opacity parameters provided for a dust model represent the average opacity coefficients that have to be used in the 3D dust RT equation. These parameters are appropriate averages which can be derived from the opacity parameters of the single grains and the grain size distribution. In fact, single grain opacities are usually expressed in terms of the parameters $Q_{\lambda,\mathrm{abs}}$ and $Q_{\lambda,\mathrm{sca}}$ which are the absorption and scattering cross sections divided by the grain geometrical cross section πa^2, as well as the anisotropy parameter $g_\lambda(a,i)$. The total extinction efficiency is $Q_{\lambda,\mathrm{ext}} = Q_{\lambda,\mathrm{abs}} + Q_{\lambda,\mathrm{sca}}$. Theoretically derived values for these coefficients are available in the literature, mainly for spherical grains (see e.g. Laor and Draine 1993 and Bruce Draine's website: https://www.astro.princeton.edu/ draine/dust/dust.diel.html). Then, the average values for k_λ (in units of cross section per grain), ω_λ and g_λ can be derived using the following formulae:

$$k_\lambda = \frac{\sum_i \int \pi a^2 Q_{\lambda,\mathrm{ext}}(a,i) n_d(a,i) da}{\sum_i \int n_d(a,i) da} \tag{1.7}$$

$$k_{\lambda,\mathrm{sca}} = \frac{\sum_i \int \pi a^2 Q_{\lambda,\mathrm{sca}}(a,i) n_d(a,i) da}{\sum_i \int n_d(a,i) da} \tag{1.8}$$

$$\omega_\lambda = k_{\lambda,sca}/k_\lambda \tag{1.9}$$

$$< g_\lambda >= \frac{\sum_i \int \pi a^2 Q_{\lambda,\mathrm{sca}}(a,i) g_\lambda(a,i) n_d(a,i) da}{\sum_i \int \pi a^2 Q_{\lambda,\mathrm{sca}}(a,i) n_d(a,i) da} \tag{1.10}$$

Note that, in order to calculate the dust heating and emission, the absorption coefficient $Q_{\lambda,\mathrm{abs}}$ has to be known for each grain size and composition (see equations 1.4 and 1.5 and next section).

1.7 Calculation of the dust temperature

The calculation of the dust temperature and emission spectra should be performed for each grain size and chemical composition, and not for a single grain with average properties. This is because grains vary considerably in their opacity parameters and their equilibrium temperature can span a wide range of values for a given radiation field spectra.

In the simple case of equilibrium between the dust particle and the radiation field, the dust temperature T of a single grain is the one for which the absorbed luminosity is balanced by the emitted luminosity. This condition can be expressed by the following equation, with the LHS proportional to the emitted luminosity and the RHS proportional to the absorbed luminosity per unit grain surface area:

$$\int Q_{\lambda,abs}(a,i) B_\lambda(T) d\lambda = (c/4\pi) \int Q_{\lambda,abs}(a,i) U_\lambda d\lambda \tag{1.11}$$

So, once the value of U_λ is known, one needs to find the right temperature T that balance the above equation for each grain size a and grain species i. When U_λ is dominated by stellar emission radiation, this has to be done only once. When the contribution of the dust emission to U_λ is not negligible, one needs to repeat this calculation several times during an RT calculation, until both U_λ and T reach convergence.

If one wants to perform the full stochastic heating calculation, one has to derive for each grain size and composition the temperature probability distribution $P(T)$. Given a RFED spectra, the dust optical properties as well as the grain heat capacities, this can be done following numerical methods such as the Guhathakurta and Draine 1989 method. In DART-Ray we use this method combined with the analytical solutions of Voit 1991. Calculation of the stochastical heating is important especially for the MIR range, where often dust emission is dominated by stochastically heated small grains reaching high temperatures rather than warm dust at equilibrium with the radiation field.

1.8 Methods to solve the 3D dust radiation transfer problem

The 3D dust radiation transfer equation is a compact description of the RT problem. However, it is a non-local non-linear integro-differential equation, and it is generally difficult to solve directly. Any numerical solution method requires multiple iterations to derive the scattering and dust emission source terms which are unknown at the beginning of the calculation. Furthermore, the problem contains six independent variables, and the memory requirements for storing the complete solution for I_λ could be beyond the capability of modern computer RAM memories of order of hundreds of Gbytes. Because of these difficulties, most of the available 3D dust RT codes do not try to provide solutions for I_λ at all grid points but rather use a Monte-Carlo (MC) method to derive the specific intensity of radiation arriving to the observer. The MC method is based on tracing the propagation of "photon particles" through a radiation transfer model using a probabilistic approach. Several probability functions, constructed on the base of the input stellar emissivity distribution, dust density and scattering parameters, are used to determine the initial position of a photon, its initial propagation direction, the position of the following scattering events and its further propagation until it is either absorbed or leaves the RT model. This method in its classical form is actually not efficient in producing images as they would be viewed by an observer. However, when combined with a ray-tracing technique called "peeling off", this method becomes much quicker. Other acceleration techniques have been developed to speed up the calculation of the absorbed luminosity throughout the system and therefore handle the dust emission RT as well. Despite the apparent differences, all processes taken account in Monte-Carlo codes are the same as those described by the RT equation.

In DART-Ray the 3D RT equation is also not solved directly for I_λ at all positions and directions. Instead the code uses ray-tracing operations to calculate the RFED, the scattering source function, the dust emission source function and the specific intensity of the radiation received by observers. The RFED is used to derived the dust heating and emission, which does not require the knowledge of I_λ but only its integration over the full solid angle at each spatial position. Determining the source functions for the scattering and the dust emission is sufficient to derive the specific intensity at the observer positions, since the latter can be derive by integrating equation 1.2 along the observer line of sight. Also, storing the scattering source function is less demanding in terms of memory compared to I_λ since the width of any angular variation on the scattering source function is regulated by the scattering phase function. Therefore, at longer wavelengths when scattering gradually becomes more isotropic, relatively few angular directions have to be sampled for a sufficiently accurate reconstruction of $j_{\lambda,\mathrm{sca}}$ at any angular direction. Instead, the I_λ angular behaviour can be unpredictably fast, since it depends not only on the scattering source functions but also on the radiation source and dust distributions.

Chapter 2

The DART-Ray RT algorithm

2.1 Model discretization and ray-tracing from radiation sources

Before describing the algorithm used in DART-Ray, it is worth introducing a few concepts that will aid the reader in understanding the rationale for the code. The starting point is the discretization of a RT model in a grid of rectangular cells. Each cell is characterized by an average volume emissivity j_λ and extinction coefficient $k_{\lambda,ext}\rho(\vec{x})$. In order to calculate the RFED and the scattering source function in each cell of the discretized model, a very basic (and very inefficient) way to do it would be to perform RT calculations from each source throughout the entire model (see figure 2.1). This involves calculating the variation of the specific intensity of the radiation produced by each source along a numerous set of directions, usually referred as "rays"[1]. The number of ray directions to consider should be high enough such that all cells in the RT model are crossed by at least a few rays.

Since all the sources are considered separately, the radiation transfer calculation for a single ray takes into account only the extinction of the initial specific intensity $I_{\lambda,0}$ along a ray. This is simply calculated as:

$$I_\lambda(\tau) = I_{\lambda,0}e^{-\tau_\lambda} \tag{2.1}$$

with τ_λ the optical depth crossed by the ray. This means that the emission produced by the cells crossed by a ray is not taken into account immediately, unless the cell is the one from where the ray originates. Instead, the emission produced in the other cells is considered later. When a ray crosses a cell, it adds a contribution δU_λ to the local RFED as well as a contribution $\delta j_{\lambda,\mathrm{sca}}$ to the local scattering source function. In this way, after all rays originated by a certain source have been followed throughout a model, one has added the contribution of that source to the RFED and scattering source function everywhere in the model. After that, one can process the rays originating from a different source and so on until all sources have been considered.

The same procedure can be used for processing the scattered light. In this case, the source luminosity is determined by the scattering source function. However, since light can be scattered multiple times, one needs to repeat the loop on the scattered light sources several times in order to include all non negligible orders of scattered light.

Finally, one can process the sources of dust emission. These sources undergo the same looping procedure although this has to be repeated several times because the dust emission depends on the RFED and viceversa. So, multiple "dust heating" iterations have to be performed until the dust emission spectra converges at all positions.

2.2 The source influence volume

The "source-to-cell" approach described above can be improved when the radiation sources do not contribute significantly to the RFED everywhere in the RT model but only within a limited region which we call the "source influence volume". The extent of this volume for each source

[1] following the intensity variation along a ray is also called "ray-tracing".

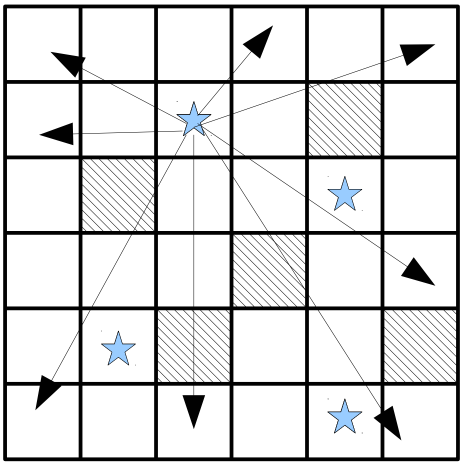

Figure 2.1: Ray-tracing from a source to all cells of the model grid.

depends on the overall stellar emission and dust distributions but also on the numerical accuracy that is desired. To reach infinitely high accuracy, the source influence volume always coincides with the entire model. However, the source influence volume is only a fraction of the entire model when it is sufficient to calculate the RFED to an accuracy of a few percent.

If one knew in advance the extent of the influence volumes around each source, it would be possible to perform ray-tracing calculation only within it and, thus, this would substantially reduce the amount of calculation needed. Unfortunately, since the RFED is one of the calculation output, one can not determine their exact extent before the RT calculation. However, during an RT calculation, one can try to figure out when a ray has clearly passed the border of the source influence volume.

In fact, one way to do this is by comparing the contribution δU_λ to the RFED carried by a ray and a lower limit $U_{\lambda,LL}$ of the RFED at the ray current position. If the RFED contribution is negligible compared to the RFED lower limit, then it is likely that its contribution will be negligible also at larger distances from the source. Mathematically, this condition can be expressed as:

$$\delta U_\lambda < f_U \times U_{\lambda,LL} \tag{2.2}$$

with the parameter f_U being low enough such that the ray intensity can be considered negligible. When condition 2.2 is satisfied for a ray, then the ray can be blocked and the algorithm can move to the next task. However, in order to use it, one needs a way to estimate a lower limit to the RFED as well as set an appropriate low value for f_U . How to determine the RFED lower limit will be shown in the next section. The value of f_U can be determined using the following formula (see Natale et al. 2017 XXX add reference when available XXX):

$$f_U \leq \frac{a_{\rm RT}}{N_s * 0.25} \tag{2.3}$$

with $a_{\rm RT}$ the numerical accuracy that one wish to reach in the calculation of the RFED and N_s the total number of radiation sources.

2.3 The three steps of the DART-Ray RT algorithm

For both the stellar emission RT and each dust heating iteration of the dust emission RT, the DART-Ray algorithm is subdivided in three steps:

1. the determination of a lower limit $U_{\lambda,\rm LL}(\vec{x})$ to the RFED distribution $U_\lambda(\vec{x})$;

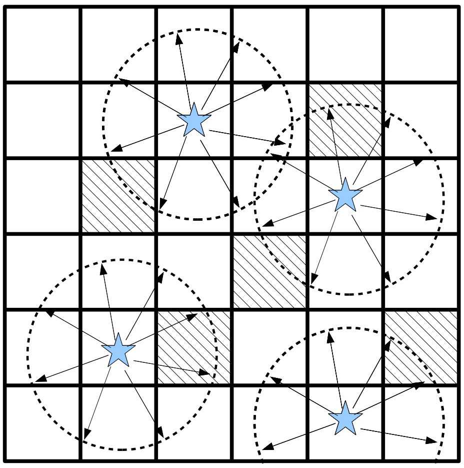

Figure 2.2: Calculation of the lower limit for the RFED. The ray tracing calculation from each source is performed only within an input defined volume. The resulting RFED is a lower limit because the contributions of the sources outside these volumes has not been considered.

2. the processing of radiation originated directly from radiation sources;

3. the processing of radiation scattered by dust.

In all these steps, the DART-Ray algorithm considers one radiation source at a time, where a source can be either a cell whose stellar or dust volume emissivity is not zero (hereafter, an "emitting cell") or a point source. The code calculates the contributions of the radiation emitted by each source to the RFED within a certain volume surrounding them. In steps 2 and 3 the contributions to the scattering source function $j_{\lambda,sca}(\vec{x},\vec{n})$ for each cell within this volume are also calculated.

During step 1, the volume considered around each source has a fixed extent chosen by the user (typically 10-20% of the entire model size). In this way, a lower limit of the RFED distribution $U_{\lambda,\mathrm{LL}}(\vec{x})$ is derived because the RFED contributions in the regions beyond these volumes are not taken into account (see figure 2.2).

In step 2, the ray-tracing calculation is performed once again from the beginning but this time the rays originating from the emitting cells are blocked if the ray contribution δU_λ to the local RFED is "negligible" at all wavelengths according to the criteria 2.2 (see figure 2.3).

Finally, during step 3 the scattered radiation stored within the dusty cells is processed. In opposition to step 2, the radiation produced by emitting cells is typically direction-dependent, since the assumed scattering phase function (Henyey-Greenstein) is in general not isotropic (see section 1.5). Nonetheless, apart from few technical differences, the calculation during this step proceeds essentially as in step 2[2]. Since scattered radiation can be scattered multiple times, several scattering iterations are needed. These iterations are stopped when the remaining scattered radiation luminosity waiting to be processed is only a small fraction f_L of the total scattered luminosity of the model as found at the end of step 2.

Although the above description of the algorithm seems applicable only to mono-chromatic calculations, its actual implementation in DART-Ray is multi-wavelength. That is, the rays carry specific intensity at many wavelengths simultaneously. The ray-stopping criteria is checked for each wavelength. If it is fulfilled only by some wavelengths but not others, the ray is not stopped but only the wavelengths for which the ray intensity is not negligible are considered further in the ray propagation.

[2] One important difference is that the value of $U_{\lambda,\mathrm{LL}}(x)$ is updated firstly with the RFED distribution found at the end of step 2 and then with that found at the end of each scattering iteration.

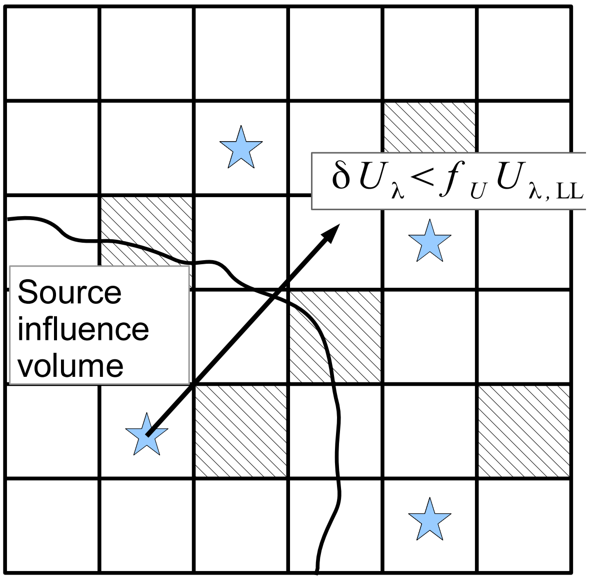

Figure 2.3: Blocking of a ray passing the border of the source influence volume.

2.4 Surface brightness maps

One major goal of RT calculation is the prediction of how astrophysical objects appear when seen along different lines of sight. This can be done with DART-Ray for observers located far away from the models as well as for observers located inside the models. The latter capability is not only for the creation of images to be used in public talks but it can be of scientific use for the radiation transfer modelling of our own galaxy.

Once the algorithm described in the previous section has been executed, all the source functions j_λ, $j_{\lambda,\mathrm{sca}}$ and $j_{\lambda,\mathrm{d}}$ have been calculated at all positions. This makes it possible to derive the variation of the specific intensity along lines directed to the observer position and thus calculated surface brightness maps as they would be detected by the observer. To ensure that all emitting cells and point sources are considered in this procedure, these ray-tracing calculations are performed once again for each source separately. That is, one first derives the initial value of the specific intensity of the radiation departing from a cell $I_{\lambda,0}$. This can be derived from the source functions as:

$$I_{\lambda,0} = \frac{(j_\lambda + j_{\lambda,\mathrm{sca}} + j_{\lambda,\mathrm{d}})\Delta r}{\Delta\tau_\lambda}(1 - e^{-\Delta\tau_\lambda}) \tag{2.4}$$

with Δr and $\Delta\tau_\lambda$ the size and optical depth of the cell respectively. Then, the specific intensity of the radiation arriving to the observer is simply:

$$I_{\lambda,obs} = I_{\lambda,0}e^{-\tau_\lambda} \tag{2.5}$$

with τ_λ the optical depth between the emitting cell and the observer position. The code calculates $I_{\lambda,obs}$ for all cells. Then, it uses volume rendering techniques to appropriately distribute the emission originating from each cell on the surface brightness maps.

Chapter 3

Installing DART-Ray

3.1 Pre-requirements

DART-Ray requires the following packages to be already installed:

- a recent version of a Fortran 2003 compiler, such as ifort 14 or later;
- an MPI installation (e.g. OPENMPI or MPICH3);
- HDF5 with the fortran bindings and the option "–enable-fortran2003" when configuring the installation (https://support.hdfgroup.org/downloads/index.html).

Please install the above packages on your machine, following the instructions in the corresponding documentation files. If the installations have been successful, you should be able to print to your unix terminal the path of the commands "mpif90" and "h5fc" using "which mpif90" and "which h5fc".

3.2 Installation

To install DART-Ray, you need to download the latest version from the DART-Ray repository on github. You can either visit the webpage

```
https://github.com/gnatale/DART-Ray
```

and download the code as a ZIP archive, or clone the repository by typing on your terminal:

```
git clone https://github.com/gnatale/DART-Ray.git.
```

DART-Ray uses a simple Makefile to compile all the programs which are part of the package. Firstly, you have to open "Makefile" using a text editor and change the Fortran compiler if required. Optionally, you can also add compiling options which are more adequate to your particular machine. Then, on the terminal, simply type:

```
make
```

to compile all the DART-Ray programs. From the list in the makefile, you can see that there are two types of programs in the DART-Ray package. Some programs have names starting with "create_adap_grid" and others start with "dartray_". The former are the program to create the input adaptive grids used in the calculation. The latter are the actual radiation transfer codes performing the calculations.

The filenames also indicate the kind of geometry used in the particular DART-Ray program. For example, "create_adap_grid_galaxy" and "dartray_galaxy" are the programs to create the grid for the analytical galaxy model and the corresponding RT code, respectively. The installation produces several compiled programs and most of the modules used are shared among them.

There are only slight differences in the programs, which can be found in the interface defining the input geometries for each model. If you learn how to modify these interfaces, you will be able to input any arbitrary 3D model in DART-Ray (see chapter 7).

Chapter 4

Preparing the input 3D grids for DART-Ray

4.1 The stellar emission and dust opacity 3D grids

The first step in an RT calculation with DART-Ray is the creation of a 3D adaptive grid, defined by the spatial coordinates and sizes of each grid element, along with the corresponding stellar volume emissivity j_ν and the extinction factor $k_\nu\rho$ (the dust density multiplied by the extinction coefficient; it is equivalent to the optical depth per unit length). In general, one needs to provide the values of j_ν and $k_\nu\rho$ for a set of UV/optical/NIR wavelengths for each grid element. However, if the grid values scale with wavelength, and are independent of the spatial position, then the user can just specify j_ν and $k_\nu\rho$ at a reference wavelength and factors needed to scale the grid. This is always true for $k_\nu\rho$, since the wavelength dependence is determined by the assumed dust model (which has to be the same throughout the RT model in the current version of DART-Ray). However, it is not necessarily true for j_ν, since the shape of the stellar emission distribution may depend on wavelength.

For the above reasons, DART-Ray accepts two kinds of input grid files:

- A "main grid" file, containing the 3D coordinates of the grid cell centres, as well as the corresponding j_ν and $k_\nu\rho$ values for a particular "reference wavelength". This file is always required;
- A set of "lambda grid" files, containing the values of j_ν and $k_\nu\rho$ at all grid positions and for the UV to NIR wavelengths used in the stellar emission RT calculation. These files are required only if the grid values at the different wavelengths are not scaled versions of each other.

To create these files, one needs to input to a grid creation program the information regarding the stellar luminosity and dust distribution, as well as the total model size (the total volume is always a cube of that size), two subdivision factors $[b1, b2]$ ($b1$ used for the subdivision of the entire model volume and $b2$ for each subsequent cell subdivision) and the threshold parameters needed to decide whether a cell is to be subdivided further or not.

The method used to create the main grid is as follows. For a chosen reference wavelength, the grid creation program subdivides the entire model volume into b1xb1xb1 cells of equal size and calculates the average j_ν and $k_\nu\rho$ within the volume of these cells. Then the program iteratively subdivides the new cells into b2xb2xb2 smaller cells, depending on whether or not the cells satisfy certain threshold criteria (e.g. sufficiently high total luminosity or optical depth). The program also ensures a smooth transition in the grid subdivision, from coarse to fine regions of spatial resolution. The subdivision iterations cease when no cells satisfy the subdivision criteria.

Before running one of the grid creation programs, the file "`input_grid_model.in`" must be prepared, along with the information needed to construct the grid for a particular model (e.g. see examples in the "*_GRIDS" directories). The `input_grid_model.in` file contains several blocks of input parameters, which are defined as fortran NAMELIST blocks. This allows for input in arbitrary order, as long as it appears within the correct NAMELIST block (identified by a “&namelist_block” within the input file). The specific input variables that are required for the grid creation can differ from model to model. However, there are some that are in common among almost all models:

- `label_model_lambda_grid` [character]: label contained within the names of the lambda grid files;
- `dir_grid` [character]: directory where the 3D grids are written;
- `grid_file` [character]: name of the HDF5 file containing the main grid;
- `grid_info_file` [character]: name of the text file containing some record of the model parameters used when creating the grid (only for archival purposes, not needed in the RT calculation);
- `units_lambda` [character]: units of the wavelengths in the wavelength input list. Microns (‘um’) is only allowed at the moment;
- `units_csize` [character]: units of the coordinates, distances and cell sizes. Only parsec is allowed;
- `file_lambda_list` [character]: file containing the list of wavelengths to be used in the grid creation. It is not necessary to use all wavelengths, however;
- `lambda_ref` [real]: Reference wavelength used in the construction of the main grid;
- `lambda_min` [real]: Minimum wavelength value for which the lambda grids have to be calculated;
- `lambda_max` [real]: Maximum wavelength value for which the lambda grids have to be calculated;
- `modelsize` [real]: linear size of the entire model in [pc];
- `base` [integer]: two element array containing the subdivision factors (b1, b2);
- `max_ncell` [integer]: maximum number of cells allowed in the grid. Note that if the maximum number of cells is reached during the grid creation, the program exits with an error without printing the grid. In this case, try again with a higher `max_ncell` or change the subdivision parameters such to obtain a grid with less cells;
- `max_lvl` [integer]: maximum number of allowed subdivisions. This parameter sets the highest spatial resolution reachable within the grid;
- `min_lvl` [integer]: minimum number of subdivisions. This parameter sets the minimum spatial resolution of the grid;
- `max_dtau` [real]: maximum optical depth allowed for a cell at the reference wavelength. If the cell optical depth is higher, the cell can be subdivided;
- `max_dlum` [real]: maximum luminosity allowed for a cell at the reference wavelength, usually defined in units of total model luminosity. If the cell luminosity is higher, the cell can be subdivided;
- `dust_model` [character]: dust model to be used in the calculation. This determines the value of k_λ used in the grid creation, as well as the grain size distribution and the grain absorption/scattering coefficients (needed in the RT calculation). Three choices are allowed at the moment: ‘DraineLi06’ (dust model of Draine and Li 2007), ‘TRUST’ (dust model used in the TRUST benchmark project), and ‘user’ (user-defined dust model. This requires several extra tables from the user, see `dust_opacity_tables`);

- `dust_opacity_tables` [character]: Q_λ opacity and g_λ tables to be used. If a standard dust_model is selected ('TRUST', 'DraineLi06'), there is no need to input this variable. It is required when `dust_model` = 'user'. There are three choices: 'DraineLi06' (same grain opacities as in the Draine and Li 2007 model); and 'TRUST' (same opacities as in the TRUST benchmark project. Note that ionized PAHs are not included in these tables); 'user' (user-provided tables). In this last case, you should specify the input table files using `file_q_gra` (graphite), `file_q_sil` (silicates), `file_q_pah_neu` (neutral PAHs) and `file_q_pah_ion` (ionized PAHs). In all cases where `dust_model`='user', you should also specify the corresponding size distribution files with `file_gra_fa` (graphite), `file_sil_fa` (silicates), `file_pah_neu_fa` (neutral PAHs) and `file_pah_ion_fa` (ionized PAHs). The grain size distributions can be restricted to only those dust species you want to include;

- `file_gra_fa` [character]: File containing input grain size distribution for Graphite grains (to be input only if dust_model = 'user');

- `file_sil_fa` [character]: File containing input grain size distribution for Silicate grains (to be input only if dust_model = 'user');

- `file_pah_neu_fa` [character]: File containing input grain size distribution for neutral PAH molecules (to be input only if dust_model = 'user');

- `file_pah_ion_fa` [character]: File containing input grain size distribution for ionized PAH molecules (to be input only if dust_model = 'user');

- `file_av_opacities` [character]: File containing the integrated/averaged opacity coefficients k_λ, $k_{\lambda,abs}$, $k_{\lambda,sca}$, $g_{\lambda,sca}$ (see section 1.6). It is read only if keyword input_av_opacities is set to TRUE. The input of this table is not a requirement, but it is a good way to check that your input size distributions and opacities are correctly input (especially if you are using a 'user' `dust_model` and `dust_opacity_tables`). DART-Ray calculates internally the same coefficients and checks that they are within a few percent compatible with those provided in `file_av_opacities`. See './DUST_OPACITY/TRUST/ZDA_BARE_GR_S_Effective.dat' for an example of this file;

- `input_av_opacities` [logical]: set to TRUE if integrated/averaged opacity coefficients k_λ, $k_{\lambda,abs}$, $k_{\lambda,sca}$, $g_{\lambda,sca}$ have to be read from input file `file_av_opacities`;

- `file_q_gra` [character]: File containing the Q_λ opacity and g_λ anisotropy coefficients for Graphite grains (to be input only if `dust_model` and `dust_opacity_tables` are 'user');

- `file_q_sil` [character]: File containing the Q_λ opacity and g_λ anisotropy coefficients for Silicate grains (to be input only if `dust_model` and `dust_opacity_tables` are 'user');

- `file_q_pah_neu` [character]: File containing the Q_λ opacity and g_λ anisotropy coefficients for neutral PAHs (to be input only if `dust_model` and `dust_opacity_tables` are 'user');

- `file_q_pah_ion` [character]: File containing the Q_λ opacity and g_λ anisotropy coefficients for ionized PAHs (to be input only if `dust_model` and `dust_opacity_tables` are 'user');

- `n_dust_size_qabs` [integer]: Array containing the number of grain sizes in the input Q_λ, g_λ tables. The values correspond to the following grain compositions (in order): Graphite, Silicates, neutral PAHs and ionized PAHs;

- `n_dust_wave_qabs` [integer]: Number of wavelengths in the input Q_λ, g_λ tables for the assumed dust_model and dust_opacity_tables. Required if dust_model = 'user' and dust_opacity_tables = 'user'. The values correspond to the following grain compositions (in order): Graphite, Silicates, neutral PAHs and ionized PAHs.

Other parameters that can be used in `input_grid_model.in` define the stellar emission and dust distribution. See the documentation for the different grid creation programs in section 4.2.

Once an input_grid_model file has been prepared for a particular model, the main grid can be created using the corresponding "`create_adap_grid_model`" programs. For example, to create the grids for a galaxy model defined by analytical functions, use the example input file by typing:

```
./create_adap_grid_galaxy ./GALAXY_GRIDS/input_grid_galaxy_example1.in.
```

The program reads `input_grid_galaxy_example1.in` and checks if the file for the main grid already exists. If not, it starts the algorithm to calculate the main grid according to the input parameters (note that the input tables for the stellar emission SEDs and the wavelength list have to placed in the same directory where the grids are printed.). As the program creates the grid, the coordinates of the newly created cells appear on the terminal window. Once the main grid creation has completed successfully, and the main grid file output correctly, one should relaunch the above command to start the calculation of the lambda grids for all the wavelengths between `lambda_min` and `lambda_max` within the input wavelength list. At the end of the grid creation, you will find the following files in the directory './GALAXY_GRIDS/EX1':

- `grid_model_ex1_main.h5`: the main grid file. Type "`h5ls grid_model_ex1_main.h5`" to see the list of arrays included in this file on the terminal. Among them, you will find: `base` (the subdivision factors), `ccoord` (the coordinates of the grid element centres), `csize` (the grid cell sizes), `dens` (the opacity per unit length $k_\nu \rho$ at the reference wavelength), `dens_stars` (the stellar luminosity at the reference wavelength within each grid element, that is, $4\pi j_\nu$), `lvl` (the subdivision level of each cell), and `cchild` (an array with the ID number of the first of the child cells of each cell. Equal to -1 if the cell is a leaf cell not further subdivided).

- `grid_model_ex1_all_l*LAMBDA*um.h5`: the lambda grids. By using the command `h5ls` as above, you can see the array list within these files. You will find only `dens` and `dens_stars`, which contain $k_\nu \rho$ and $4\pi j_\nu$ for the corresponding wavelength LAMBDA.

IMPORTANT NOTE: the creation of the main grid is usually more time consuming than the creation of the lambda grids. This is because during the creation of the main grid the program derives coordinates and other quantities used by the RT code, which determine the cell position within the grid. However, during the calculation of the lambda grids, only the values of $k_\nu \rho$ and $4\pi j_\nu$ are derived for the grid elements of the main grid already created. However, it is not always necessary to make a new main grid for a different RT model. If you are confident that the distribution of cells in a certain main grid can be used for a model with, for example, different geometrical parameters, you can simply make an `input_grid_model.in` file specifying the main grid file already existing and changing the required parameters. You might also want to change `label_model_lambda_grid` to differentiate different models. Once you launch the command line above, lambda grids will be calculated with the same cell coordinates and sizes, but obviously different values for j_ν and $k_\nu \rho$.

4.2 The standard grid creation programs

In the following subsections, we provide a brief explanation of the standard grid creation programs contained in the public version of DART-Ray, together with the lists of the possible input variables for creating the grids (the meaning of the parameters already defined in section 4.1 is not repeated here). Although these programs can be used to create many kinds of RT models, typically specific applications require some modifications (for example, the analytical formula defining the distribution of stars and dust). How to modify these programs is described in chapter 7.

4.2.1 `create_adap_grid_galaxy`

This program creates a 3D grid composed of up to three stellar emission components (an old stellar disk, a young stellar disk and a bulge) and two dust disks (a thick and a thin dust disk). For the stellar components, one can include all of them or just include one, using the variable `grid_type`. The dust disks are always included, but one can set their opacity using `tau1` and `tau2` (setting one or both of these values to zero is equivalent to removing the corresponding dust components). The stellar emission spectra can be provided using the input files `file_old_star_sed` (old disk and bulge) and `file_young_star_sed` (young disk). The spatially integrated luminosities of each stellar component will be equal to the "unit luminosities" listed in these files times the parameter `old` for the old disk, `SFR` for the young disc, and the product `old*bd_ratio` for the bulge. The user can choose different types of radial/vertical profiles for the disk components using the keywords `old_disk_type` (old disk), `young_disk_type` (young disk), `thick_disk_type` (thick dust disk), and `thin_disk_type` (thin dust disk).

In the following, we list all the possible input parameters (for each NAMELIST block) that can be used in the input file for this program:

&**galaxy_input_strings** ! All variables in this block are of type [character]

- `label_model_lambda_grid`
- `dir_grid`
- `grid_file`
- `grid_info_file`
- `file_lambda_list`
- `units_lambda`
- `grid_type`: Specifies which stellar emission components have to be included: 'all'= all components; 'disk' = only old stellar disk; 'tdisk' = only thin stellar disk; 'bulge' = only bulge component. Note that the chosen `grid_type` is added within the grid file names;
- `old_disk_type`: Old stellar disk profile type. Possible choices:

  - 'expR_expz' (pure double exponential profile):
  $\rho(R,z) = \rho_o exp(-R/hs - |z|/z_s)$

  - 'expR_sech2z' (exponential radial profile and $sech^2$ vertical profile):
  $\rho(R,z) = \rho_o exp(-R/hs) sech^2(|z|/z_s)$

  - 'flared_expz' (flared profile with exponential vertical profile. This profile is the one used to model the Milky Way in Popescu et al. (2017). It is composed of two regions: an inner region ($R < R1$) where the radial profile is determined by the linear term $[\frac{R}{R1} * (1-\chi) + \chi]$, which can be used to model a linear decrease or increase towards the centre, and an outer region in which the profile decreases exponentially with radius. The profile contains a variable vertical scale height $z_s(R)$. The term $\frac{z_s(0)}{z_s(R)}$ is such to guarantee that the vertical integration of the profile is always equal to the integration of an exponential profile with fixed scale height $z_s(0)$):
  $\gamma = log(\frac{z_s(R_{sun}) - z_s(0)}{z_s(R1) - z_s(0)})/log(\frac{R_{sun}}{R1})$
  $z_s(R) = z_s(0) + (z_s(R1) - z_s(0)) * (R/R1)^\gamma$
  $\rho(R,z) = \rho_o[\frac{R}{R1} * (1-\chi) + \chi] * \frac{z_s(0)}{z_s(R)} * exp(-\frac{R1}{hs}) * exp(-\frac{|z|}{z_s(R)}) \qquad \text{if R < R1}$
  $\rho(R,z) = \rho_o \frac{z_s(0)}{z_s(R)} * exp(-\frac{R}{hs}) * exp(-\frac{|z|}{z_s(R)}) \qquad \text{if R} \geq \text{R1}$

  - 'flared_sech2z':
  as 'flared_expz', but with $sech^2(\frac{|z|}{z_s(R)})$, instead of $exp(-\frac{|z|}{z_s(R)})$;

  - 'ellipt_expR_expz (elliptical exponential profile. This can be used to simulate an asymmetric structure such as a stellar bar)':
  $z_s(R)$ and γ defined as for 'flared_expz'
  $\rho(x,y,z) = \rho_o \frac{z_s(0)}{z_s(R)} * exp\left(-\sqrt{(\frac{x'}{hs})^2 + (\frac{y'}{hs2})^2}\right) * exp(-\frac{|z|}{z_s(R)}) \qquad \text{if R < R1}$
  $\rho(x,y,z) = 0 \qquad \text{if R} \geq \text{R1}$
  Note that the reference frame x'y' can be rotated with respect to the XY axis of the 3D grid.

- 'ellipt_expR_sech2z':
as 'ellipt_expR_expz', but with $sech^2(\frac{|z|}{z_s(R)})$, instead of $exp(-\frac{|z|}{z_s(R)})$.

- `young_disk_type`: Young stellar disk profile type. Same choices as for old_disk_type.
- `thick_disk_type`: Thick dust disk profile type. Same choices as for old_disk_type.
- `thin_disk_type`: Thin dust disk profile type. Same choices as for old_disk_type.
- `dust_model`
- `dust_opacity_tables`
- `file_gra_fa`
- `file_sil_fa`
- `file_pah_neu_fa`
- `file_pah_ion_fa`
- `file_av_opacities`
- `file_q_gra`
- `file_q_sil`
- `file_q_pah_neu`
- `file_q_pah_ion`
- `file_old_star_sed`: File containing the SED to be used by the old stellar component in units luminosities to be multiplied by the `old` parameter. Currently, DART-Ray accepts only SEDs in [W/Hz] or [erg/Hz].
- `file_young_star_sed`: File containing the SED to be used by the young stellar components in units luminosities to be multiplied by the `sfr` parameter. Currently, DART-Ray accepts only SEDs in [W/Hz] or [erg/Hz].
- `subdivision_criteria`: Cell subdivision criteria to be used. Choices are:
- 'standard': cell subdivision IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau` OR stellar disk luminosity > `max_dlum`*total luminosity OR subdivision level < `min_lvl` OR ($|z|$ < `z_subd_lim` AND R < `R_subd_lim`))
- 'milky way' : subdivision IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau` OR stellar disk luminosity > `max_dlum`*total luminosity OR subdivision level < `min_lvl` OR ($|z|$ < `z_subd_lim` AND R < `R_subd_lim`) OR (cell subtended solid angle from position (`rsun`, 0,0) > `omega_max` AND $|z|$-half cell size < `z_subd_lim2`))

&**galaxy_input_var**

- `lambda_ref` [real]
- `lambda_min` [real]
- `lambda_max` [real]
- `rtrun` [real]: Radial truncation radius for all disks [pc];
- `rsun` [real]: Third radial distance used in the "flared" profile [pc]. Called rsun because it has been used to set the sun position in Milky Way model of Popescu et al. (2017);
- `max_z` [real]: Maximum $|z|$ allowed for non zero stellar emissivity and dust density [pc];
- `max_rad` [real]: Maximum R allowed for non zero stellar emissivity and dust density [pc];

- sha [real]: Factor determining the sharpness of the profiles close to the truncation radius rtrun. The profile goes to zero within a radial interval equal to sha in units of rtrun;
- omega_max [real]: Maximum subtended solid angle of a cell seen from the position (rsun(), 0,0). To use this variable you have to set subdivision_criteria = 'milky_way';
- modelsize [real]
- base [integer]
- max_ncell [integer]
- max_lvl [integer]
- min_lvl [integer]
- max_dtau [real]
- max_dlum [real]
- n_dust_size_qabs [integer]
- n_dust_wave_qabs [integer]
- z_subd_lim [real]: Z coordinate below which cells have to be subdivided if their R is less than R_subd_lim [pc].
- R_subd_lim [real]: R coordinate below which cells have to be subdivided if their z is less than z_subd_lim [pc].
- z_subd_lim2 [real]: Z coordinate used in the 'milky_way' subdivision criteria to limit the subdivisions of cells with small subtended solid angle from the position (rsun, 0,0) [pc]. See subdivision_criteria.

&**galaxy_input_var_old_disk**

- old [real]: Scaling factor applied to the input old stellar luminosity spectra;
- hs_disk_b [real]: B-band (4430 A) old stellar disk scale length [pc];
- zs_disk [real]: Scale height for the old stellar disk (basic value) [pc];
- zs_disk_r1 [real]: Scale height for the old stellar disk at the inner radius hsin [pc];
- zs_disk_rsun [real]: Scale height for the old stellar disk at the radius rsun [pc];
- chi_disk [real]: Parameter used in the "flared" profile to specify the steepness of the radial profile within the inner radius hsin;
- hsin [real]: Inner radius of the old stellar disk (R1) [pc];
- hs_disk_arr [real]: Scale length values for the old stellar disk at each wavelength of the input wavelength grid [pc]. In the case of the elliptical exponential profile, this is the X-axis scale length. By default, all values are set equal to hs_disk_b. If different values are required, hs_disk_arr can be specified in the input. The wavelength indices of these input values have to be specified in id_hs_disk_arr;
- hs_disk2_arr [real]: Same as hs_disk_arr but for the second scale length required in the elliptical exponential disk profile type [pc];
- id_hs_disk_arr [integer]: Wavelength grid indices of the scale length values specified in the input hs_disk_arr and hs_disk2_arr. The first index, corresponding to the first wavelength in file_lambda_list, is 0 (not 1!);
- theta_disk_ellipt [real]: Rotation angle in [deg] around the Z-axis for the elliptical exponential disk profile.

&**galaxy_input_var_young_disk**

- `sfr` [real]: Scaling factor applied to the young stellar disk luminosity spectra;
- `hs_tdisk` [real]: Scale length of the young stellar disk or, in the case of the ellipsoidal exponential profile, X-axis scale length [pc];
- `hs_tdisk2` [real]: Y-axis young stellar disk scale length for the ellipsoidal exponential profile [pc];
- `zs_tdisk` [real]: Scale height of the young stellar disk (basic value) [pc];
- `zs_tdisk_r1` [real]: Scale height of the young stellar disk at the inner radius `hs1in` (R1) [pc];
- `zs_tdisk_rsun` [real]: Scale height of the young stellar disk at the radius `rsun` [pc];
- `chi_tdisk` [real]: Parameter used in the "flared" profile to specify the steepness of the radial profile within the inner radius `hs1in`;
- `hs1in` [real]: Inner radius of the young stellar disk (R1) [pc];
- `theta_tdisk_ellipt` [real]: Rotation angle in [deg] around the Z-axis for the elliptical exponential thin disk profile.

&**galaxy_input_var_bulge**

- `reff` [real]: Bulge effective radius [pc];
- `acap_bulge` [real]: Inner truncation radius for the bulge profile [pc];
- `ellipt` [real]: Bulge ellipticity. The normalized radial coordinate m for the bulge volume emissivity is defined as $m = \sqrt{R^2 + (z/ellipt)^2/R_{eff}}$;
- `mtrunc` [real]: Outern truncation radius for the bulge in units of effective radii;
- `bd_ratio` [real]: Bulge-to-Disk luminosity ratio. Used to define luminosity scaling for the bulge, that is, bd_ratio*old*lnu_old;
- `nsersic` [integer]: Bulge sersic index (see inside subroutine `av_star_bulge` for the volume emissivity formula used in each case. From R. J. Tuffs, private communication);
- `theta_bulge`: Rotation angle around the Z-axis for the bulge profile;
- `ellipt_xy`: Bulge XY ellipticity. The radial coordinate R for the bulge volume emissivity is defined as $R = \sqrt{x^2 + (y/ellipt\_xy)^2}$.

&**galaxy_input_var_thick_dust_disk**

- `tau1` [real]: B-band (4430 A) central face-on optical depth of the thick dust disk in the case of the pure exponential profile. In the case of the flared profiles, this is the optical depth one would have if the exponential profile for $R > R1$ continues until $R = 0$;
- `hd_disk` [real]: Scale length of the thick dust disk [pc];
- `zd_disk` [real]: Scale height for the thick dust disk (basic value) [pc];
- `zd_disk_r1` [real]: Scale height for the thick dust disk at the inner radius `hdin` (R1) [pc];
- `zd_disk_rsun` [real]: Scale height for the thick dust disk at the radius `rsun` [pc];
- `chi_dust_disk` [real]: Parameter used in the "flared" profile to specify the steepness of the radial profile within the inner radius `hdin`;
- `hdin` [real]: Inner radius for the thick dust disk (R1) [pc];

- theta_dust_disk_ellipt [real]: Rotation angle in [deg] around the Z-axis for the elliptical exponential profile for the thick dust disk.

&**galaxy_input_var_thin_dust_disk**

- tau2 [real]: B-band (4430 A) central face-on optical depth of the thin dust disk in the case of the pure exponential profile. In the case of the flared profiles, this is the optical depth one would have if the exponential profile for $R > R1$ continues until $R = 0$;
- hd_tdisk [real]: Scale length of the thin dust disk [pc];
- zd_tdisk [real]: Scale height for the thin dust disk (basic value) [pc];
- zd_tdisk_r1 [real]: Scale height for the thin dust disk at the inner radius hd1in (R1) [pc];
- zd_tdisk_rsun [real]: Scale height for the thin dust disk at the radius rsun [pc];
- chi_dust_tdisk [real]: Parameter used in the "flared" profile to specify the steepness of the radial profile within the inner radius hd1in;
- hd1in [real]: Inner radius for the thin dust disk (R1) [pc];
- theta_dust_tdisk_ellipt [real]: Rotation angle in [deg] around the Z-axis for the elliptical exponential profile for the thin dust disk.

&**galaxy_input_logical**

- input_av_opacities [logical]

4.2.2 create_adap_grid_Nbody

This grid creation program prepares a grid based on the distribution of stellar and gas particles in an N-body/SPH simulation. The simulation file has to be provided in H5DF format (see description file_nbody_sph for details). The stellar library used to convert stellar mass into stellar luminosity can be chosen using stellar_library. The gas mass is converted into dust mass proportionally to the gas metallicity (the gas-to-dust ratio of the dust models corresponds to the solar metallicity. For other values, the gas-to-dust ratio is scaled accordingly). A maximum temperature for the gas containing dust can be chosen using gastemp_limit.

The following is the complete list of possible input parameters:

&**Nbody_sph_input_strings** ! All variables in this block are of type [character]

- label_model_lambda_grid
- dir_grid
- grid_file
- grid_info_file
- file_lambda_list
- units_lambda
- dust_model
- dust_opacity_tables
- file_gra_fa
- file_sil_fa
- file_pah_neu_fa
- file_pah_ion_fa

- `file_av_opacities`
- `file_q_gra`
- `file_q_sil`
- `file_q_pah_neu`
- `file_q_pah_ion`
- `file_nbody_sph`: File name of the imported Nbody-SPH simulation. This file contains the following arrays:
  - 'agestar': the age of the stellar particles in Gyr;
  - 'fehgas': [Fe/H] ratio of the gas particles;
  - 'fehstar': [Fe/H] ratio of the stellar particles;
  - 'gascoord': 3D coordinates of the gas particles;
  - 'gastemp': Gas particle temperature [K];
  - 'mgas': Gas particle mass in M(sun);
  - 'mstar': Star particle mass in M(sun);
  - 'ofegas': [O/Fe] ratio for the gas particles;
  - 'starcoord': 3D coordinates of the stellar particles.
  This file can be created from a tipsy file using the python script tipsy2dartray.py. Note that the gas metallicity is inferred from the O abundance, the stellar metallicity from the Fe abundance.
- `stellar_library`: Name of the stellar library that has to be used to convert mass into luminosity. Choices are: 'starburst99' (standard starburst99 Single age stellar population SED, Kroupa IMF), 'maraston05_kr_rhb' (Maraston et al.2005 SSP SEDs, Kroupa IMF, red horizontal branch), and 'user' (user provided table, see `file_stellar_library`);
- `file_stellar_library`: Name of the HDF5 file containing the stellar library specified in stellar_library. This is an input parameter only if stellar_library = 'user'. This file contains three arrays: lambda_lib_arr (wavelength list in [um]), met_arr (metallicity), and lum_to_mass_arr (stellar luminosity-to-mass ratio in erg/s/Hz/M(sun)). See e.g. STELLAR_LIBRARIES/starburst99/table_lum_mass_vs_age_met_starburst99.h5.
- `subdivision_criteria` Subdivision criteria to be used. Choices are:
  - 'standard': subdivision IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau`*optical depth averaged over entire model OR stellar luminosity > `max_dlum`*total model luminosity OR subdivision level < `min_lvl`)

&**Nbody_sph_input_var**

- `modelsize` [real]
- `lambda_ref` [real]
- `lambda_min` [real]
- `lambda_max` [real]
- `base` [integer]
- `max_ncell` [integer]
- `max_lvl` [integer]
- `min_lvl` [integer]
- `max_dtau` [integer]
- `max_dlum` [integer]
- `n_dust_size_qabs` [integer]

- `n_dust_wave_qabs` [integer]
- `z_sun` [real]: Solar metallicity value. The default value is 0.018, but it can be modified in the input if necessary. Value checking is done within the routines that use this parameter.
- `gastemp_limit` [real]: Maximum gas temperature [K] for gas containing dust particles. Hotter gas particles are considered dust-free.

&**Nbody_sph_input_logical**

- `input_av_opacities` [logical]

4.2.3 `create_adap_grid_trustI`

It prepares the grid for a slab of dust illuminated by a star. This is the geometry used in the first benchmark of the TRUST benchmark project. The dimensions of the slab and the star position are hard coded in user_routines_trustI.f90. The vertical optical depth can be changed using `tau_z`. This model does not require lambda grids, because both the stellar luminosity and dust opacity distributions at each wavelength differ only by global scaling factors.

List of possible input parameters:

&**trustI_input_strings** ! All variables in this block are of type [character]

- `dir_grid`
- `grid_file`
- `grid_info_file`
- `file_lambda_list`
- `units_lambda`
- `dust_model`
- `dust_opacity_tables`
- `file_gra_fa`
- `file_sil_fa`
- `file_pah_neu_fa`
- `file_pah_ion_fa`
- `file_av_opacities`
- `file_q_gra`
- `file_q_sil`
- `file_q_pah_neu`
- `file_q_pah_ion`
- `subdivision_criteria`: Subdivision criteria to be used. Choices are:
  - 'standard': subdivision IF subdivision level < `max_lvl` AND (cell upper border coinciding with slab upper border OR (cell optical depth > `max_dtau` AND subdivision level <`min_lvl_in`) OR subdivision level < `min_lvl`)

&**trustI_input_var**

- `modelsize` [real]
- `tau_z` [real] : Vertical optical depth of the slab at the reference wavelength.

- lambda_ref [real]
- base [integer]
- max_ncell [integer]
- max_lvl [integer]
- min_lvl [integer]
- max_dtau [integer]
- max_dlum [integer]
- n_dust_size_qabs [integer]
- n_dust_wave_qabs [integer]
- min_lvl_in [integer] : Minimum subdivision level within the slab for the "standard" subdivision_criteria.

&**trustI_input_logical**

- input_av_opacities [logical]

4.2.4 create_adap_grid_magtar

This program has been developed for the modelling of dust ring aroung SGR+1900 (see Natale et al. 2017). It can be used for creating the dust configurations used in that paper by selecting the appropriate dust_geometry.

List of possible input parameters:

&**magtar_input_strings** ! All variables in this block are of type [character]

- dir_grid
- grid_file
- grid_info_file
- file_lambda_list
- units_lambda
- dust_model
- dust_opacity_tables
- file_gra_fa
- file_sil_fa
- file_pah_neu_fa
- file_pah_ion_fa
- file_av_opacities
- file_q_gra
- file_q_sil
- file_q_pah_neu
- file_q_pah_ion

- `dust_geometry`:Type of dust geometries. Choices:
  - 'shell':
  $R^2 = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}$
  $\rho = \rho_o$ for $\sqrt{|R^2 - 1|} < \Delta R$
  $\rho = 0$ otherwise
  - 'cavity':
  $\rho = \rho_o$ for $R > 1$
  $\rho = 0$ otherwise
  - 'wind':
  $\rho_d(R) = \rho_d(R_d)\left(\frac{R}{R_d}\right)^2$ for $R < R_d$,
  $\rho_d(R) = \rho_d(R_d)\left(\frac{R}{R_d}\right)^{-2}$ for $R > R_d$
  with $R_d = 1$.

- `subdivision_criteria`: Subdivision criteria to be used. Choices are:
  -'shell': subdivide IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau` OR subdivision level < `min_lvl`)
  - 'cavity': subdivide IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau` AND R < `R_subd_lim`)
  - 'wind' : same as for 'cavity'

&**magtar_input_var**

- `modelsize` [real]

- `tau_z` [real] : Optical depth per pc at the reference wavelength for the regions containing dust in the 'shell' and 'cavity' profiles. For the 'wind' profile, this is the optical depth per pc at $R = 1$.

- `lambda_ref` [real]

- `base` [integer]

- `ax` [real] : Ellipsoidal semi-axis along the X-axis [pc];

- `by` [real] : Ellipsoidal semi-axis along the Y-axis [pc];

- `cz` [real] : Ellipsoidal semi-axis along the Z-axis [pc].

- `elrad_width` [real] : Semi-width of the dust shell in normalized radial units (ΔR);

- `max_ncell` [integer]

- `max_lvl` [integer]

- `min_lvl` [integer]

- `max_dtau` [integer]

- `max_dlum` [integer]

- `n_dust_size_qabs` [integer]

- `n_dust_wave_qabs` [integer]

- `R_subd_lim` [real]: normalized R coordinate below which cells have to be subdivided if their optical depth is higher than `max_dtau`.

&**magtar_input_logical**

- `input_av_opacities` [logical]

4.2.5 create_adap_grid_2dto3d

This program can be used to derive a 3D grid from an axisymmetric distribution of j_ν and $k_\nu\rho$ defined on an (R,z) grid. This is useful if the axisymmetric distribution cannot be defined analytically (for example if it has been derived using a numerical 2D code). To use this program, one needs separate input tables for j_ν and $k_\nu\rho$ for each wavelength. These tables have to be stored into the directory `dir_grid_2d` and their file names have to follow the form 'grid_'+`label_model_2d`+'_l'+wavelength+'um.dat'. The first line of the 2D tables has to be a comment. All following lines have to contain in order: R, z, j_ν and $k_\nu\rho$. See example files within the directory `./2DTO3D/EX1`.

&**m2dto3d_input_strings** ! All variables in this block are of type [character]

- `label_model_lambda_grid`
- `dir_grid`
- `grid_file`
- `grid_info_file`
- `file_lambda_list`
- `units_lambda`
- `dust_model`
- `dust_opacity_tables`
- `file_gra_fa`
- `file_sil_fa`
- `file_pah_neu_fa`
- `file_pah_ion_fa`
- `file_av_opacities`
- `file_q_gra`
- `file_q_sil`
- `file_q_pah_neu`
- `file_q_pah_ion`
- `subdivision_criteria`: Cell subdivision criteria to be used. Choices are:
  - 'standard': cell subdivision IF subdivision level < `max_lvl` AND (cell optical depth > `max_dtau` OR stellar disk luminosity > `max_dlum`*total luminosity OR subdivision level < `min_lvl` OR ($|z|$ < `z_subd_lim` AND R < `R_subd_lim`));
- `dir_grid_2d`: Directory where the 2D grids of emissivity and dust extinction coefficients are stored;
- `label_model_2d`: label contained in the 2D grid file names.

&**m2dto3d_input_var**

- `lambda_ref` [real]
- `lambda_min` [real]
- `lambda_max` [real]
- `modelsize` [real]

- `base` [integer]
- `max_ncell` [integer]
- `max_lvl` [integer]
- `min_lvl` [integer]
- `max_dtau` [real]
- `max_dlum` [real]
- `n_dust_size_qabs` [integer]
- `n_dust_wave_qabs` [integer]
- `z_subd_lim` [real]: Z coordinate below which cells have to be subdivided if their R is less than `R_subd_lim` [pc].
- `R_subd_lim` [real]: R coordinate below which cells have to be subdivided if their z is less than `z_subd_lim` [pc].

&**m2dto3d_input_logical**

- `input_av_opacities` [logical]

4.3 Adding point sources

There is no need to specify the position and luminosity of point sources when creating the grids. As shown in the next chapter, point sources at arbitrary positions within the 3D grid can be specified directly in the input file for the RT calculation.

4.4 Choosing the dust model

There are two default dust models in DART-Ray: the dust model of Draine and Li 2007 and the TRUST benchmark dust model (based on the model BARE-GR-S of Zubko, Dwek, and Arendt 2004), both representing the average dust properties of the local Milky Way. These two dust models can be chosen by simply using the input parameter "`dust_model`" and assigning the values 'TRUST' or 'DraineLi06'. In this way, the grain size distributions and the opacity coefficients for the single grains are set to the values in those works. It is also possible to input a user defined dust model by inputting `dust_model = 'user'`. In this case, the size distribution and the opacity coefficient files can be specified by the user.

The following paragraphs contain a detailed description of the input of a 'user' dust model. These can be skipped if you are only using the standard `dust_model` = **'TRUST' or 'DraineLi06'**.

If the 'user' dust model is selected, then the user has to provide at least the size distribution files. The names of the grain size distribution files have to be input through the input variables `file_gra_fa` (graphite), `file_sil_fa` (silicates), `file_pah_neu_fa` (neutral PAH), and `file_pah_ion_fa` (ionized PAH). An example of how these files have to be formatted can be found in the file
'./DUST_OPACITY/TRUST/ZDA_BARE_GR_S_SzDist_Gra.dat'. The format should be stricly followed with same number of comment lines at the top of the file and with the same units. The size distribution files only have to be provided for the dust species included. At least one of these files have to be input.

As well as the size distributions, one has to specify which "Q_λ" opacity coefficients and "g_λ" anisotropy parameters are to be used (see section 1.6). To use the same Q_λ and g_λ parameters as the 'TRUST' or 'DraineLi06' models, the input parameter `dust_opacity_tables` should be defined accordingly (e.g. `dust_opacity_tables` = 'TRUST'). If one wants to use other Q_λ and g_λ coefficient tables, then select `dust_opacity_table` = 'user' and input the name of the opacity table files using `file_q_gra` (Graphite), `file_q_sil` (Silicates), `file_q_pah_neu` (neutral PAH) and `file_q_pah_ion` (ionized PAH). The only files that need to be included are for the dust species for which the size distributions are provided. Note that the formatting of the Q_λ and g_λ tables have to be identical to that used for the standard dust models 'TRUST' and 'DraineLi06' (see e.g. DUST_OPACITY/TRUST/Gra_121_1201.dat). However, one can use a different number of grain sizes and a different number of wavelengths for which the coefficients are tabulated in the user provided files. One can specify the number of grain sizes considered for each grain size composition using the parameter `n_dust_size_qabs` and the number of wavelengths using `n_dust_wave_qabs`.

A user providing their own size distributions and Q_λ, g_λ tables could easily provide a table in the wrong format, or parameters with the incorrect units. For this reason it is worth providing a file with the integrated/average opacity coefficients k_λ, $k_{\lambda,abs}$, $k_{\lambda,sca}$, $g_{\lambda,sca}$ (see section 1.6) calculated independently. To do this, one has to set `input_av_opacities` = .TRUE. and provide the name of the table file using `file_av_opacities` (see './DUST OPACITY/TRUST/ZDA BARE GR S Effective.dat' for an example of this file). DART-Ray recalculates the integrated/average opacity coefficients from the size distributions and the opacity tables, and checks that they are consistent to within 5% with the values provided in `file_av_opacities`. Note that the values for k_λ, $k_{\lambda,abs}$, $k_{\lambda,sca}$, $g_{\lambda,sca}$ actually used in the RT calculations are those provided by the user in `file_av_opacities` if this file is input.

4.5 Modifying the cell subdivision criteria

The grid subdivision criteria cannot be input using an input file, as they are hard-coded in each of the grid creation programs `create_adap_grid_model`. There are some already implemented subdivision criteria that can be selected using `subdivision_criteria` (see above documentation for each grid creation program). All these criteria are implemented in the logical function `subdivision` within the 'create_adap_grid_model.f90' files. This function takes as input the cell centre coordinates and size and it returns TRUE if the corresponding cell has to be further subdivided. In case one wants to modify or add different subdivision criteria, one just has to add another case to the `select case` list inside this function and use the newly defined `subdivision_criteria` in the input file to create the grid. Note that the program has to be recompiled to make sure that the applied changes are effective.

Chapter 5

The DART-Ray RT programs

5.1 Starting an RT calculation

Once the 3D grids have been created, one has to launch the RT calculation using the appropriate "`dartray_model`" program (the programs usually differ only in some subroutines within the '`user_routines_model`' module). But before that, one has to set the environmental variable `OMP_NUM_THREADS`, which gives the number of threads used in the OpenMP parallelized regions (essentially the number of CPUs used per computer cluster node). This has to be done using the shell commands `setenv` or `export`, depending on whether the shell environment you are using is csh or bash, that is:

```
setenv OMP_NUM_THREADS 8
```

to set the number of threads to 8 for csh shells or:

```
export OMP_NUM_THREADS=8
```

for bash shells. In general, it is recommended to set the OpenMP thread number to the maximum number of CPUs available on the used machine/cluster node. Then, to launch a DART-Ray run, one has to use an appropriate MPI command. For example, for several MPI implementations, use

```
mpirun -n 2 ./dartray_galaxy ./GALAXY_GRIDS/input_galaxy.in,
```

which runs dartray with 2 MPI processes, each using the number of CPUs specified in OMP_NUM_THREADS. The dartray program launched in the example above is that for the galaxy model reading the input file `input_galaxy.in` (the content of this input file will be explained later in this chapter). Typically, one uses MPI with multiple processes only when running DART-Ray on a computer cluster. Furthermore, in that case, most likely you have to prepare a batch scheduling script (e.g. a PBS script), containing the command line above and specifying the number of OpenMP threads and MPI processes you want to reserve for the job (see documentation for the computer cluster you are using to prepare this script). Note that DART-Ray accepts any combination of MPI processes and OpenMP threads, but it requires the same number of OpenMP threads for all cluster nodes.

5.2 The output files

After an RT run, depending on the chosen output options, there can be several output files containing the labels listed below within their names. The additional labels '_part2' and '_dir' are included in the file names of the output for the direct light processing phase (not containing the contributions to the output quantity due to scattered light). The additional label '_dust' is present for files containing the quantities calculated during the dust emission RT. All output files are in HDF5 format, unless specified otherwise. The following list shows all the possible output filenames and the arrays contained within them. For each array, the meaning of their subscripts is specified within square brackets[1]. Note that there is an output file for each wavelength for the RFED files, the scattering source function files and the files for the radiation specific intensity as received by the observers.

Output files:

- '**ufield_part1**': `u_fest[cell ID]`, lower limit to the radiation field energy density distribution as derived during the pre-calculation phase. Note that the RFED is in units of 'J/Hz/pc^3' or 'erg/Hz/pc^3' for the stellar emission RT output (depending on whether the units of the stellar luminosity in the 3D grid are in 'W/Hz' or 'erg/s/Hz'). The RFED is always in 'W/Å/pc^3' for the dust emission RT output;
- '**ufield**': `u_final[cell ID]`, radiation field energy density distribution as derived after an RT run. Same units as above;
- '**scaspe_tot**': `scaspe_tot[angular direction ID, cell ID]`, scattering source function as derived at the end of the RT run. The values in this array correspond to $j_{\lambda,\mathrm{sca}}\Omega_{\mathrm{HP,sca}}V_c$, with $\Omega_{\mathrm{HP,sca}}$ the solid angle associated with the HealPix spherical pixels used to discretize the scattering source function, and V_c the cell volume. The units are the same as those used for the stellar luminosity for the stellar emission RT. For the dust emission RT, the units are always 'W/Å'. Note that the files for each wavelength have the same 'cell' dimension, but they might have different 'angular direction' dimension. Also, the angular directions for which the scattering source function is stored contain the HealPix directions first, for the angular resolution chosen by the code (see section 5.3.11), and then the external observer directions input in the RT calculation;
- '**scaspe_part2**': `scaspe[angular direction ID, cell ID]`, scattering source function distribution as derived at the end of the direct light processing. Same comments as above for 'scaspe_tot';
- '**sed**': `sed_arr[wavelength ID, external observer ID]`, spectral energy distributions in Jy for the total radiation (direct + scattered) reaching all the external observer positions. Note that the values depend on the assumed observer distance;
- '**sed_dir**': `sed_arr_dir[wavelength ID, external observer ID]`, spectral energy distributions in Jy for the direct light radiation reaching all the external observer positions. Same comment as above for 'sed';
- '**i_obs**': `i_obs[radiation source ID, external observer ID]`, radiation specific intensity reaching the external observers for the radiation produced by each grid element and each point source. The units are 'W/Hz/pc^2/sr' or 'erg/s/Hz/pc^2/sr' for the stellar emission RT, depending on whether the stellar emission luminosity is in 'W/Hz/pc^3' or 'erg/s/Hz/pc^3' respectively. For the dust emission RT, the units are always 'W/Å/pc^2/sr;
- '**i_obs_in**': `i_obs_in[radiation source ID, internal observer ID]`, radiation specific intensity reaching the internal observers for the radiation produced by each grid element and each point source. Same comments as above for 'i_obs';
- '**maps**': `map_arr_out[X pixel ID, Y pixel ID, wavelength ID, external observer ID]`, surface brightness maps for the external observers in MJy/sr; `lambda_arr_maps[wavelength ID]`, wavelength list for the output surface brightness maps in μm; `size_maps`, physical size of the map sides in pc;
- '**maps_in**': `map_in_arr_out[spherical pixel ID, wavelength ID, internal observed ID]`, surface brightness maps for the internal observers in MJy/sr. Note that the maps are in HealPix NESTED pixelation scheme with Cartesian axis oriented as in the RT model;

[1] When using the command `h5ls` to show an HDF5 file content, note that the order of the subscripts is reversed compared to that of the arrays in FORTRAN 90

lambda_arr_maps[wavelength ID], wavelength list for the output surface brightness maps in μm;

- '**lum_lost**': lum_lost[wavelength ID], file containing the luminosity neglected during the RT calculation because of the ray blocking criteria. Same units as the stellar luminosity for the stellar emission RT. Always in W/m for the dust emission RT;
- '**psel_av**': psel_av_arr[dust heating iteration ID, scattering iteration ID, radiation source ID], average ray-path length in pc for the rays originated by each grid cell and each point source and at each scattering/dust heating iteration;
- '**info**': Text file containing a log of the input parameters used in the RT calculation.

5.3 Choosing the input parameters

All the DART-Ray programs accept the same list of input parameters within the input_model.in file. As for the input_grid_model.in file, used to create the 3D grids, the input parameters are separated in NAMELIST blocks (some input parameters are the same as those used in input_grid_model.in). In this section, we list the possible input parameters within each NAMELIST block in alphabetical order. In the following sections, we provide explanations for groups of variables depending on the common part of the calculation that they affect. In the next chapter, some practical examples are shown.

5.3.1 List of all input parameters

&**dartray_input_strings** ! All the parameters in this block are of CHARACTER type

- dir_grid: directory where the 3D grids are located;
- dir_runs: Directory for the output files. If it does not exist, it is created automatically;
- dust_heating_type: Type of dust heating. Choices are: 'eff' (effective single grain. Equilibrium dust temperature/emission calculated for a single grain with average opacity parameters), 'equ' (Equilibrium dust temperature/emission calculated for each grain specified by the input grain size distributions), 'sto' (Full non-equilibrium dust temperature/emission calculation for all grains specified by the input grain size distributions), and 'sto_lib' (as 'sto', but using the adaptive SED library approach of Natale et al. 2015).
- dust_model: see section 4.1;
- dust_opacity_tables: see section 4.1;
- file_av_opacities: see section 4.1;
- file_calorimetry_Gra: File containing the grain density, specific enthalpy and specific heat capacity for Graphite and PAH grains as a function of grain temperature. This has to be specified if dust_opacity_table = 'user' AND stochastic heating dust emission has to be calculated. See e.g. ./DUST_OPACITY/TRUST/Graphitic_Calorimetry_1000.dat for the specific format of the tables to be used. The number of temperatures can be input using n_dust_temp_cal;
- file_calorimetry_Sil: File containing the grain density, specific enthalpy and specific heat capacity for Silicates grains as a function of grain temperature. Same comments as for file_calorimetry_Gra;
- file_dir_out: File containing the angular directions θ and ϕ specifying the lines of sight of the far-away observers. The first line has to be a comment. The following lines have to be the list of θs (first column) and ϕ (second column) in radiants. Note that θ is defined as the angle between the line of sight and the Z axis of the 3D grid, ϕ is the line-of-sight rotation angle measured on the XY plane from the X axis;

- `file_gra_fa`: see section 4.1;
- `file_lambda_list`: file containing the list of wavelengths in μm to be used in the RT calculation. It is not necessary to use all wavelengths;
- `file_pah_neu_fa`: see section 4.1;
- `file_pah_ion_fa`: see section 4.1;
- `file_param_src`: File containing the properties used to define the spectra of the point sources. Its format depends on the specific RT model since it is read in the "user_routines_model" modules;
- `file_pos_obs`: File containing the 3D coordinates of observers inside the RT model. The first line has to be a comment. The following lines contain the list of X-coordinate (first column), Y-coordinate (second column), Z-coordinate (third column) for the position of each internal observer;
- `file_p_src`: File containing the 3D coordinates for the point sources within the RT model The first line has to be a comment. The following lines contain the X, Y,Z coordinates for the position of each point source;
- `file_q_gra`: see section 4.1;
- `file_q_pah_neu`: see section 4.1;
- `file_q_pah_ion`: see section 4.1;
- `file_q_sil`: see section 4.1;
- `file_sil_fa`: see section 4.1;
- `label_model_lambda_grid`: label contained within the names of the lambda grid files;
- `label_model_out`: Label contained in all the output file names;
- `param_to_project`: Quantity to be projected on observer maps in the 'projection' rt_algorithm. Choices are: 'stellar_emission' and 'optical_depth';
- `rt_algorithm`: Type of RT algorithm to be used. Choices are:
  - 'main': this is the standard RT algorithm. Firstly, it proceeds with the calculation of the radiation field energy density, scattering source function, outgoing radiation specific intensity and surface brightness maps for the stellar emission. Then it calculates the same quantities for the dust emission (unless no_dust_rt = .TRUE.);
  - '2D': same as the 'main' RT algorithm, but for axisymmetric models. In this case, the code checks that the input grid is indeed axisymmetric and then performs the RT calculations taking into account the symmetries. This mode performs RT calculations up to a factor 8 times faster than the main mode. However, it is not possible to calculate surface brightness maps using this mode. If this is required, one can output the scattering source function by selecting `print_scaspe_tot` = .TRUE. and then calculate maps using the 'i_obs' algorithm;
  - 'dust' and 'dust_2D': these RT algorithms perform only the dust emission RT calculations. To use these modes, the stellar emission RT has to have already been performed and the output radiation fields saved on disk. The two modes are for the normal and 2D algorithm, respectively;
  - 'sed' and 'sed_dust': these modes can be used to calculate the integrated SEDs and the maps using the outgoing specific intensity 'i_obs' files saved in a previous RT calculation. The two modes are for the stellar emission and the dust emission respectively;
  - 'i_obs' and 'i_obs_dust': these modes can be used to calculate the specific intensity reaching the observers and the corresponding surface brightness maps and SEDs for arbitrary observer positions without repeating a complete RT calculation. These modes can only be used if the scattering source function is to be printed on disk (see `print_scaspe_tot`).

The two modes are for the stellar emission and dust emission respectively;
- 'projection': this mode can be used to make surface brightness maps of the stellar emission without any dust or maps of the dust optical depth. Useful to check that the input grid has been correctly derived before starting a complete RT calculation.

- `units_csize`: Units of grid cell sizes. The only possible choice is 'pc'. A reminder!;
- `units_lambda`: Units for the input wavelength array. The only choice is 'um' (microns). Another reminder!
- `units_luminosity`: Units of input stellar emission luminosity. Choices are 'erg/s/Hz', 'W/Hz';

&**dartray_input_var**

- `accuracy` [real]: Accuracy required for the RFED at each position. This parameter determines the value of f_U (see section 2.2) through the formula f_U = `accuracy`/(number of radiation sources x 0.25). Allowed range = [0,1];
- `bm_par` [real]: Minimum number of ray-intersections per cell during the pre-iteration and direct-light processing for the stellar emission RT. Allowed range = [0,1000];
- `bm_par_max` [real]: Maximum number of ray-intersections per cell. Allowed range = [100* max([`bm_par`, `bm_par_sca`]), infinity]'
- `bm_par_sca` [real]: Minimum number of ray-intersections per cell during the scattered light processing for the stellar emission RT and all phases for the dust emission RT. Allowed range = [0,1000];
- `conv_en_lim` [real]: This is the parameter f_L used to determine when to stop the scattering iterations (see section 2.3). This happens when, at the end of a scattering iteration, the remaining scattered luminosity to be processed is less than f_L times the total scattered luminosity;
- `dist_obs` [real]: External observer distance [pc] used when calculating integrated SED;
- `ind_i_obs` [integer]: List of indices of the wavelengths of the observer specific intensity arrays (i_obs) to be printed on disk. By default ind_i_obs contains all the wavelength indices. Note that the first index is 0 not 1;
- `ind_out_maps` [integer]: List of indices of the wavelengths for which the surface brightness maps have to be printed. By default ind_out_maps contains all the wavelength indices. Note that the first index is 0 not 1;
- `kp_maps` [integer]: Parameter determining the number of pixels on the internal observer surface brightness maps. The number of pixels is equal to $12*2^{2\mathtt{kp\_maps}}$;
- `kp_sca_max` [integer]: HEALPix parameter used to determine the maximum allowed number of angular directions for the sampling of the scattering source function arrays (the "`scaspe`" arrays): `npix_max` = $12*2^{2\mathtt{kp\_sca\_max}}$. The allowed range is currently [0,4]. However, this can be changed in the subroutine `check_input`;
- `lambda_ref` [real]: Reference wavelength in μm. Its use depends on the particular model. For calculations without lambda grids, this is the wavelength for the stellar emission and dust density stored on the main grid file. For calculation using lambda grids, the reference wavelength has to coincide with the wavelength of a lambda grid. In this case, the corresponding lambda grid is used as a reference to check that the dust opacity in all lambda grids is consistent with the assumed dust model;

- map_size_factor [integer]: The linear size of the external observer maps is equal to map_size_factor*modelsize (see section 4.1 for the definition of modelsize). map_size_factor cannot be less than 0.2. However, a value of at least 1.8 is recommended to avoid cells being projected outside the observer map. Note that if the 'i_obs' files are printed on disk, the maps can be recalculated using the 'sed' RT algorithms;
- max_lambda_stars [real]: Maximum wavelength used for the stellar emission wavelength grid [μm]. If this is not input, the code will consider all wavelengths in the input wavelength grid for the stellar emission RT calculation;
- max_sca_iterations [integer]: Maximum number of scattering iterations. It requires limit_scattering_iterations = .TRUE.
- min_lambda_dust [real]: Minimum wavelength in the dust emission RT calculations [μm]. If this is not input, the code assumes the default value min_lambda_dust = 1 micron;
- n_dust_size_qabs [integer]: see section 4.1;
- n_dust_temp_cal [integer]: Number of dust temperatures in the input files for the grain specific enthalpy/ specific capacity (see file_calorimetry_Gra and file_calorimetry_Sil).
- n_dust_wave_qabs [integer]: see section 4.1;
- npixel_maps [integer]: Number of pixels for each side of the external observer maps;
- rad_lim [real]: Maximum distance relative to modelsize (see section 4.1) that can be crossed by the rays in the pre-calculation phase. Allowed range = [0,2];
- x_wall_coord [real]: Positions of the X-axis perpendicular walls in relative coordinates (0 = -modelsize/2., 1 = modelsize/2);
- y_wall_coord [real]: Positions of the Y-axis perpendicular walls in relative coordinates (0 = -modelsize/2., 1 = modelsize/2);
- z_wall_coord [real]: Positions of the Z-axis perpendicular walls in relative coordinates (0 = -modelsize/2., 1 = modelsize/2);

&**dartray_input_logical** ! All parameters in this block are of LOGICAL type

- input_av_opacities: : see section 4.1;
- limit_scattering_iterations: TRUE if a maximum number of scattering iterations has to be set using max_sca_iterations;
- no_communications: TRUE if no communication mode is to be used. This means that the scaspe and i_obs arrays are not distributed among the different MPI processes and copies of the entire arrays are created for each MPI process instead. This mode requires more memory for each node and is recommended when either considering a small number of wavelengths or a relatively low 3D grid resolution. The advantage of using this method is that no communication is performed between nodes during the RT calculation. It is the fastest parallelized mode available in DART-Ray;
- no_dust_rt: TRUE if no dust emission RT has to be performed;
- only_direct_rt: TRUE if only direct light is to be processed;
- print_maps: TRUE if surface brightness maps for the external observers have to be calculated and printed;
- print_maps_in: TRUE if surface brightness maps for the internal observers have to be calculated and printed;
- print_output_part1: TRUE if files 'ufield_part1' have to be saved at the end of pre-calculation phase;

- print_output_part2: TRUE if the output of part 2 has to be written on disk. This should be done if one needs the u_final, i_obs and i_obs_in arrays with radiation contributions only due to direct light. Note that the scaspe arrays are not written by default. To print those arrays as well at the end of the direct light processing, you need to set print_scaspe_part2 = .TRUE. If you print all the part 2 output, including the scaspe array, you will be able to restart the calculation directly from the first scattering iteration;

- print_psel_av: TRUE if psel_av_arr array is to be output;

- print_scaspe_part2: TRUE if scaspe arrays have to be written on disk at the end of the direct light processing. This is not implied by default in print_output_part2 because the scaspe arrays can take a LOT of disk space. If you want to be able to restart the calculation from the scattering iterations, the 'scaspe' files for part 2 are necessary;

- print_scaspe_tot: TRUE if 'scaspe_tot' files have to be printed at the end of the scattering iterations. The scaspe_tot arrays can then be used in RT algorithm 'i_obs' (see rt_algorithm).

- print_sed: TRUE if 'sed' files have to be written;

- restore_file_mpi: TRUE if previously written files have to be restored during runs with more than one MPI process. In case only one MPI process is present, a choice is requested on the terminal;

- sequential_scattering: TRUE if the scattered luminosity is propagated only for the same scattering order during each scattering iteration;

- test_run: TRUE if the ray tracing routines do not have to be run. Useful to check that the subroutines are running fine and the output arrays are correctly saved. However, the results will be incorrect and therefore these files should be deleted afterwards;

- use_dir_out: TRUE if file_dir_out has to be read;

- use_lambda_grid: TRUE if lambda grids have to be used;

- use_p_src: TRUE if file_p_src has to be read;

- use_pos_obs: TRUE if file_pos_obs has to be read;

- x_wall_on: TRUE if wall perpendicular to X direction is set;

- y_wall_on: TRUE if wall perpendicular to Y direction is set;

- z_wall_on: TRUE if wall perpendicular to Z direction is set.

5.3.2 Choosing the RT algorithm

&**dartray_input_strings**: rt_algorithm
&**dartray_input_logical**: no_dust_rt

The choice of the RT algorithm to use (using rt_algorithm) depends on your needs for the calculation. The usual choice is the 'main' algorithm, which allows you to calculate all possible output: the radiation field energy density, the scattering source function, the radiation specific intensity reaching the observers and the observer surface brightness maps. This algorithm can produce these quantities both for the stellar emission and the dust emission and it is required for non-axisymmetric models and with few observer lines-of-sight/internal positions (e.g. < 10). If only the stellar emission RT calculations are needed, one can set no_dust_rt = .TRUE..

If your RT model is axisymmetric, you can still use the 'main' algorithm, but note that the '2D' RT algorithm is a faster alternative. This algorithm is intrinsically 3D, but it takes advantage of the symmetries in the RT model and it avoids the unnecessary repetition of many calculations. In this way, it can perform RT calculations up to 8 times faster than the 'main' algorithm. However, for technical reasons, the calculation of surface brightness maps is not implemented in this algorithm. You can still calculate maps by outputting the scattering source function (using print_scaspe_tot) and then by using the 'i_obs' algorithm (see below).

If the stellar emission RT has already been performed, using the 'main' or the '2D' algorithms, and the corresponding radiation field energy density files printed on disk, it is possible to run only the dust emission RT using the 'dust' and 'dust_2D' algorithms. In some cases you may want to perform the dust emission calculation, using e.g. a different type of dust heating/emission (see keyword dust_heating_type), without repeating the stellar emission RT calculation. The 'dust_2D' mode can only be used for axisymmetric models, as is the case with the '2D' algorithm.

If the radiation specific intensity 'i_obs' files have been printed using the 'main' or 'dust' modes, it is possible to recalculate the integrated SEDs, or the surface brightness maps, for the same observer lines-of-sight/positions used in the complete RT calculations. This can be performed using the 'sed' or 'sed_dust' modes (the first for the stellar emission and the second for the dust emission). Recalculating the SEDs or the maps may be required, if one wants to use different parameters for the maps (e.g. number of pixels), for the SEDs (e.g. observer distance), or if these files have been deleted by mistake.

If one wants to calculate the radiation specific intensity reaching the observers, and the corresponding surface brightness maps for many observer lines-of-sight/positions, it is often unpractical to use the standard modes ('main'/'dust') because the memory requirements may be very high and the RT calculations very slow. For this reason, one can use the 'i_obs' and 'i_obs_dust' modes. These modes accept arbitrary lists of observer lines-of-sight or internal 3D positions and calculate the 'i_obs' and 'maps' files provided that the scattering source function has already been saved on disk. Note that the scattering source function files are not required if only the direct light is to be included. This can be done by using only_direct_rt. Note also that the additional label '_int' is added to the output files names to distinguish them from those produced using the other modes.

Finally, another RT algorithm is the 'projection' algorithm, which can be used to obtain quickly surface brightness maps of the stellar emission without dust, as well as maps of optical depth at any wavelength. This mode is recommended to ensure that the input stellar emission and dust opacity distribution are correctly defined in the 3D main grid and lambda grid files.

5.3.3 Choosing the output files

&**dartray_input_strings**: label_model_out, dir_runs
&**dartary_input_logical**: print_maps, print_maps_in, print_output_part1, print_output_part2, print_psel_av, print_scaspe_part2, print_scaspe_tot, print_sed

The output file names for a specific RT calculation can be distinguished by selecting an arbitrary label with label_model_out. This label is contained in all output file names, together with the labels corresponding to the type of output file (see section 5.2). The directory where the output files are saved can be specified with dir_runs.

The output of DART-Ray can be large in terms of memory. For this reason, there are several options controlling the saved output files. The logical input variables regulating the output all start with 'print_'. If one does not specify any of the 'print_' variables, they are all set equal to .FALSE. by default. In this case, if one runs any of the RT algorithms, the output files will be only the 'ufield' and the 'i_obs' files. To save the integrated SEDs for the radiation reaching the external observer, set print_sed = .TRUE.. If the surface brightness maps are to be calculated and saved, one can use the keywords print_maps (external observer maps) and print_maps_in (internal observer maps). To output quantities for the direct light, set the keyword print_output_part2. By default, this will print the 'ufield' and the 'i_obs' files at the end of the direct light processing. For the direct light quantities, the SEDs and the maps are saved only if the keywords print_sed, print_maps and print_maps_in are .TRUE.. The scattering source function file, as derived at the end of the direct light processing, is printed only if print_scaspe_part2 is .TRUE.. To save the final values of the scattering source function, use print_scaspe_tot. The latter is necessary if one wants to use the 'i_obs' algorithm including the scattered light contributions. It is also possible to save the distribution of the lower limit of the radiation field energy density by using the keyword print_output_part1.

5.3.4 Including point sources

&**dartray_input_strings**: `file_param_src, file_p_src`
&**dartary_input_logical**: `use_p_src`

Point sources can be inserted at arbitrary positions within the 3D grid. To do that, one needs to set the logical keyword `use_p_src` = .TRUE. and provide the point source 3D positions in `file_p_src`. The code reads this file and allocates the luminosity arrays for the sources. The source spectra is set within the user_routines_model modules (see e.g. subroutine `read_assign_param_src` within user_routines_magtar.f90). It is not possible to have more than two point sources within a single 3D grid element. Also, in the 2D mode, only a single source at the centre of the grid is allowed. If you want to provide a set of input parameters to determine the source spectra, you can input them in the file `file_param_src`. There is no specific format you have to use, since this file has to be read in the user_routines_model modules.

5.3.5 Choosing the accuracy

&**dartary_input_var**: `accuracy`

The accuracy of RT calculations depends on many factors, including the grid resolution in all dimensions (spatial, angular and wavelength). The error in the grid discretization is difficult to determine during an RT calculation and it can only be measured by performing RT calculations with successively higher resolutions. DART-Ray contains another source of inaccuracy, due to the blocking of rays carrying "negligible" contributions to the RFED. The inaccuracy due to the ray blocking can be limited by setting the parameter `accuracy`. The value assigned to this parameter is the maximum inaccuracy that one wants to allow for the RFED. For example, to set a maximum inaccuracy of 5%, one has to set `accuracy` = 0.05. This parameter determines the value of f_U, the threshold parameter actually used to decide whether the rays have to be blocked or not (see section 2.2).

5.3.6 Choosing the type of dust emission spectra

&**dartray_input_strings**: `dust_heating_type`

Three types of dust emission spectra can be calculated with DART-Ray, which can be selected using the keyword `dust_heating_type`: equilibrium emission from a single grain with effective dust properties ('eff'); total equilibrium emission from each single grain defined by the input grain size distributions ('equ'); and full stochastically heated dust emission ('sto'). The last type is the more computationally demanding of the three. However, it derives the most realistic mid-infrared dust emission spectra, including the PAH line emission features.

5.3.7 Choosing the wavelength grid

&**dartray_input_strings**: `file_lambda_list`
&**dartary_input_var**: `max_lambda_stars, min_lambda_dust`

A wavelength list in μm has to be provided within the input file `file_lambda_list`. There is no constraint on the number of wavelengths that can be input, or their distribution with wavelength, if only the radiation field and surface brightness maps are to be calculated. This is because the stellar emission RT can be handled independently at each wavelength. It is strongly advised to have at least about 15-20 wavelengths covering the UV to NIR range and at least 15-20 wavelengths covering the **entire** infrared regime up to the submm, if one wants to also calculate the dust emission. The results for both the infrared radiation fields and the dust emission maps are dependent on the resolution of the wavelength grid in the entire UV to submm range. In particular, you have to make sure that the infrared range is covered completely, from the NIR to the submm, otherwise the derived dust emission temperature and spectra will be **VERY** wrong!

Although the entire wavelength grid can be specified in `file_lambda_list`, there is no need to perform the RT calculation at all wavelengths for both the stellar emission and the dust emission.

In the case of the stellar emission, one can set a maximum wavelength using `max_lambda_stars`. Only the wavelengths in the list less than this maximum value will be considered in the stellar emission RT calculation. Usually a maximum wavelength in the NIR-MIR range is adequate, since the stars typically emit only a very small fraction of their luminosity at wavelengths $> 10\mu$m. One can set a minimum wavelength for the dust emission RT using `min_lambda_dust`. Wavelengths higher than this value will only be considered in the dust emission RT. A safe choice is to set this value equal to 1μm, since dust emission tends to be negligible at wavelengths lower than this value.

5.3.8 Choosing the ray angular density

&**dartary_input_var**: `bm_par, bm_par_max, bm_par_sca`

Another factor, which can affect the accuracy of the calculation, is the minimum number of rays from a radiation source which intersect the cells of the 3D grid. This number should be of at least a few rays for an accurate calculation. Too few rays make the calculation inaccurate, while too many make it very slow, so a balanced choice should be made. Typically, ~ 5 rays is a good value to use. You can set the minimum number of rays per intersected cell using two input parameters: `bm_par` (the minimum number of rays per cell used in the pre-calculation and in the direct light processing for the stellar emission RT) and `bm_par_sca` (used in the scattered light processing for the stellar emission RT and in all phases for the dust emission RT). One can also set the maximum number of rays per cell using the parameter `bm_par_max`. In some cases, this can be useful because the grid can become gradually coarser in spatial resolution and the number of rays crossing large cells much higher than necessary. The number of rays is reduced by merging them when their number is higher than the maximum allowed number. A value for `bm_par_max` of the order of 150-200 might be appropriate. Smaller values, although acceptable, may lead to several merging - splitting cycles during the ray propagation, which are computationally inefficient.

5.3.9 Choosing the observer positions and the surface brightness map parameters

&**dartray_input_strings**: `file_dir_out, file_pos_obs`
&**dartary_input_var**: `ind_out_maps , kp_maps, map_size_factor, npixel_maps`
&**dartary_input_logical**: `print_maps, print_maps_in, print_sed, use_dir_out, use_pos_obs`

Observers can be located either far away or inside an RT model. In the former case, one has to specify the lines-of-sight directions of the distant observers. This can be done by setting `use_dir_out` = .TRUE. and by including the angles defining the lines-of-sight directions within the file `file_dir_out`. For each line-of-sight, the angles to specify are θ (the angle between the line-of-sight and the Z-axis of the 3D grid) and ϕ (the angle between the projection of the line-of-sight on the XY plane and the X-axis). The two angles have to be specified in radians and the first line of the file has to be a comment.

In the case of internal observers, one has to specify the 3D positions of those observers within the models. This can be done by setting `use_pos_obs` = .TRUE. and by specifying the X, Y, Z coordinates of the observers within the file `file_pos_obs`. In this case, the first line of the file has to be a comment, also.

If the above keywords are set and the input file provided, the code output the radiation specific intensity files, including the 'i_obs' and 'i_obs_in' arrays, but not necessarily the corresponding surface brightness maps. To calculate and save the maps, one has to set the keywords `print_maps` and `print_maps_in` for the far-away observers and the internal ones, respectively. For the external observers maps, one can set the number of pixels per map side using `npixel_maps` and the map extent using `map_size_factor`. The latter gives the map physical size in units of fraction of `modelsize` (see section 4.1). For the internal observers, one can set the number of spherical pixels of the full sky surface brightness maps using `kp_maps` (the number of pixels is $12 * 2^{2*kp\_maps}$). Note that the HealPix scheme is used in the NESTED format.

Typically, one performs the RT calculation at many wavelengths, but the surface brightness maps may not be required at all wavelengths, especially when the pixel number is high and the maps require a lot of disk space. To select the wavelengths for which the maps have to be printed on the 'maps' and 'maps_in' files, one can use `ind_i_maps`. Using this variable, one can input an arbitrary number of indices corresponding to the wavelengths in `file_lambda_list` for which the surface brightness maps are to be saved.

5.3.10 Choosing the limit radius for the pre-calculation phase

&**dartary_input_var**: `rad_lim`

During the pre-calculation phase, the radiation transfer calculations are performed from each radiation source until a user defined limit radius is reached. This radius can be input using `rad_lim`, which is the limit radius in units of `modelsize` (see section 4.1). Typically, it is sufficient to use `rad_lim` = 0.1- 0.2. If the radiation sources are confined to a relatively small fraction of the entire RT model, it may be that there are parts of the 3D grid which are not intersected by any ray during the pre-calculation phase. Thus, for those 3D grid cells the lower limit of the RFED will be zero and the criteria for blocking the rays will not work if a ray intersects any of those cells. For example, in a disc galaxy model, where all the stellar emission is close to the galaxy plane, the pre-calculation will probably fail to provide a lower limit to the RFED at high vertical distances from the plane. This could slow down the calculation during the direct light processing phase. However, if the distribution of both stars and dust does not extend much vertically, and one is not interested in calculating the RFED at high vertical distances, one can solve this problem by applying walls parallel to the galaxy plane and embedding the entire distribution of stars and dust (see section 5.3.22).

5.3.11 Choosing maximum angular resolution for the scattering source function

&**dartary_input_var**: `kp_sca_max`

In order to save RAM memory, DART-Ray adopts a wavelength-dependent angular resolution for the scattering source function. At long infrared wavelengths, where scattering is more isotropic, a lower angular resolution for the scattering source function is sufficient, while a higher resolution is required at short UV/optical wavelengths. The number of spherical pixels defining the angular directions stored in the scattering source function is given by $n_{pix} = 12 * 2^{2kp\_sca}$ where *kp_sca* is an integer value (the HealPix NESTED pixelation scheme is used to subdivide the sphere). DART-Ray contains criteria to set the value of *kp_sca* automatically at each wavelength, depending on the value of the scattering anisotropy parameter g_λ. However, it requires a maximum allowed value of *kp_sca* to avoid extremely high angular resolution at very short wavelengths. This maximum value can be input using `kp_sca_max`. A suggested value is 2. Lower values make the calculation run faster and reduce the RAM memory requirements at the expense of lower accuracy. Higher values have the opposite effect, and may make the calculations impossible to run, but that depends on the specifications of your machine (in particular the RAM memory).

5.3.12 Choosing the luminosity threshold parameter to stop the scattering iterations

&**dartary_input_var**: `conv_en_lim`

DART-Ray ceases scattering iterations when the unprocessed scattered radiation luminosity falls below a threshold fraction of the total scattered luminosity (the one derived at the end of the direct light processing). This fraction can be set using `conv_en_lim` (equivalent to f_L, see section 2.3). Recommended values are in the range 0.001 - 0.01. Note that the criteria to stop the scattering iterations has to be satisfied at all wavelengths separately.

5.3.13 Including the lambda grids

&**dartray_input_strings**: `label_model_lambda_grid, dir_grid`
&**dartray_input_var**: `lambda_ref`
&**dartary_input_logical**: `use_lambda_grid`

DART-Ray uses lambda grids only if the keyword `use_lambda_grid` is set to .TRUE.. In that case, it searches for the lambda grid files in the `dir_grid` directory. These files should contain the same label `label_model_lambda_grid` that was specified when they were created by the grid creation programs.

DART-Ray checks that dust opacity distribution stored in each of the lambda grid files is consistent with the dust model assumed in the calculation. For this purpose, a reference wavelength, corresponding to the lambda grid to which all the other grids are compared, has to be specified using `lambda_ref`.

5.3.14 Setting the units for the luminosity, cell size and wavelength

&**dartray_input_strings**: `units_luminosity, units_csize, units_lambda`

DART-Ray accepts stellar luminosities in two units ('W/Hz' or 'erg/s/Hz') which have to be specified in `units_luminosity`. This is crucial for calculating the dust heating/emission correctly. Currently, DART-Ray accepts only cell size units in pc and wavelength units in μm. The two input variables specifying these units `units_csize, units_lambda` have to be present in the input file, but they are just there to remind the user of the restrictions on the units.

5.3.15 Setting the dust model

&**dartray_input_strings**: `dust_model, dust_opacity_tables, file_gra_fa, file_sil_fa, file_pah_neu_fa, file_pah_ion_fa, file_av_opacities, file_q_gra, file_q_sil, file_q_pah_neu, file_q_pah_ion, file_calorimetry_gra, file_calorimetry_sil`
&**dartray_input_var**: `n_dust_temp_cal`

The dust model has to be specified in the input files for the RT calculation using the same input variables that are used in the grid creation routines (see section 4.4). Note that the calorimetry files `file_calorimetry_gra` and `file_calorimetry_sil` have to be provided, if the user chooses a 'user' `dust_model` and `dust_opacity`, as well as the stochastically heated dust emission with `dust_heating_type`.

5.3.16 MPI Parallelization: 'no communication' and 'communication' mode

&**dartary_input_logical**: `no_communication`

In addition to the OpenMP parallelization, which is used to run the code on a single shared memory machine, DART-Ray allows for two modes of parallelization using MPI to use the code on computer clusters. These two modes regulate the way the information is exchanged between the cluster nodes. The suggested mode is the "no communication" mode, which can be used by setting `no_communication` = .TRUE.. In this mode, all the arrays used in the calculation are replicated in all computer cluster nodes and communication is performed only at the end of each radiation transfer calculation step, when the results from the single nodes are combined. This mode is relatively expensive in terms of memory, but it is the fastest mode because inter-node communication and processing of the exchange information is reduced to a minimum. If the amount of memory to be used is very high, one can use the "communication" mode by setting `no_communication` = .FALSE.. In this mode, the scattering source function and the 'i_obs' arrays are not replicated, but they are distributed among the nodes instead. Some communication is periodically needed during the calculation, together with incoming data processing, which makes this mode significantly slower compared to the "no communication" mode.

5.3.17 Avoiding the dust emission RT

&**dartary_input_logical**: `no_dust_rt`

It is possible to run the RT algorithms 'main' and '2D' only for the stellar emission RT, without performing the dust emission RT. This can be done by setting `no_dust_rt` = .TRUE.. Once the radiation field files have been saved, the dust emission RT can be started using the 'dust' and 'dust_2D' algorithms.

5.3.18 Running DART-Ray in TEST mode

&**dartary_input_logical**: `test_run`

Before running long RT calculations, it is worth running a simulation in the TEST mode (`test_run` = .TRUE.). That is, all the processes of input reading, array allocation, and output file writing are performed without the time consuming radiative transfer calculations. In this way, one can quickly check that there is sufficient disk space for the output. It would be frustrating following a long calculation to discover that the output cannot be written because of lack of disk space or writing permission! Obviously, the output files do not contain useful results and should be deleted before starting the actual calculation.

5.3.19 Saving intermediate step output files and restarting the calculations from intermediate steps

&**dartary_input_logical**: `print_output_part1, print_output_part2, print_scaspe_part2, restore_file_mpi`

RT calculations can be very long and for several reasons you may want to interrupt a calculation without having to restart it from the beginning. In the case of the stellar emission RT, this can be done by using all possible output files from the pre-calculation phase, or from the direct light processing phase. If one sets `print_output_part1`, the files for the pre-calculation step are written on disk. Then one is able to restart the RT calculation directly from the beginning of the direct light processing. If one sets `print_output_part2` as well as `print_scaspe_part2`, all the necessary output files from the direct light processing step are written, so that one can restart the calculation directly from the beginning of the scattering iterations. Note that the code exits if any of the output files written at the end of the scattering iterations is found. Also, restarting from intermediate points happens only if all the necessary intermediate files are written and the keyword `restore_file_mpi` = .TRUE..

5.3.20 Sequential scattering mode and scattering iteration limit

&**dartary_input_var**: `max_sca_iterations`
&**dartary_input_logical**: `sequential_scattering, limit_scattering_iterations`

Each scattering iteration in DART-Ray does not necessarily represent the processing of a single order scattered light (e.g. light scattered only for the first/second/third time). This might create some anisotropies in the radiation fields and surface brightness maps for some particular models (especially very optically thick models). To avoid this, it is possible to process the scattered light one order at the time using the keyword `sequential_scattering` = .TRUE.. Note that this "sequential scattering" mode may require significantly more RAM memory because of some additional arrays that are needed.

One can also limit the number of scattering iterations by setting `limit_scattering_iterations` = .TRUE. and inputting the maximum number of scattering iterations allowed using `max_sca_iterations`.

5.3.21 Processing only the direct light

&**dartary_input_logical**: `only_direct_rt`

In case one wants to run the RT calculation including only the direct light processing, without the scattering iterations, this can be one by setting `only_direct_rt` = .TRUE..

5.3.22 Setting walls

&**dartary_input_var**: `x_wall_coord, y_wall_coord , z_wall_coord`
&**dartary_input_logical**: `x wall on, y wall on, z wall on`

If the distribution of sources and dust in the 3D model you are considering occupies only a small part of the 3D grid, and you are interested in obtaining the surface brightness maps and not the radiation field energy density at all points in the RT model, you can limit the calculation of the RFED to within a subvolume of the entire grid volume. This can be done by using "walls" to define planes perpendicular to the main grid axis, which delimit this subvolume. To set the walls, you can use the keywords `x_wall_on`, `y_wall_on` and `z_wall_on` to set pairs of walls perpendicular to the X, Y and Z axis respectively. To set the position of these walls, you have to use the input variables `x_wall_coord`, `y_wall_coord` and `z_wall_coord`. If the corresponding logical variable is set, you can use these variables to input the relative coordinates of the wall using two numbers between 0 and 1. For example, to set the walls perpendicular to the X axis, one at 20% and the other at 80% of the entire model size, you can input `x_wall_on` = .TRUE. and `x_wall_coord` = 0.2, 0.8. You can use all three types of walls simultaneously, but note that the results for the radiation fields within the subvolume delimited by these walls and the surface brightness maps will be correct only if the subvolume includes the entire distribution of stars and dust. The exception are the point sources for which the walls do not have any effect. Therefore, you can place point sources outside the region delimited by the walls, but still get the correct results for the RFED within the delimited region and the surface brightness maps.

5.3.23 Measuring the average path crossed by rays

For diagnostic reasons you may want to know the average path lengths of the rays departing from each radiation source in the RT model. You can learn that by setting `print_psel_av = .TRUE.`. Then the code will calculate this quantity for each radiation source and for each calculation phase (direct light processing, scattering iterations), and output the 'psel_av' files at the end of the RT run.

Chapter 6

Grid creation and RT run examples

In this chapter, we show some complete examples of grid creation and RT runs using the four DART-Ray programs included in the public release.

Before running any of the commands below, remember to set the number N of OpenMP threads using:

```
setenv OMP_NUM_THREADS N
```

for csh shells or:

```
export OMP_NUM_THREADS=N
```

for bash shells, with N equal to the number of CPUs you would like to use on each shared memory machine. The examples below are for using DART-Ray with a single computer. The grid creation programs are meant to be used on a single machine, so you should always launch them as shown below. The DART-Ray RT programs are MPI+OpenMP parallelized and can be used on computer clusters. To do that, you should use the standard commands to launch MPI applications, such as `mpirun`, preceding on the same line the commands shown below. Note that, most probably, you will need to prepare an appropriate batch scheduling script to use DART-Ray on a computer cluster (e.g. a PBS script, see documentation for the system you are using).

6.1 Analytical galaxy model

In this example, we create the grids for an analytical galaxy model and run the corresponding stellar emission and dust emission RT calculations. This model is a low resolution version of the Milky Way galaxy model presented in Popescu et al. 2017.

6.1.1 Grid creation

To create the main grid for the example model, `cd` to the directory where you installed DART-Ray and type:

```
./create_adap_grid_galaxy ./GALAXY_GRIDS/input_grid_galaxy_example1.in
```

(the program will ask you to check that the names for the output files are correct. Press RETURN to continue with the grid creation).

After the main grid creation has been completed, retype the above command to calculate the lambda grids. When complete you will find all the grid files in the directory `./GALAXY_GRIDS/EX1`.

The grid characteristics depend on the content of the input file `input_grid_galaxy_example1.in`. We show its content below, together with explanations for each line.

&galaxy_input_strings ! NAMELIST block including all the character input variables.

`label_model_lambda_grid=‘model_ex1’,` ! Label contained in the lambda grid file names.

`dir_grid=‘./GALAXY_GRIDS/EX1’,` ! Directory where grids and auxiliary files are located (e.g. wavelength list).

`grid_file=‘grid_model_ex1_main.h5’,` ! Main grid file name.

`grid_info_file=‘grid_model_ex1_info.dat’,` ! Info file name.

`file_lambda_list=‘lambda_list_pt11.dat’,` ! Wavelength list file name. Wavelengths have to be in microns and in ascending order. Note that this file has to be present in `dir_grid` before calculating the grids.

`units_lambda = ‘um’,` ! Wavelength units (always microns).

`grid_type=‘all’,` ! Type of grid. ‘all’ means that all stellar components are included.

`old_disk_type=‘flared_sech2z’,` ! Type of old stellar disk: ‘flared_sech2z’ means flared profile with hyperbolic secant square profile in the vertical direction.

`young_disk_type=‘flared_sech2z’,` ! Type of young stellar disk.

`thick_disk_type=‘flared_sech2z’,` ! Type of thick dust disk.

`thin_disk_type=‘flared_sech2z’,` ! Type of thin dust disk.

`dust_model = ‘DraineLi06’,` ! Assumed dust model. ‘DraineLi06’ is the dust model of Draine and Li 2007.

`file_av_opacities = ‘./DUST_OPACITY/DraineLi06/kappagrainwd01_q06_effective.dat’` ! File containing the integrated/average opacity coefficient calculated indipendently by the user. This requires `input_av_opacities = .TRUE.`. Not required when using the standard dust models, but recommended for user-provided dust models.

`file_old_star_sed = ‘pt11_old_star_sed.dat’` ! File containing the units luminosities for the old stellar disk and bulge. This file has to be located in `dir_grid`. The total luminosity of the old stellar disk at each wavelength will be equal to `old` times the corresponding luminosity interpolated from the unit luminosity spectra in this file. For the bulge, the total luminosity will be equal to `old`*`bd_ratio` times the unit luminosity.

`file_young_star_sed = ‘pt11_young_star_sed.dat’` ! File containing the units luminosity for the young stellar disk. Same comments as for `file_old_star`, but the scaling factor for the young stellar disk is `sfr`, not `old`.

`subdivision_criteria = ‘standard’` ! Subdivision criteria used during the main grid creation. ‘Standard’ criteria includes maximum cell luminosity and optical depth thresholds, as well as an inner region forced to have the highest possible resolution.
/

&galaxy_input_var ! NAMELIST block containing numerical parameters used in the grid creation.

`lambda_ref= 0.443,` ! Reference wavelength, used for the main grid creation [μm].

`lambda_min= 0.090,` ! Minimum wavelength for lambda grid creation [μm].

`lambda_max= 5.0 ,` ! Maximum wavelength for lambda grid creation [μm].

`rtrun=14000.,` ! Truncation radius [pc].

`rsun=8000.,` ! "Sun" radial distance [pc].

`max_z=4000,` ! Maximum vertical height for stars and dust [pc].

`max_rad=14000,` ! Maximum radial distance for stars and dust [pc].

`sha=0.012,` ! Smoothing parameter for profile values close to truncation radius.

`sha1=0.012,` ! As above but for the thin dust and stellar disks.

`modelsize = 28000.  ,` ! Model linear size [pc].

`base=3,3,` ! Subdivision factors.

`max_ncell=3.E6,` ! Maximum number of grid cells allowed.

`max_lvl=4,` ! Maximum subdivision level allowed.

`min_lvl=2,` ! Minimum subdivision level for all grid cells.

`max_dtau=0.01,` ! Maximum cell optical depth allowed at the reference wavelength.

`max_dlum=0.5E-6` ! Maximum cell luminosity allowed in units relative to the total model luminosity at the reference wavelength.

`z_subd_lim = 50` ! Vertical limit for the inner region within which the highest resolution is applied [pc].

`R_subd_lim = 4000` ! Radial limit for the inner region within which the highest resolution is applied [pc].
/

&galaxy_input_var_old_disk ! Old stellar disk parameters.

`old =0.353,` ! Scaling factor for the old stellar luminosity.

`hs_disk_b=3200.,` ! Standard scale length [pc].

`zs_disk=140,` ! Standard scale height [pc].

`zs_disk_r1=170.,` ! Scale height at the inner radius `hsin` [pc].

`zs_disk_rsun=300.,` ! Scale height at the sun radial distance [pc].

`chi_disk = 0.5,` ! Parameter determining the slope of the radial profile within the inner radius `hsin`.

`hsin=4500` ! Inner radius [pc].

`id_hs_disk_arr = 12,13,14,15` ! Wavelength indices for the corresponding scale length values specified in `hs_disk_arr`. Note that wavelength counter starts with zero.

`hs_disk_arr = 1800, 2200, 2400, 2600 !` Scale lengths for the wavelength specified in `id_hs_disk_arr` [pc]. Given the wavelength list in 'lambda_list_pt11.dat', the wavelengths for which these scale lengths are applied are 1.2, 2.2, 3.5 and 4.9 μm.
/

&galaxy_input_var_young_disk ! Young stellar disk parameters .

`sfr=1.,` ! Scaling factor for the young stellar disk luminosity.

`hs_tdisk=3200,` ! Scale length [pc].

`zs_tdisk=50,` ! Standard scale height [pc].

`zs_tdisk_r1=67,` ! Scale height at the inner radius `hs1in` [pc].

`zs_tdisk_rsun=90,` ! Scale height at the sun radial distance [pc].

`chi_tdisk = 0.5,` ! Parameter determining the slope of the radial profile within the inner radius `hs1in`.

`hs1in=4500,` ! Inner radius for the young stellar disk [pc].
/

&galaxy_input_var_bulge ! Bulge parameters .

`reff=400.,` ! Effective radius [pc].

`acap_bulge=40.,` ! Inner truncation radius [pc].

`ellipt=0.3,` ! Bulge ellipticity.

`mtrunc=3,` ! Bulge truncation radius in units of effective radius.

`bd_ratio=1.,` ! Bulge-to-disk luminosity ratio. See `file_old_star_sed`.

`nsersic=4` ! Sersic index.
/

&galaxy_input_var_thick_dust_disk ! Thick dust disk parameters.

`tau1= 3.17,` ! Central vertical optical depth at 4430 A.

`hd_disk=5200.,` ! Scale length [pc].

`zd_disk=140.,` ! Standard scale height [pc].

`zd_disk_r1=140.,` ! Scale height at the inner radius `hdin` [pc].

`zd_disk_rsun=140.,` ! Scale height at the sun radial distance [pc].

`chi_dust_disk = 0.5,` ! Parameter determining the slope of the radial profile within the inner radius `hdin`.

`hdin= 4500,` ! Inner radius [pc].
/

&galaxy_input_var_thin_dust_disk ! Thin dust disk.

`tau2= 0.61 ,` ! Central vertical optical depth at 4430 A.

`hd_tdisk=3200,` ! Scale length [pc].

`zd_tdisk=50,` ! Standard scale height [pc].

`zd_tdisk_r1=67,` ! Scale height at the inner radius `hd1in` [pc].

`zd_tdisk_rsun=90,` ! Scale height at the sun radial distance [pc].

`chi_dust_tdisk = 0.5,` ! Parameter determining the slope of the radial profile within the inner radius `hd1in`.

`hd1in=4500,` ! Inner radius [pc].
/

&galaxy_input_logical ! Logical variables.

`input_av_opacities = .TRUE.` ! If set TRUE, then `file_av_opacities` is read and compared with integrated/average opacity factors calculated by DART-Ray from the opacity coefficient of single grains.
/

6.1.2 RT runs

Once the grids have been created, one can launch the RT run using the following command (DO NOT TYPE IT YET, READ THE FOLLOWING FIRST):

```
./dartray_galaxy ./GALAXY_GRIDS/input_galaxy_example1.in.
```

The input file `input_galaxy_example1.in` included in the public release is prepared for an RT calculation for the stellar emission using the ‘2D’ algorithm (3D RT algorithm performed on a axisymmetric 3D grid, but some calculations are avoided by taking advantage of the symmetries). We explain its content line by line below. By making minor changes to this file, you can run different types of RT algorithms as explained in the subsections below.

&dartray_input_strings NAMELIST block containing all character variables.

`label_model_lambda_grid=‘model_ex1_all’,` ! Label contained within the lambda grid files. Note that the additional `_all` which comes from `grid_type` in the grid creation input file.

`label_model_out=‘model_ex1_all’,` ! Label contained in the output files.

`grid_file=‘grid_model_ex1_main.h5’,` ! Main grid file name.

`file_dir_out=‘dir_out_galaxy.dat’,` ! File containing the lines-of-sight directions for the external observer.

`file_lambda_list=‘lambda_list_pt11.dat’,` ! Wavelength list file.

`dir_runs=‘./RUNS/GALAXY_EX1’,` ! Directory where the output files have to be saved. It is created automatically when absent.

`dir_grid=‘./GALAXY_GRIDS/EX1’,` ! Directory where the grids are located.

`rt_algorithm=‘2D’` ! RT algorithm to be used. ‘2D’ is the RT algorithm for the stellar emission RT, which can be used for axisymmetric models.

`units_lambda = ‘um’,` ! Wavelength units (always microns).

`dust_model = ‘DraineLi06’,` ! Assumed dust model. ‘DraineLi06’ is the dust model of Draine and Li 2007.

`file_av_opacities = ‘./DUST_OPACITY/DraineLi06/kappagrainwd01_q06_effective.dat’,` ! File containing the integrated/average opacity coefficient calculated indipendently by the user. To read this file, it is required that `input_av_opacities = .TRUE.`. Not required when using the standard dust models, but recommended for user-provided dust models.

`units_csize = ‘pc’,` ! Cell size units (always parsec).

`units_luminosity = ‘W/Hz’,` ! Stellar luminosity units.

`dust_heating_type = ‘eff’` ! Dust heating type. ‘eff’ stays for effective grain.
/

&dartray_input_var ! NAMELIST block containing all numerical variables

`lambda_ref = 0.443` ! Reference wavelength [μm].

`kp_sca_max=1,` ! Maximum angular resolution parameter determining the maximum number of sampling directions for the scattering source function.

`rad_lim=0.1,` ! Limit radius to be used in the pre-calculation phase in units of `modelsize`.

`accuracy=0.3,` ! Accuracy parameter. To hasten the sample calculation, we set a relatively low accuracy, only 30%

`conv_en_lim=0.01,` ! Luminosity threshold parameter determining when the scattering iterations are to be stopped. The current values mean that the iterations will stop when less than 1% of the total scattered luminosity is to be processed in the next scattering iteration.

`bm_par=5,` ! Minimum number of rays per intersected cell during the precalculation and direct light processing for the stellar emission.

`bm_par_sca=5,` ! Same as `bm_par`, but for the scattering iterations and for all phases for the dust emission RT.

`bm_par_max=200,` ! Maximum number of rays per intersected cell.

`max_lambda_stars = 5,` ! Maximum wavelength used in the stellar emission RT [μm].

`min_lambda_dust = 1.,` ! Minimum wavelength used in the dust emission RT [μm].

`dist_obs = 1E6` ! Distance of the external observers [pc].

`npixel_maps = 300` ! Number of pixels per side for the external observer surface brightness maps.

`map_size_factor = 1.8` ! Physical linear size of the external observer surface brightness maps in units of `modelsize`.

`z_wall_coord = 0.3, 0.7` ! Position of the walls perpendicular to the Z axis in units of `modelsize`.
/

&dartray_input_logical ! NAMELIST block containing all logical variables.

`use_lambda_grid = .TRUE.,` ! TRUE when lambda grids has to be used.

`use_dir_out = .TRUE.,` ! TRUE when external observer directions have to be read.

`restore_file_mpi = .TRUE.,` ! TRUE when intermediate files have to be restored if existing.

`print_sed = .TRUE.` ! TRUE if SEDs have to be calculated and saved.

`print_maps = .TRUE.` ! TRUE if surface brightness maps for the external observers have to be calculated and saved.

`print_scaspe_tot = .TRUE.` ! TRUE if the scattering source function has to be saved. IMPORTANT: It is not recommended to set this variable to TRUE, unless the 'i_obs' RT algorithm has to be used later. Saving the scattering source function may require a lot of disk space.

`test_run = .FALSE.` ! TRUE if the code has to run in TEST mode.

`input_av_opacities = .TRUE.,` ! If set TRUE, then `file_av_opacities` is read and compared with integrated/average opacity factors calculated by DART-Ray from the opacity coefficient of single grains.

`no_communications = .TRUE.` ! TRUE if the code has to run in 'no communication' mode when more than one node is used.

`no_dust_rt = .TRUE.` ! TRUE if the dust emission RT calculation does not have to be performed.

`z_wall_on = .TRUE.` ! TRUE when walls perpendicular to the Z axis have to be considered.
/

TEST run

Before running the actual RT calculations, it is worth conducting a TEST run, in which the code perform all the steps, except for the long ray-tracing routines. In this way, you ensure that everything is working correctly, including the allocation of large arrays into memory and the writing of the output files. This is particularly useful for long calculations, as it reveals technical problems that may be encountered in the actual RT calculations.

To run the code in TEST mode, you have to simply set:

```
test_run = .TRUE.
```

in the input file. Then run the code as you would normally, using the command above. You should be able to complete the run rather quickly and find the output files in the directory `./RUNS/GALAXY_EX1`. If you encounter problems that interrupt the calculation, you can determine the nature of the problem from the error message. Possible errors include a limit to the amount of RAM memory that can be allocated by a single process, insufficient disk space for printing the output files, and incorrectly set writing permissions. Once you manage to run the code successfully in the TEST mode, you can delete the output files and move on to the real RT calculations. **Do not forget to reset `test_run = .FALSE.` before continuing!**

Projection mode

Another check before the RT calculation, is to produce maps of the stellar emission without dust and of the dust optical depth. This is useful for checking that the stellar and dust distribution in the 3D grid are in line with expectations. It can be performed using the projection mode. To use this mode, set:

```
rt_algorithm='projection'
```

and insert the following line within the NAMELIST block **&dartray_input_strings**:

```
param_to_project = 'stellar_emission'
```

for making maps of the stellar emission, or:

```
param_to_project = 'optical_depth'
```

for making maps of the optical depth. By running the code in this mode, which should not take long, you will obtain the 'maps' output files. You can then open these files with a program of your choice (e.g. in IDL or python) to visualise the maps, convert them to FITS files, or perform an analysis of them.

Standard RT run

To run the standard RT calculations, you can simply leave the input file as above with:

```
rt_algorithm='2D'.
```

This will make the code use the 2D RT algorithm, which can be used for axisymmetric models and it is up to a factor 8 times faster than the 'main' RT algorithm. The downside is that the observer maps cannot be calculated directly within the run; they have to be derived later using the 'i_obs' algorithm. Note that in the input file above we set:

```
no_dust_rt = .TRUE.,
```

which means that the dust emission RT calculation will not be performed when the stellar emission RT has been completed. You can set this parameter to .FALSE., so that all the RT calculations will be done in one go, if desired. However, in general, RT runs take considerable time and there is always the danger of system failures or time limitations, but you could simply run the dust emission RT calculation afterwards setting:

```
rt_algorithm='dust2D'.
```

Note that you can change the type of dust heating/emission using `dust_heating_type`. Also, note that we set:

```
print_scaspe_tot = .TRUE..
```

This is, in general, not recommended because the scattering source function can consume a lot of disk space. However, for the 2D algorithm, writing these files is necessary because the scattering source function is needed later, when we will use the 'i_obs' algorithm to calculate the surface brightness maps.

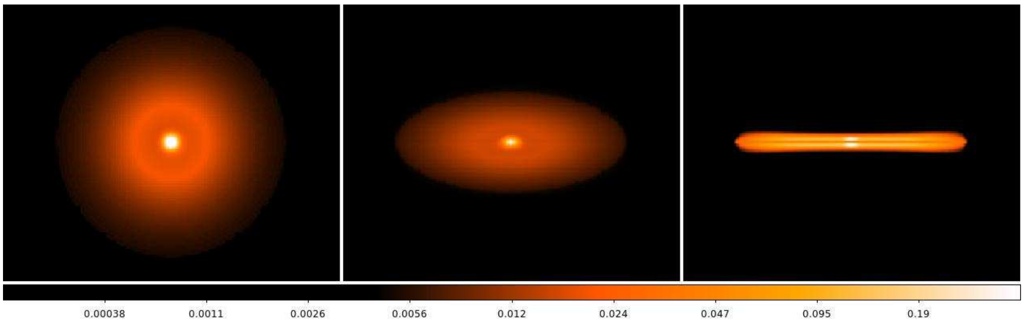

Figure 6.1: 4430 Å images of the example model of the axisymmetric galaxy at face-on (left), intermediate (middle) and edge-on (right) inclinations.

I_OBS mode

The ‘i_obs’ mode is used to calculate the radiation specific intensity ‘i_obs’ files and the surface brightness maps at arbitrary lines-of-sight (for the external observers) and 3D positions (for the internal observers), provided that the scattering source function files have been saved on disk at the end of a standard RT run. To use this mode, you have to set:

```
rt_algorithm=‘i_obs’
```

for the stellar emission RT or:

```
rt_algorithm=‘i_obs_dust’
```

for the dust emission RT. This RT algorithm should work relatively fast, unless you have many observers. At the end of the RT run, you will find in the output directory the files ‘i_obs’, ‘sed’ and ‘maps’ with the additional label ‘_int’ (to distinguish them from those created by standard ‘main’ RT algorithm). After reading the ‘maps’ files with a program of your choice (able to read files in HDF5 format), you can derive images like the ones shown in figure 6.1.

Note that after this calculation has been completed, the scattering source function ‘scaspe_tot’ files can be deleted if they are not needed for other reasons.

6.2 N-body/SPH galaxy simulation

Here we show how to import the output of an N-Body/SPH simulation, construct the corresponding 3D grid, and run the RT calculations with DART-Ray. The galaxy simulation has been provided by Victor Debattista (see Portaluri et al. 2017).

6.2.1 Grid creation

The example N-body/SPH simulation files, output from the Gasoline code, can be found on the DART-Ray website as a compressed tar.gz archive. You can download it and uncompress it in the directory `NBODY_SPH_GRIDS`. After that, you will find the following files in that directory:

- `run708mainDiff.param`: Simulation input file;
- `run708mainDiff.01000`: Particle positions, mass, gas temperature, stellar ages;
- `run708mainDiff.01000.OxMassFrac`: Oxygen mass fraction for the particles;
- `run708mainDiff.01000.FeMassFrac`: Iron mass fraction for the particles.

All files except for the simulation input file are in Tipsy format. To create an HDF5 table with the same particle distribution and properties that can be read by DART-Ray, one can use the python script tipsy2dartray.py included in the DART-Ray package. One just needs to run the script and provide the location and name of the N-body/SPH simulation:

```
   python ./tipsy2dartray.py
Please input the name of your simulation file:  './NBODY_SPH_GRIDS/run708mainDiff.01000'.
```

The program creates a file output.hdf5 which you can rename and move to the grid directory:

```
mv output.hdf5 ./NBODY_SPH_GRIDS/EX1/run708mainDiff.01000_unbar.h5.
```

The file run708mainDiff.01000_unbar.h5 will be input through the variable file_nbody_sph to the grid creation program (see section 4.2.2) and contains all the quantities needed to translate stellar particles into stellar luminosity and gas particles into dust opacity distribution. If your simulation files are not in Tipsy format, you can still import the simulation to DART-Ray by constructing an HDF5 file including all the arrays as those required for file_nbody_sph and input that in the grid creation program.

Then you can create the main 3D grid using:

```
./create_adap_grid_Nbody_SPH ./NBODY_SPH_GRIDS/input_grid_nbody_example1.in.
```

When the main grid creation has completed, retype the above command to calculate the lambda grids. Afterwards you will find all the grid files in the directory ./NBODY_SPH_GRIDS/EX1.

In the following we explain the content of the input file input_grid_nbody_example1.in:

&nbody_sph_input_strings

label_model_lambda_grid='run708mainDiff.01000_LD', ! Label contain within the lambda grid files.

grid_file='grid_run708mainDiff.01000_LD_main.h5', ! Main grid file name.

grid_info_file='grid_run708mainDiff.01000_LD_info.dat', ! Info file name.

file_nbody_sph = 'run708mainDiff.01000_unbar.h5', ! HDF5 file containing the distribution and properties of particles derived by the NBody/SPH simulation.

dir_grid='./NBODY_SPH_GRIDS/EX1', ! Directory where the grids are to be written.

file_lambda_list = 'lambda_list_trustI_basic.dat', ! File containing the wavelength list.

units_lambda = 'um', ! Wavelength units (always microns).

dust_model = 'TRUST', ! Assumed dust model. 'TRUST' means dust model of the TRUST dust benchmark project.

file_av_opacities = './DUST_OPACITY/TRUST/ZDA_BARE_GR_S_Effective.dat' ! File containing the integrated/averaged opacities and scattering anisotropy coefficients derived by the user. It is used only if input_av_opacities = .TRUE.

stellar_library = 'starburst99' ! Stellar emission library assumed to convert stellar particle mass into stellar luminosity. 'starburst99' means standard single age stellar population spectra from STARBURST99 (assuming Kroupa IMF).

`subdivision_criteria = ‘standard’` ! Subdivision criteria. ‘standard’ criteria subdivides cells if they exceed a threshold luminosity or optical depth.
/

&nbody_sph_input_var

`modelsize = 20000` ! Linear size of the RT model [pc].

`lambda_ref= 0.43287613` ! Reference wavelength used to create the main grid [μm].

`lambda_min = 0.09` ! Minimum wavelength when calculating the lambda grids [μm].

`lambda_max = 30.` ! Maximum wavelength when calculating the lambda grids [μm].

`base=3,3,` ! Subdivision factors.

`max_ncell=4.E6` ! Maximum allowed number of cells in the grid.

`max_lvl=5` ! Maximum allowed number of subdivision levels.

`min_lvl=1.` ! Minimum number of subdivisions.

`max_dtau= 0.01` ! Parameter used in the subdivision criteria. Cells are subdivided if their optical depth is higher than `max_dtau`*average optical depth of the entire model.

`max_dlum=0.10E-4` ! Parameter used in the subdivision criteria. Cells are subdivided if their total luminosity is higher than `max_dlum`* total RT model luminosity.
/

&nbody_sph_input_logical

`input_av_opacities = .TRUE.,` ! If set TRUE, then `file_av_opacities` is read and compared with integrated/average opacity factors calculated by DART-Ray from the opacity coefficient of single grains.
/

6.2.2 RT runs

Once the grids have been created, you can launch the RT run using the following command:

```
./dartray_Nbody_SPH ./NBODY_SPH_GRIDS/input_nbody_example1.in.
```

The input file `input_nbody_example1.in` is prepared to start an RT calculation for the stellar emission using the ‘main’ algorithm. We explain its content line by line below. By making minor changes to this file, one can run different types of RT algorithms as explained in the subsections below.

&dartray_input_strings ! NAMELIST block containing all character variables

`label_model_lambda_grid='run708mainDiff.01000_unbar_LD',` ! Label contained within the lambda grid files.

`label_model_out='run708mainDiff.01000_unbar_LD',` ! Label contained in the output files.

`grid_file=‘grid_run708mainDiff.01000_unbar_LD_main.h5’,` ! Main grid file name.

`file_dir_out=‘dir_out_galaxy.dat’,` ! File containing the lines-of-sight directions for the external observer.

`file_lambda_list=‘lambda_list_trustI_basic.dat’,` ! Wavelength list file.

`dir_runs=‘./RUNS/NBODY_SPH_EX1/’,` ! Directory where the output files have to be saved. If it doesn’t exist, it is created automatically.

`dir_grid=’./NBODY_SPH_GRIDS/EX1/’,` ! Directory where the grids are located.

`rt_algorithm=‘main’` ! RT algorithm to be used. ‘main’ is the standard RT algorithm.

`units_lambda = ‘um’,` ! Wavelength units (always microns).

`dust_model = ‘TRUST’,` ! Assumed dust model. ‘TRUST‘ is the dust model used in the TRUST benchmark project.

`file_av_opacities = ‘./DUST_OPACITY/TRUST/ZDA_BARE_GR_S_Effective.dat’,` ! File containing the integrated/average opacity coefficient calculated indipendently by the user. To read this file, it is required that `input av opacities = .TRUE.`. Not required when using the standard dust models, but recommended for user-provided dust models.

`units_csize = ‘pc’,` ! Cell size units (always parsec).

`units_luminosity = ‘erg/s/Hz’,` ! Stellar luminosity units.

`dust_heating_type = ‘sto_lib’` ! Dust heating type. ‘sto_lib‘ stays for stochastic heating using SED adaptive library approach. /

&dartray_input_var ! NAMELIST block containing all numerical variables

`lambda_ref = 0.10000000` ! Reference wavelength [μm].

`kp_sca_max=1,` ! Maximum angular resolution parameter determining the maximum number of sampling directions for the scattering source function.

`rad_lim=0.10,` ! Limit radius to be used in the pre-calculation phase in units of modelsize.

`accuracy=0.20,` ! Accuracy parameter. To make the sample calculation faster, we set a quite low accuracy: only 20%.

`conv_en_lim=0.01,` ! Luminosity threshold parameter determining when the scattering iterations have to be stopped. The current values means that the iterations will be stopped when only less than 1% of the total scattered luminosity would be processed in the next scattering iteration.

`bm_par=5,` ! Minimum number of rays per intersected cell during the precalculation and direct light processing for the stellar emission.

`bm_par_sca=5,` ! Same as bm par but for the scattering iterations and for all phases for the dust emission RT.

`bm_par_max=200,` ! Maximum number of rays per intersected cell.

`max_lambda_stars = 30,` ! Maximum wavelength used in the stellar emission RT [μm].

`min_lambda_dust = 3.,` ! Minimum wavelength used in the dust emission RT [μm].

`dist_obs = 10E3` ! Distance of the external observers [pc].

`npixel_maps = 400` ! Number of pixels per side for the external observer surface brightness maps.

`map_size_factor = 1.` ! Physical linear size of the external observer surface brightness maps in units of `modelsize`.

`ind_out_maps = 2, 7, 11, 21, 26, 33, 43` ! Wavelength indices for the surface brightness maps that have to be calculated and output. Note that the first element of the wavelength list has index 0 not 1. Given the wavelength list in `lambda_list_trustI_basic.dat`, the indices correspond to the following wavelengths: 0.152, 0.433, 1., 8.111, 23.10, 100., 811. μm.
/

&dartray_input_logical ! NAMELIST block containing all logical variables.

`use_lambda_grid = .TRUE.,` ! TRUE when lambda grids has to be used.

`use_dir_out = .TRUE.,` ! TRUE when external observer directions have to be read.

`restore_file_mpi = .TRUE.,` ! TRUE when intermediate files have to be restored if existing.

`print_maps = .TRUE.` ! TRUE if external observer maps have to be calculated and saved.

`print_sed = .TRUE.` ! TRUE if SEDs have to be calculated and saved.

`input_av_opacities = .TRUE.,` ! If set TRUE, then `file_av_opacities` is read and compared with integrated/average opacity factors calculated by DART-Ray from the opacity coefficient of single grains.

`no_communications = .TRUE.` ! TRUE if the code has to run in 'no communication' mode when more than one node is used.

`no_dust_rt = .FALSE.` ! TRUE if the dust emission RT calculation does not have to be performed.

`test_run = .FALSE.` ! TRUE when walls perpendicular to the Z axis have to be considered.
/

Standard RT run

If you run the code using the above command, without editing the input file, you will find in the output folder './RUNS/NBODY_SPH_EX1' the output files 'ufield', 'i_obs' , 'sed' and 'maps' for both the stellar emission and dust emission calculations. You can open the 'maps' files with a program of your choice, and extract the surface brightness maps calculated for the lines of sight directions defined in 'dir_out_galaxy.dat' and the wavelengths selected using the variable `ind_out_maps`. In figure 6.2, we show the maps for the stellar emission at 0.433μm.

SED run

You may want to recalculate the SEDs, or the surface brightness maps, using different parameters specifying their properties. You can do that by using the 'sed' RT algorithm. Just modify

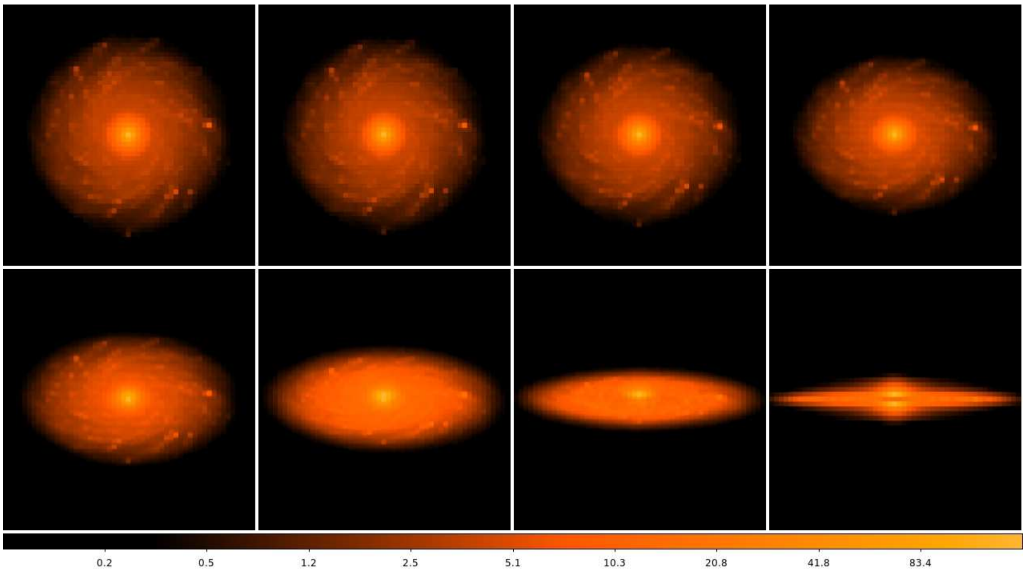

Figure 6.2: Images of the N-body/SPH simulation example at 0.433μm and a set of inclinations from face-on to edge-on. See also high-resolution model images on the DART-Ray webpage.

the value for `rt_algorithm` in the input file `input_nbody_example1.in` as:

```
rt_algorithm = ‘sed’.
```

Then, in the same file, modify the value of variables such as `dist_obs`, `npixel_maps`, `map_size_factor` or `ind_out_maps` as you wish. Then run the code using the same command as before and check the differences in the results!

6.3 TRUST slab benchmark

In the folder ‘./TRUSTI[_GRIDS’, you can find the input files for the grid creation and the RT run for a low resolution version of the TRUST slab benchmark model with $\tau(1\mu m) = 0.01$ (see Gordon et al. 2017). The grid creation can be performed using the following command line:

```
./create_adap_grid_trustI ./TRUSTI_GRIDS/input_grid_trustI_tau0.01_LD.in.
```

This model does not use lambda grids. So, you have to use the above command only once. After that, you can run the RT calculation by using:

```
./dartray_trustI ./TRUST_GRIDS/input_trustI_tau0.01_LD.in
```

The output files will be saved in ‘./RUNS/TRUSTI/tau0.01_LD’. See inside the input files for all the parameter values and options.

6.4 Dust shell surrounding a star

Example files to create a low resolution model for a dust shell can be found in ‘./MAGTAR_GRIDS/’. To create the main grid, just type the command line:

```
./create_adap_grid_magtar ./MAGTAR_GRIDS/input_grid_shell.in.
```

Lambda grids are not created for this model. To run the RT calculations, use the command:

```
./dartray_magtar ./MAGTAR_GRIDS/input_shell.in.
```

All parameter and options used in this example can be found within the input files.

6.5 2D to 3D emissivity distribution

Example files to create a 3D grid of volume emissivity from a 2D distribution can be found in './2DTO3D_GRIDS/'. To create the main grid, use the command line:

```
./create_adap_grid_2dto3d ./2DTO3D_GRIDS/input_grid_2dto3d_example.in.
```

Run again the above command to create the lambda grids. The input 2D grids in ASCII format used in this example can be found in the directory './2DTO3D_GRIDS/EX1'. These correspond to the thick dust disk emission distribution found by Popescu et al. 2017. Note that to use this program you have to place the values of R and z exactly in the same ascending order as in the example files. An error is output if you do not follow the right ordering.

The RT calculations for this kind of models can be done using the galaxy DART-Ray program. To run the example calculation, type:

```
./dartray_galaxy ./2DTO3D_GRIDS/input_2dto3d_example.in.
```

This example simply produces the surface brightness maps for an internal observer located at the sun position.

Chapter 7

How to modify DART-Ray for your own RT problem

If you would like to use DART-Ray to solve the RT problem for geometries which are different from those of the four DART-Ray programs included in the public release, you can write your own subroutines specifying the required dust and stellar distributions. In this chapter, we explain how to do this. A basic knowledge of FORTRAN90, OpenMP and makefile is required to be able to modify the code successfully.

7.1 Grid creation program

To create the main and lambda grids for your particular dust/star geometry, you have to write your own `create_adap_grid_model.f90` and `user_routines_model_.f90` files. In order to do that, you can use the templates included in the directory `PROGRAM_TEMPLATES`. The files in this directory show the general structure of the four standard DART-Ray programs, and contain comments showing you where to modify/add subroutines.

Specifically you should:

- add all the variables, corresponding to specific model parameters, within the variable declaration part at the top of the `user_routines_model` module;
- add the model variables to be read from the `input_grid_model.in` file, within the NAMELIST block declarations within `user_routines_model`;
- if you have added extra NAMELIST blocks, modify the subroutine `read_input_model` within `user_routines_model` to take this into account;
- modify the subroutine `set_model_input` within `user_routines_model`. This subroutine should call all the subroutines needed to derive all the quantities required to calculate the dust opacity coefficient $k_\lambda \rho_d$ and the stellar luminosity density j_λ at each position and for a single wavelength `lambda`. Note that this subroutine is called at two points within `create_adap_grid_model`: during the main grid creation and within the wavelength loop to calculate the lambda grids. You should ensure that the wavelength `lambda` for which the values have to be derived is the correct one in all cases;
- modify the subroutine `calc_dens`, which returns the average dust opacity coefficient and stellar luminosity density given a certain grid cell position and size. The values of these quantities will be stored in the arrays `dens` for the dust opacity and `dens_stars` for the stellar luminosity. Make sure you set the correct line in all places where the `calc_dens` is called;
- if you want to implement particular criteria for the cell subdivision, modify the function `subdivision`;

- add the model parameters to the list of parameters printed on the info file by the subroutine `print_info_file`;

Once you have prepared the above two files, copy them to the main directory and modify `makefile` accordingly to compile them. To distinguish your particular model from all others, we suggest that you assign a label to "model" which is appropriate to your RT model geometry and which is used for the file names and by all subroutines containing this label.

7.2 DART-Ray program

After having prepared the grid creation program, you just need to prepare the corresponding `dartray_model` program. To do this, you only need to make minor modifications to the `dartray_model.f90` file, which you can find in the directory `PROGRAM_TEMPLATES`.

Firstly, you should rename the label "model" to the one you chose for the grid creation program and user routine module. Remember to rename "model" both in the file name and where it is required within the file itself.

Then, if your model does not use point sources, you should simply comment the line calling the subroutine `set_model`. Instead, in the opposite case, you should add a subroutine `set_model` to the `user_routines_model` module. This subroutine simply assigns the luminosity to the point sources, which is stored in the array `lum_p_src_arr`. See the subroutines `set_trustI` and `set_magtar` in the TRUST and magnetar user routine modules for examples on how to do that.

That's it! At this point you should just copy this file to the main directory and modify makefile to compile the new `dartray_model` program. Before running your RT calculations, it is strongly advised to use the 'projection' algorithm to check that the stellar and dust opacity distributions are the ones you expect.

References

[1] B. T. Draine and A. Li. “Infrared Emission from Interstellar Dust. IV. The Silicate-Graphite-PAH Model in the Post-Spitzer Era”. In: ApJ 657 (Mar. 2007), pp. 810–837. DOI: 10.1086/511055. eprint: astro-ph/0608003.

[2] K. D. Gordon et al. “TRUST. I. A 3D externally illuminated slab benchmark for dust radiative transfer”. In: A&A 603, A114 (July 2017), A114. DOI: 10.1051/0004-6361/201629976. arXiv: 1704.06584.

[3] P. Guhathakurta and B. T. Draine. “Temperature fluctuations in interstellar grains. I - Computational method and sublimation of small grains”. In: ApJ 345 (Oct. 1989), pp. 230–244. DOI: 10.1086/167899.

[4] L. G. Henyey and J. L. Greenstein. “Diffuse radiation in the Galaxy”. In: ApJ 93 (Jan. 1941), pp. 70–83. DOI: 10.1086/144246.

[5] A. Laor and B. T. Draine. “Spectroscopic constraints on the properties of dust in active galactic nuclei”. In: ApJ 402 (Jan. 1993), pp. 441–468. DOI: 10.1086/172149.

[6] G. Natale et al. “Dust Radiative Transfer Modeling of the Infrared Ring around the Magnetar SGR 1900+14”. In: ApJ 837, 9 (Mar. 2017), p. 9. DOI: 10.3847/1538-4357/aa5c82. arXiv: 1701.07442 [astro-ph.HE].

[7] G. Natale et al. “Predicting the stellar and non-equilibrium dust emission spectra of high-resolution simulated galaxies with DART-RAY”. In: MNRAS 449 (May 2015), pp. 243–267. DOI: 10.1093/mnras/stv286. arXiv: 1502.03315.

[8] C. C. Popescu et al. “A radiation transfer model for the Milky Way: I. Radiation fields and application to high-energy astrophysics”. In: MNRAS 470 (Sept. 2017), pp. 2539–2558. DOI: 10.1093/mnras/stx1282. arXiv: 1705.06652.

[9] E. Portaluri et al. “The kinematics of σ-drop bulges from spectral synthesis modelling of a hydrodynamical simulation”. In: MNRAS 467 (May 2017), pp. 1008–1015. DOI: 10.1093/mnras/stx172. arXiv: 1702.02162.

[10] J. Steinacker, M. Baes, and K. D. Gordon. “Three-Dimensional Dust Radiative Transfer*”. In: ARA&A 51 (Aug. 2013), pp. 63–104. DOI: 10.1146/annurev-astro-082812-141042. arXiv: 1303.4998 [astro-ph.IM].

[11] G. M. Voit. “X-ray irradiation of interstellar grains in active galaxies - Evaporation and infrared spectra”. In: ApJ 379 (Sept. 1991), pp. 122–140. DOI: 10.1086/170490.

[12] V. Zubko, E. Dwek, and R. G. Arendt. “Interstellar Dust Models Consistent with Extinction, Emission, and Abundance Constraints”. In: ApJS 152 (June 2004), pp. 211–249. DOI: 10.1086/382351. eprint: astro-ph/0312641.