Chapter 6

Radiative Transfer

> Nel mezzo del cammin di nostra vita, mi ritrovai per una selva oscura, ché la diritta via era smarrita.
> When halfway through our life's journey, I found myself in a gloomy wood, because the path which led straight had been lost.
>
> Dante Alighieri

Our human perception of the universe is dominated by electromagnetic radiation. Radiation forms the basis for essentially all astronomical research, including all of the great images reported by observational astronomers. Radiative transport is the foundation for understanding how radiation interacts with matter. Solving the differential equations describing the interaction between radiation and matter is no trivial task. In AMUSE, radiative transfer is addressed numerically using two quite distinct techniques: by photon propagation through a grid, and by Monte Carlo ray-tracing. These approaches are complementary. In AMUSE, these codes are typically used to determine the temperature and ionization state of the gas. Radiative transfer is expensive in terms of computer time, particularly if the time scales are governed by the hydrodynamics or the gravitational dynamics of the environment.

6.1 In a Nutshell

Radiative transfer deals with the way electromagnetic radiation travels through space and interacts with matter. Radiation is mediated by photons, which have energy, direction, and polarization state. Because the number of photons is generally very large, we normally work with statistical quantities, such as mean intensity and flux, that describe the bulk properties of the radiation field. These quantities may be frequency-dependent, or integrated over some portion of the electromagnetic spectrum. Radiation travels freely through vacuum without loss of energy, but when a ray interacts with matter, energy can be injected or removed.

Radiative flux. The radiative flux F is defined as the total amount of energy passing through a unit of area of some surface per unit of time. Flux is measured in units of erg s^{-1} cm^{-2}, or the SI equivalent. If (as is usually the case) the energy is distributed

over a range of frequencies, it is convenient to work in terms of $F\nu$ (unit erg s^{-1} cm^{-2} Hz^{-1}), where $F\nu d\nu$ is the energy flux in the frequency range $[\nu, \nu + d\nu)$. In that case, the total (bolometric) flux is

$$F = \int_0^\infty F_\nu \, d\nu. \tag{6.1}$$

Radiative intensity. In general, a radiation field will consist of photons moving in all directions at any point in space. The specific intensity I is the radiative flux per steradian,

$$I \equiv \frac{F}{\Delta\Omega}, \tag{6.2}$$

where $\Delta\Omega$ is solid angle around some direction $\hat{\boldsymbol{s}}$. Here, we adopt the convention that the unit vector $\hat{\boldsymbol{s}}$ represents a direction at any point in space in which radiation moves. Later, we will use the scalar s to describe distance along the direction (ray) $\hat{\boldsymbol{s}}$. For a frequency-dependent field, the radiative intensity is $I_\nu(\boldsymbol{x}, \hat{\boldsymbol{s}})$ (units erg s^{-1} cm^{-2} Hz^{-1} ster^{-1}).

Now imagine an element of area with normal unit vector $\hat{\boldsymbol{n}}$. The radiative flux crossing this area moving outward (i.e., in the direction $\hat{\boldsymbol{n}}$) is

$$F_\nu = \int I_\nu \, \cos\theta \, d\Omega, \tag{6.3}$$

where $\cos\theta = \hat{\boldsymbol{n}} \cdot \hat{\boldsymbol{s}} > 0$. For a locally isotropic field, $F_\nu = \pi I_\nu$. In vacuum, the intensity I_ν is constant. For radiation from a point source, the flux at distance r from a source scales as $1/r^2$. The constancy of the intensity along the path in vacuum plays a fundamental role in radiative transport and forms the basis for the radiative transfer equation.

Blackbody radiation is the radiation field emitted by a perfect emitter at temperature T. The intensity of radiation from that surface is

$$I_\nu \;=\; B_\nu(T) \;=\; \frac{2h\nu^3}{c^2}\left(e^{h\nu/kT} - 1\right)^{-1}, \tag{6.4}$$

independent of emission angle. Inverting this equation allows us to determine the temperature of a blackbody emitter by measuring the intensity I_ν in some direction. In practice, an emitting object is not a perfect blackbody, but has some thermal emissivity ε_ν. The thermal flux emitted from its surface (Equation 6.3) then is

$$F_\nu = \varepsilon_\nu \pi B_\nu(T). \tag{6.5}$$

6.1.1 Underlying Equations

In vacuum, the intensity I_ν is constant, but when radiation interacts with matter, I_ν changes. Matter along a ray may absorb radiation, decreasing I_ν, or emit energy, increasing I_ν. In addition, radiation can be scattered by the medium—changed in direction but not in energy. The resulting *transfer equation* describes these processes.

Many excellent books have been written on this subject (Chandrasekhar, 1960; Peraiah, 2001), often developing the details in specific geometries, such as the spherically symmetric and locally planar atmospheres of stars. Here, we will avoid specific geometries, as we are mostly interested in the general problem, where no simplifying assumptions exist.

6.1.1.1 Absorption

Consider first a beam of radiation passing through a purely absorbing medium. As the beam traverses distance δs, the probability that a photon of frequency ν will be absorbed is $\delta P = \alpha_\nu \delta s$. The absorption coefficient α_ν can be expressed in terms of the density ρ and opacity κ_ν of the medium: $\alpha_\nu = \rho\kappa_\nu$. Thus, the fractional change in intensity of the beam is $\delta I_\nu / I_\nu = -\alpha_\nu \delta s$, so

$$\frac{dI_\nu}{ds} = -\alpha_\nu I_\nu. \tag{6.6}$$

The solution to this equation is

$$I_\nu(s) = I_\nu(0)\, e^{-\tau_\nu(s)}, \tag{6.7}$$

where $\tau_\nu(s) = \int_0^s \alpha_\nu(s')\, ds'$ is the *optical depth* of the medium.

If $\tau_\nu \ll 1$, absorption is negligible and the medium is said to be *optically thin*. If $\tau_\nu \gtrsim 1$, the medium is *optically thick*. In terms of $d\tau_\nu = \alpha_\nu ds$, Equation 6.6 becomes

$$\frac{dI_\nu}{d\tau_\nu} = -I_\nu. \tag{6.8}$$

The absorption coefficient α_ν has dimensions of inverse length. The distance a photon of frequency ν is expected to travel before being absorbed is the mean free path, $l_{\nu,\mathrm{free}} = 1/\alpha_\nu$.

6.1.1.2 Emission

In an optically thin medium, absorption may be ignored and the change in the intensity I along some trajectory (ray) depends only on the emission j_ν (unit $\mathrm{erg\,s^{-1}\,cm^{-3}\,Hz^{-1}\,ster^{-1}}$) by material along that line. For isotropic emission, the intensity changes according to

$$\frac{dI_\nu}{ds} = j_\nu(s) \tag{6.9}$$

If j_ν is independent of I_ν, integrating Equation 6.9 gives

$$I_\nu(s) = I_\nu(0) + \int_0^s j_\nu(s')\, ds'. \tag{6.10}$$

The first term on the right-hand side is the incoming (or background) intensity, which in astronomical situations is often either absent or due to a background star or galaxy. The second term describes the total emission between 0 and s.

For a medium with both absorption and emission, Equations 6.6 and 6.9 combine to give

$$\frac{dI_\nu}{ds} = j_\nu(s) - \alpha_\nu(s) I_\nu, \tag{6.11}$$

which is easily solved:

$$I_\nu(s) = I_\nu(0) e^{-\tau_\nu(0,s)} + \int_0^s j_\nu(s') e^{-\tau_\nu(s',s)}\, ds', \tag{6.12}$$

where $\tau(s_0, s_1) = \int_{s_0}^{s_1} \alpha_\nu ds$ is the optical depth between s_0 and s_1. The first term on the right-hand side of Equation 6.12 is the attenuated input field; the second may be identified as the (partly reabsorbed) radiation emitted by the medium. Often the radiative transfer equation is rewritten in terms of the *source function* $S_\nu \equiv j_\nu/\alpha_\nu$:

$$\frac{dI_\nu}{ds} = \alpha_\nu(s) \left[S_\nu(s) - I_\nu\right]. \tag{6.13}$$

A system is in local thermodynamic equilibrium (LTE) if the local gas temperature equals the local Planck temperature of the radiation field (see Section 4.1.2.1). For a medium in LTE at temperature T, it can be shown that $j_\nu = \alpha_\nu B_\nu(T)$, so $S_\nu = B_\nu[T(s)]$. However, many of the systems we will study are not in LTE. Indeed, the whole idea of performing radiative transfer simulations is to be able to make qualitative and quantitative statements about systems that may be far from LTE. This does not pose a serious limitation to our numerical solution, but it does make it difficult to validate our results from first principles.

6.1.1.3 Scattering

Pure absorption heats the absorbing medium, while pure emission cools it. A medium may also scatter radiation. Scattering may be thought of as absorption followed immediately by re-emission of the absorbed energy: it neither heats nor cools the gas, so it is unimportant for the local temperature structure of the medium. However, it is often critical in determining the details of the radiation field—and hence the appearance of a region.

The removal of photons from a beam due to scattering can be described by a scattering absorption coefficient $\alpha_\nu^{\rm scat} = \kappa_\nu^{\rm scat}\rho$, where $\kappa_\nu^{\rm scat}$ is the scattering opacity. Thus, the total absorption coefficient has contributions from both pure absorption and from scattering:

$$\alpha_\nu = \alpha_\nu^{\rm abs} + \alpha_\nu^{\rm scat}. \tag{6.14}$$

Similarly, the emission j_ν has contributions from pure emission and from scattering:

$$j_\nu = j_\nu^{\rm emis} + j_\nu^{\rm scat}. \tag{6.15}$$

Here $j_\nu^{\rm emis}$ corresponds to the absorption coefficient $\alpha_\nu^{\rm abs}$, so $j_\nu^{\rm emis} = \alpha_\nu^{\rm abs} B_\nu(T)$ in LTE.

The expression for $j_\nu^{\rm scat}$ introduces considerable complexity into the transfer equation, as it couples beams moving in all directions at any point. We denote by $p_\nu(\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}')$ the probability that a photon initially moving in direction $\hat{\boldsymbol{s}}'$ is scattered into direction

$\hat{\boldsymbol{s}}$ (i.e., into the direction of the beam under consideration in the transfer equation), where $\int p_\nu(\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}')\, d\Omega' = 1$. It follows that

$$j_\nu^{\text{scat}} = \int \alpha_\nu^{\text{scat}} I_\nu(\hat{\boldsymbol{s}}')\, p_\nu(\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}')\, d\Omega'. \tag{6.16}$$

Scattering turns a relatively simple ordinary differential equation into an integro-differential equation that is much harder to solve.

The physics of the scattering interaction is contained entirely within the function p_ν. If the medium is composed of particles with radii a much smaller than the wavelength ($\lambda = c/\nu$) of the radiation, the interaction is in the so-called Rayleigh regime, which is well-understood from a theoretical perspective (Bohren & Huffman, 1983). If the particles have radii much larger than the wavelength ($a \gg \lambda$), as is generally the case with dust particles, then the scattering cross section is $\sigma \sim \pi a^2$ and we are in the geometric optics regime. Radiative transfer through a medium of large particles can be complex due to effects at the edges of the particles, or if the particles are partially transparent or (usually) not spherical. Radiative transfer through a medium composed of particles with sizes comparable to the wavelengths of interest ($a \simeq \lambda$) is most challenging, and we refer the interested reader to Chandrasekhar's textbook on the topic (Chandrasekhar, 1960).

6.1.2 Physics of the Integrator

When numerically solving the radiative transfer equation (Equation 6.11), the objective may be to construct an image of the simulation, compute the energy spectrum, or determine the temperature and ionization structure of a gaseous body. These are rather different objectives, and a wide range of numerical methods have been developed to address each.

Radiative transfer is difficult because the emission j_ν and the absorption α_ν are generally not known in advance—the radiation field $I_\nu(\mathbf{x}, \hat{\boldsymbol{s}})$ we wish to compute affects both. Local absorption and emission of energy can change the temperature and ionization structure of the medium, in turn producing large changes in j_ν and/or α_ν. At the same time, all the rays from any direction that pass through location $\mathbf{x}$ contribute via scattering to the transfer equation at $\mathbf{x}$. Even if scattering can be neglected, in a system with multiple sources (such as a star-forming region), conditions at any location depend on the solution to the transfer equation along multiple directions. In other words, radiation from one region of a molecular cloud affects conditions in another part of the cloud, which in turn may affect conditions in the original region. Solving this problem is the major challenge in computational radiative transfer.

Photons travel at the speed of light. For the problems at hand, we assume that the solution to the radiative transfer equation is established instantaneously each time the solver is called. The underlying assumption is that the effective travel time of a photon is much shorter than the time scale for local changes in the scattering medium though which the photons travel. We do not expect that this assumption will pose a problem or significantly affect simulation results in any negative way (meaning that the consequences of the assumption are likely smaller than the uncertainties in the numerical scheme).

However, we know that the photon travel time can be long, in some cases, compared to the material response time. In such cases, delay times may be important. For example, it may be the case that the matter needs some characteristic time to respond by cooling down or heating up. In material with large optical depth, radiation may undergo many absorptions and re-emissions and the accumulation of short response times can become important. Although probably not very relevant in astrophysics, this aspect of radiative transfer has been included in the study of photonic crystals (Baba, 2008).

6.2 Radiative Transfer in AMUSE

6.2.1 Radiative Transfer Modules

Table 6.1 lists the three radiative transfer codes currently available as standard in AMUSE. Note that it does not include the radiative transfer routines built into the `Flash` (Fryxell *et al.*, 2000) hydro module; see Section 5.] The codes listed differ significantly in their description of both the radiation field and the medium with which it interacts, but all employ essentially the same iterative process to solve the transfer equation and determine the intensity field $I_\nu(\mathbf{x})$ at each spatial location.

A starting approximation to $I_\nu(\mathbf{x})$ is typically the solution obtained during the previous invocation of the module. Initially, we might simply set $I_\nu = 0$ everywhere. The emission $j_\nu(\mathbf{x})$ is then calculated at every location, taking into account scattering from the existing field, emission from the gas, and injection of energy from stars and other external sources. Finally, the transfer equation (Equation 6.11) is solved along each photon trajectory to give a new estimate for $I_\nu(\mathbf{x})$. The matter density is held static while the radiation field is updated, but its temperature and ionization structure (and hence its emission and absorption) change at each iteration in response to the changing radiative environment. As illustrated in Figure 6.1, this iterative process stops when changes to I_ν become acceptably small.

We now discuss some specifics of the AMUSE modules listed in Table 6.1. Although the AMUSE philosophy is to hide such details whenever possible, the radiative transfer modules are more complex than gravitational (see Chapter 3) or even hydrodynamical (see Chapter 5) codes, and a little inside knowledge can be useful in deciding which radiative module to use for a given problem.

Data structure. The three modules (see Table 6.1) differ widely in their description of the medium through which the radiation propagates. Mocassin bins the gas onto a Cartesian grid, although there is a compile option to use a Voronoi (1908) tessellated grid (Hubber *et al.*, 2016), while SimpleX uses an unstructured Delaunay-tessellated grid (Delaunay, 1934), with more grid points in regions of higher opacity. The tessellated versions works best with SPH hydro simulations whereas regular Cartesian grid methods would be more suitable in combination with grid-based hydrodynamics codes. SPHray has no grid, but rather (as the name suggests) propagates photons through the interpolated density and temperature (as well as emission and absorption) fields defined

Module	Mocassin (ref. 1)	SimpleX (2)	SPHRay (3)
Language	Fortran 95	C++	Fortran 95
Parallelization	MPI	OPENMP	OPENMP
Data Structure	Cartesian grid	Unstructured Delaunay grid	No grid; SPH particle distribution
Radiation Representation	Monte Carlo energy packets	Photon numbers and directions	Monte Carlo energy packets
Ray Propagation	Absorption computed from transfer equation; (re)emission creates new packets	Photons transported to adjacent cells; absorption computed from transfer equation; (re)emission/scattering adds/redirects photons	Absorption computed from transfer equation; (re)emission creates new packets
Time Dependence	Thermal and ionization equilibrium	Thermal and ionization equilibrium	Time-dependent temperature equation; nonequilibrium solution
Physics	Photoionization, photoelectric heating, and recombination; dust absorption	Photoionization, photoelectric heating, and recombination; dust absorption	Photoionization, photoelectric heating, and recombination

Table 6.1: Radiative transfer codes incorporated into AMUSE. References: (1) Ercolano *et al.* (2003), (2) Altay *et al.* (2008); Paardekooper *et al.* (2010), (3) Ritzerveld & Icke (2006); Altay *et al.* (2008, 2011). In § B.7 we present a more general overview of the various code application domains.

by an SPH hydro simulation.

Radiation representation. Mocassin and SPHray are Monte Carlo methods, using a large number (typically millions) of energy packets to model the radiation field. The packets are created at sources and by emission/scattering events, and are removed by absorption as each packet propagates through the medium. SimpleX keeps track of the numbers of photons in each grid cell moving in the directions of the adjacent cell centers (a variable number, but averaging ∼16 in 3D). Photons created by sources may be absorbed, emitted, or scattered as they move from cell to cell. Because only a limited number of directions is possible, the speed of the code is independent of the number of sources.

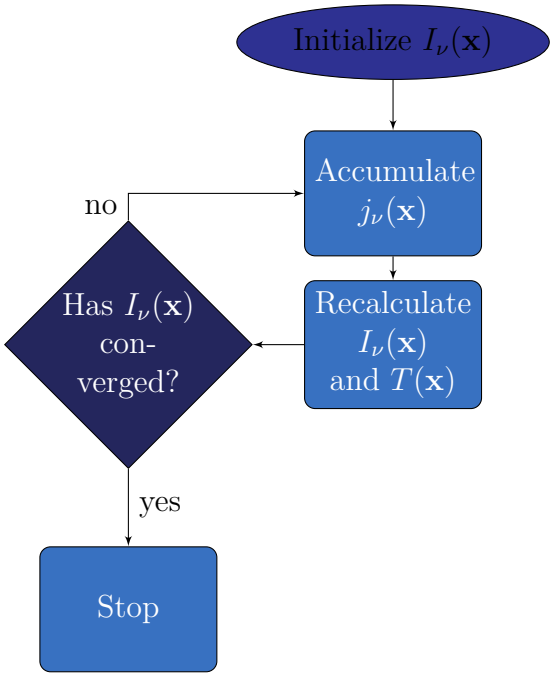

Figure 6.1: Workflow for a radiative transfer code.

Ray propagation. In the Monte Carlo codes Mocassin and SPHray, each energy packet performs a random walk through the medium. Along each portion of a packet's trajectory, the absorption/scattering probability is

$$p(\tau) = e^{-\tau}, \tag{6.17}$$

where τ is optical depth measured along the path (Section 6.1.1.2). Operationally, the optical depth to the next absorption (or scattering) is determined by choosing a random number $\xi \in [0, 1)$ and setting $\tau = -\ln(1 - \xi)$. The directions of emitted or scattered packets are chosen randomly (uniformly for emission or according to $p_\nu(\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}')$ for scattering; see Section 6.1.1.3). The process is repeated until the packet is absorbed or leaves the computational domain. SimpleX uses a similar optical depth formulation to determine the amount of absorption/scattering between one cell center and the next, then modifies the photon distribution at each cell center to model emission and scattering. Because photons cannot move in straight lines on the irregular grid, care is taken to ensure that photons moving through optically thin regions on average maintain the correct direction.

Time dependence. Mocassin and SimpleX determine the physical conditions in the medium by assuming local radiative and ionization equilibrium at each iteration, and therefore return the steady-state solution to the problem after each invocation. SPHray solves an energy-balance equation for the temperature and ionization state of the medium as it computes the radiation field, and thus it generally returns an instantaneous nonequilibrium (time-dependent) solution.

Physics of the medium. All modules contain treatments of photoionization, photoelectric heating, recombination, and other directly related atomic/molecular cooling processes. Dust absorption is treated in all three radiative-transfer codes, although the implementations differ and not all internal possibilities are exposed to the current AMUSE interface. For SPHray, it is possible to set a stellar radiation spectrum, which is realized by means of a table that can be identified via the function

```
rt_code = SPHray()
rt_code.function get_spectra_file('spectra_file_name')
```

Normally, this file refers to a standard file in the AMUSE repository. It generally uses seven frequency channels to resolve a radiative source. Absorption in SPHray is computed from the transfer equation in each cell, after which re-emission creates new source. The transfer equation is solved using a semi-implicit iteration scheme (Iliev *et al.*, 2006). Mocassin allows multiple sources with different spectra and can be used in conjunction with dust absorption, but this functionality is currently not supported by the AMUSE interface.

6.2.2 Ionization of a Molecular Cloud

We can run a radiative transfer code in AMUSE in much the same way as running an *N*-body, stellar evolution, or hydrodynamics code. The procedure is a little more elaborate because radiative transfer requires two ingredients—a gaseous region through which the radiation transfer is to be calculated, and (because our principal application will be star-forming regions or supernova nebulae) a list of ionizing sources. Because the modules used to solve the radiative transfer equations differ so greatly, the user must decide in advance which one to use. Ideally, we would like to be able to switch back and forth between modules as needed, but that is not currently feasible in AMUSE.

The radiative transfer codes in the AMUSE framework share essentially the same basic interface, although they differ slightly in detail (see Table 6.1). Each is well-suited to the problem of an embedded star cluster with radiative feedback. Listing 6.1 presents a simple example of how to start and run a radiative transfer problem in AMUSE. It is not very elaborate, but it demonstrates the basic functionality and calling sequence for running a radiative transfer problem. In this example, we opt for the `SimpleX` code.

We start by creating a gaseous region—in this case, a homogeneous spherical distribution of N neutral gas particles (Listing 6.2). Each gas particle has a specific internal energy of $(9\ \mathrm{km\ s^{-1}})^2$, but no ionizing radiative flux. We must explicitly set the ionizing flux of every particle to 0 photons^{-1}; otherwise they will become sources themselves, like the star at the center. We also set the ionization fraction xion to zero.

```
    ism = new_plummer_gas_model(N, converter)
    ism.flux = 0. | units.s**-1
    ism.xion = 0.0
```

```
75 def main(number_of_particles, Lstar, boxsize, time_end):
76     ism = generate_ism_initial_conditions(number_of_particles, boxsize)
77
78     source = Particle()
79     source.position = (0, 0, 0) | units.parsec
80     source.flux = Lstar/(20. | units.eV)
81     source.rho = ism.rho.max()
82     source.xion = ism.xion.max()
83     source.u = (9. | units.kms)**2
84
85     radiative = Simplex()
86     radiative.parameters.box_size = boxsize
87     radiative.particles.add_particle(source)
88     radiative.particles.add_particles(ism)
89
90     radiative.evolve_model(time_end)
91     print(f"min ionization: {radiative.particles.xion.min()}")
92     print(f"average ionization: {radiative.particles.xion.mean()}")
93     print(f"max ionization: {radiative.particles.xion.max()}")
94     plot_ionization_fraction(radiative.particles.position,
95                              radiative.particles.xion)
96     radiative.stop()
```

Listing 6.1: Minimal routine to determine the global degree of ionization of a Plummer (1911) distribution of hydrogen gas with a scale radius of 3 pc and a mean density of 32 atoms/cm^3. The initial conditions are generated using the code in List. 6.2. Each of the 10^4 particles was irradiated by a central star with luminosity 100 $L_\odot$ for a time interval of 10^6 yr. This snippet was taken from the source code in `${AMUSE_DIR}/examples/textbook/rad_minimal.py`.

```
54 def generate_ism_initial_conditions(number_of_particles, boxsize):
55     converter = nbody_system.nbody_to_si(10 | units.MSun, 3 | units.parsec)
56     ism = new_plummer_gas_model(number_of_particles, converter)
57     ism.flux = 0. | units.s**-1
58     ism.xion = 0.0
59
60     hydro = Fi(converter)
61     hydro.gas_particles.add_particles(ism)
62     hydro.evolve_model(1 | units.hour)
63     hydro.gas_particles.new_channel_to(ism).copy()
64     hydro.stop()
65     ism = ism.select(lambda r: r.length() < 0.5*boxsize, ["position"])
66     print(
67         f"Max density: {ism.rho.max().in_(units.MSun/units.parsec**3)} ",
68         f"{ism.rho.max().in_(units.amu/units.cm**3)}"
69     )
70     return ism
```

Listing 6.2: Routing for generating a gas distribution as initial conditions for the radiative transfer calculation (see List 6.1). This snippet was taken from `${AMUSE_DIR}/examples/textbook/rad_minimal.py`.

In order to allow the radiative transfer code to use the gas as a scattering medium, the local density of the gas particles must be determined. In principle, this parameter could have been determined by the Plummer generating function, but here we use an SPH code (Hernquist & Katz, 1989; Pelupessy *et al.*, 2004; Gerritsen & Icke, 1997) to calculate the density for each SPH particle. In the following snippet, we instantiate

the SPH code Fi and run it for a short while (1 min in this example) before copying the relevant information back to the local interstellar medium particles.

```
    hydro = Fi(converter)
    hydro.gas_particles.add_particles(ism)
    hydro.evolve_model(1|units.minute)
    hydro.gas_particles.new_channel_to(ism).copy()
```

For SimpleX and SPHray, it is important to ensure that there is no scattering nor any sources outside the box, and that there is not too much vacuum inside the box. It can be tricky to choose the right box size, and in a coupled simulation where the sources and the gas particles are moving, this can make it difficult to keep the numerical solver stable. In the ray-tracing radiative-transfer code Mocassin, this is less of an issue. In our small illustration, we need only remove the particles that are outside the box, which we do via a Python Lambda function. (See Appendix F.3.6 for more on this handy Python operation.) [**I think Lambda operations are outdated now? – Steven**]

```
    ism = ism.select(lambda r: r.length() < 0.5*boxsize, ["position"])
    ism = ism[ism.position.length() < 0.5 * boxsize]  # Steven's updated code
```

We now can initialize the radiative source, which is a single particle at the origin of our coordinate system. The radiative transfer code requires the ionizing luminosity of the radiative source to be expressed in photons per second. We make the conversion, assuming that each photon deposits 20 eV, so if the total ionizing luminosity is Lstar (W), then the quantity passed to the radiative transfer code is Lstar/(20. | units.eV), or 1.2×10^{46} photon s^{-1} in this case.

```
    source = Particle()
    source.position = (0, 0, 0) | units.parsec
    source.flux = Lstar/(20. | units.eV)
    source.rho = ism.rho.max()
    source.xion = ism.xion.max()
    source.u = (9. | units.kms)**2
```

Because a source can also be a scattering object, it has attributes for flux, density, ionization state, and internal energy (source.u). The flux level, ionization rate, and local density are determined from the scattering particle. For the density, we adopt the maximum density of the SPH particles, which is close to the assumed density in our uniform sphere. In our example, this is $\sim$0.4 $M_\odot$ pc^{-3}.

At this point, we start the radiative transfer code. Because SimpleX is a Delaunay-tessellated radiative transfer solver, we have to tell it the size of the box within which we want it to solve radiative transfer equations. This will pose some interesting challenges later, when we combine the radiative solver with gravitational and/or hydrodynamics.

```
    rad = SimpleX()
    rad.parameters.box_size = boxsize
```

As noted earlier, the correct operation of the SimpleX code is quite sensitive to the size of the box, and may require some experimentation.

```
    radiative.particles.add_particle(source)
    radiative.particles.add_particles(ism)
```

It is interesting to note that, in contrast to the hydrodynamical solvers discussed in Section 5.2.2, some radiative transfer codes have only one set of particles, whether they are sources or part of the scattering medium (but see Section 6.6.1 for a counterexample).

We are now ready to solve the radiative transfer equation by evolving the model.

```
    rad.evolve_model(t_end)
```

Here, we integrate the radiative transfer to t_end = 1 Myr. This calculation is fast because it is carried out for a relatively small number of scattering particles and a single radiative source. However, we will see that, when multiple sources are introduced and/or the scattering medium is resolved at higher resolution, this operation can quickly become far more computationally intensive than the gravitational dynamics, stellar evolution, and even the hydrodynamics calculations combined. For some problems, the radiation integration time interval may be as short as a few years, which is usually small compared to the time scales on which other processes operate (see Figure 1.6), making radiative transfer one of the most expensive operations in a multiphysics environment.

The left plot in Figure 6.6 presents the result of the simulation at an age of 1 Myr. By that time, the ionization front has progressed outward to a distance of roughly one-third of the gas cloud's radius. A theoretical calculation of the ionization and recombination processes operating in the cloud (Strömgren 1939) indicates that the transition from fully ionized to unionized gas should be sharp and occur at a radius of 1.2 pc (dashed line). The reason for the rather broad spread actually seen is given in the right-hand figure, which shows a slice through the center of the cloud. The circles represent the SPH particles in the simulation; their colors indicate degree of ionization and their radii the smoothing length, which provides a rough indication of the scale of the SimpleX tesselation and hence the resolution of the calculation.

Figure 6.2: Degree of ionization $\xi_{\rm ion}$ due to a $100\,{\rm L}_\odot$ star at the center of a 3 pc spherically symmetric Plummer distribution of 10^4 gas particles. The red curve gives the average degree of ionization, binned in intervals of 0.1 pc. The figure was made using the script in List. 6.1 and took 20 seconds on a laptop, see also `${AMUSE_DIR}/examples/textbook/rad_minimal.py`.

6.2.3 Coupling Radiative Transfer with Hydrodynamics

Combining radiative transport with hydrodynamics is not trivial. In principle, we could adopt a linear coupling strategy between two modules, as discussed in Chapter 7, but a bridge-like approach (see Chapter 8) gives superior results, in terms of both better performance and smaller errors. However, as we have not yet discussed bridge

[**check if this is still true – Steven**], we adopt a simpler but still effective approach similar to the interlaced temporal discretization discussed in Section 7.2.1.

We have defined a small radhydro class to simplify this function, to be incorporated later into a bridge with (for example) a gravity solver. In order for this class to comply with bridge specifications, it must have a few extra functions: the most important is evolve_model(). In addition, to make life easier, we include a method to copy data from the two worker codes to the framework, and one to clean up after we are done. For safety (and convenience), we start by synchronizing all relevant data between the two worker codes and the framework, giving us a self-consistent data set to work with. For additional safety, albeit probably unnecessary in this example, we perform a similar synchronization after the two worker codes have reached their objectives.

Between the two synchronization operations, we run the hydrodynamical simulation for half a time step, followed by the radiative transfer simulation for a full time step, and end with a second half of a time step for the hydrodynamics (see also Section 7.2.1):

```
    while t < (tend-timestep/2):
        self.hydro.evolve_model(t+timestep/2.)
        update_rad_from_hydro_channel.copy()
        self.rad.evolve_model(t+timestep)
        update_hydro_from_rad_channel.copy()
        self.hydro.evolve_model(t+timestep)
        t = self.model_time
```

The rest of the radiation–hydrodynamics class is fairly straightforward. In Section 6.6.1, we present an example calculation using this RadHydro class using the source code listing in ${AMUSE_DIR}/examples/textbook/supernova_irradiation.py.

6.3 Examples

6.3.1 Heating of a Protoplanetary Disk

The abundance of several species of short-lived radionuclides (SLRs), such as ^{26}Al and ^{60}Fe, in the solar system has puzzled astronomers for some time (MacPherson *et al.*, 1995). Key questions include: How did these volatile elements, particularly ^{26}Al, enter the solar system? How did they come to be part of meteorites? and how did chondrules (the metallic grains containing these elements) form before chondrites (the meteorites that contain them)? There seems to be a lot we do not really understand.

One possible explanation involves a nearby supernova when the Sun was still part of its parent star cluster. This is a two-stage process. First, the radiation from a bright supernova illuminates and heats the disk. Second, a few decades later, the hot ionized gas from the explosion plows through it. The idea is that it might be possible for the first burst of radiation to heat the disk enough to vaporize the dust, such that the subsequent wave of radioactive material could deposit the SLRs. Later cooling of the disk might subsequently have caused these SLRs to become part of the newly formed dust. This is all speculative, but worthy of further exploration.

Here, we study the first stage of the process: the heating and ionization of a protoplanetary circumstellar disk by the radiation from the supernova. For the gaseous disk, we adopt the protoplanetary disk model discussed in Section 5.3.2. To mimic the thick protoplanetary disk, we adopt a Safronov–Toomre Q parameter of $Q = 25$, rather than $Q = 2$ as in the earlier example of a thin disk (see Section 5.3.2). The disk was constructed as in the earlier example, with 10^4 equal-mass SPH particles at distances between 1 au and 100 au from the Sun, a total mass of 0.01 $M_\odot$ and a power-law density profile with an exponent of -2. We arbitrarily assume that the disk is oriented at an angle of 15° (almost face-on) to the line of sight of the exploding star. The hydrodynamical simulation was run using `Fi` with an isothermal equation of state and an artificial viscosity parameter of $\alpha = 0.1$ (see Section 5.2.4).

For the evolution of the supernova luminosity, we follow Sanders *et al.* (2015) and adopt a simple broken power-law luminosity function with characteristic luminosity and time scales. The script ${AMUSE_DIR}/examples/textbook/supernova_IIp_Lightcurve.py presents code from Sanders *et al.* (2015) to reproduce those light curves. We adopt two distinct supernovae, SN PS1-11aof, with typical luminosities of a few times $\sim 10^9\,L_\odot$ when the peak is reached about 10 days after the onset of the explosion. In our example, we adopt the best-fit parameters for this type IIp supernova, which has a peak luminosity of 2×10^{43} erg/s, and the other supernova PS1-10a is somewhat dimmer. The light curves for two supernovae are shown as the blue curves in Figure 6.3.

Figure 6.3: Light curves for the type IIp supernovae PS1-10a and PS1-11aof (dashed and solid blue curves, respectively, scale in blue on the left, Sanders *et al.*, 2015). The red curves give the temperature of our model disk at distances of 0.15, 0.3 and 0.4 pc (from left to right). The mean disk temperature (axis scale in red on the right) gives the disk temperature calculated using supernova PS1-10a at $d = 0.15$ pc and $\Theta = 15°$ (the dashed curve) and PS1-11aof at a distance of 0.3 pc with $\Theta = 45°$ and at $d = 0.4$ pc with $\Theta = 15°$ (the solid curves). The thin green line (upper right) indicates a cooling of 0.3 K/hour, which is consistent with cooling after the supernova radiation has heated the disk for SN PS1-11aof, and which is sufficient for the formation of vitreous structures in the chondrules of the circumstellar disk. The calculations are performed at a resolution in the radiative transfer code of `Nray/dt`$= 10^7/$`(1|units.hour)`. The figure was made using the script `${AMUSE_DIR}/examples/textbook/supernova_irradiation.py`. The calculation took several days on a laptop, and for this reason we include the data files and the routine to plot those in `plot_supernova_IIp_and_disk_temperature.py`.

We used the radiative transfer code `SPHRay`, which is well suited to such simulations, because it does not impose any limits on the spatial geometry or the computational domain. The hot supernova radiation was mimicked by setting some additional parameters in the radiative transfer code.

```
1    radiative = SPHRay(redirection="file", number_of_workers=24)
2    radiative.parameters.number_of_rays = Nray/dt
3    radiative.parameters.default_spectral_type = -3.
```

Here, number_of_rays = Nray/dt defines the number of rays per time step used in the radiative transfer solver. In our case, we adopted `Nray/dt`$= 10^7/$`(1| units.hour)`. The parameter `default_spectral_type` is used to indicate the slope of the source's spectral energy distribution. For a supernova, a value of -3 is closer to the actual observed spectral energy distribution than the default value of -1.

In the radiation transport code, the irradiating source is represented by a particle,

called src_particle. We can update this particle's parameters at run time by using a separate channel.

```
    channel_from _src_to_rad = \
            supernova.particles.new_channel_to(radiative.src_particles)
    supernova.evolve_model(time)
    channel_from_src_to_rad.copy_attributes(["luminosity"])
```

Note that there is only one source particle in our calculation, even though the code's denotative is plural.

The calling sequence for the radiative transfer and hydrodynamics modules is similar in structure to Listing 7.2 in Section 7.2.3, where we combined stellar evolution with gravitational dynamics. The two codes are called radiative and hydro. Both codes have appropriate channels defined to and from the local particle sets: disk for the SPH particles, star for the stellar particle in the hydro code, and supernova.particles for the local source particle to the radiative transfer code. The event loop then becomes:

```
    while time<t_end:
        time += dt/2
        supernova.evolve_model(time)
        channel_from_src_to_rad.copy_attributes(["luminosity"])
        radiative.evolve_model(time)
        channel_rad_to_hydro.copy()

        time += dt/2
        hydro.evolve_model(time)
        hydro_to_disk.copy()

        supernova.evolve_model(time)
        channel_from_src_to_rad.copy_attributes(["luminosity"])
        update_source_particle(time)
        radiative.evolve_model(time)
        rad_to_disk.copy()
```

First, we update the supernova luminosity, perform an evolution step with a time step of dt/2 in the radiative transfer code, and copy the irradiated SPH particle data to the hydrodynamics code. Next, we update the time to a full step, evolve the hydrodynamics of the disk, and copy the data back to the framework. Finally, we repeat the first snippet, updating the supernova luminosity and performing another half radiative time step. In the last line, we copy the data from the radiative transfer code back to the local particle set.

The red curves in Figure 6.3 show the temperature evolution of two identical protoplanetary disks around 1 $M_\odot$ stars, at distances of 0.15 pc (left dashed curve for SN PS1-10a), at 0.3 and 0.4 pc (right two solid curves for SN PS1-11aof) from the supernova. The dashed curve was computed using supernova SN PS1-10a, and the solid curves using the considerably brighter SN PS1-11aof. The mean temperature in the disk exceeds 1500 K for each of these supernova and distances, which is high enough to melt agglomerates with the highest melting points. Carrying out any of these simulations with a lower resolution, using less than 10^7 rays per hour, in the

radiative transport calculation would lead to lower temperatures, whereas adopting a larger number of rays does not significantly affect the results, except that the script takes longer to run.

Following the supernova radiation will be the arrival of the shock wave. This is a more hydrodynamical process, which we will discuss further in Section 9.2.

6.3.2 Ionization Front in an H_2 Region

Most radiative transfer codes require a large number of inputs, usually including an ionizing source and an absorbing medium. The former is often a star, and the latter a gaseous region. We addressed stellar evolution quite extensively in Chapter 4 and hydrodynamics of gaseous bodies in Chapter 5. If you skipped these chapters, this might be a good juncture to revisit them before continuing.

```
25 def make_grid(number_of_grid_cells, length, constant_hydrogen_density,
26              inner_radius, outer_radius):
27     grid = Grid.create([number_of_grid_cells] * 3,
28                        length.as_vector_with_length(3))
29     grid.radius = grid.position.lengths()
30     grid.hydrogen_density = constant_hydrogen_density
31     grid.hydrogen_density[grid.radius <= inner_radius] = 0 | units.cm ** -3
32     grid.hydrogen_density[grid.radius >= outer_radius] = 0 | units.cm ** -3
33     return grid
```

Listing 6.3: Initialization of a grid with a uniform-density spherical cloud.

In this example, we use stellar evolution and hydrodynamics together with radiative transfer to calculate the radial electron temperature structure of an initially homogeneous H_2 region with density 100 cm^{-3}, which is illuminated by a 120 $M_\odot$ (initial mass) star at an age of 3.3 Myr. At that age, such a star has, according to the SeBa stellar evolution code (Portegies Zwart & Verbunt, 1996, 2012), a mass of 56.8 $M_\odot$ a luminosity of about $1.5 \times 10^6 L_\odot$, and a temperature of 50,000 K. The stellar evolution was carried out using the simple script given in Section 4.3. After the star is located at the cloud center, we initialize the gas distribution in a grid as in Listing 6.3. We then set up the radiative transfer module; Listing 6.4 presents a snippet of code to do this using `Mocassin`. Of course, these initial conditions are a bit contrived. One might wonder how a 3.3 Myr old star of 120 $M_\odot$ could arrive in the middle of a cloud of molecular hydrogen, but anything is possible in the computer.

```
83 def initiate_radiative_transfer_code(outer_radius, Ngrid):
84     radiative_transfer = Mocassin(number_of_workers=4)
85
86     radiative_transfer.set_input_directory(
87         radiative_transfer.get_default_input_directory())
88     radiative_transfer.set_mocassin_output_directory(
89         radiative_transfer.output_directory + os.sep)
90     radiative_transfer.initialize_code()
91     radiative_transfer.set_symmetricXYZ(True)
92
93     setup_grid(radiative_transfer, outer_radius, Ngrid)
```

```
    setup_abundancies(radiative_transfer)

    radiative_transfer.parameters.initial_nebular_temperature \
        = 6000.0 | units.K
    radiative_transfer.parameters.high_limit_of_the_frequency_mesh \
        = 15 | mocassin_rydberg_unit
    radiative_transfer.parameters.low_limit_of_the_frequency_mesh \
        = 1.001e-5 | mocassin_rydberg_unit

    radiative_transfer.parameters.total_number_of_photons = 10000000
    radiative_transfer.parameters.total_number_of_points_in_frequency_mesh \
        = 600
    radiative_transfer.parameters.convergence_limit = 0.09
    radiative_transfer.parameters.number_of_ionisation_stages = 6
    radiative_transfer.commit_parameters()

    return radiative_transfer
```

Listing 6.4: Routine to initialize the `Mocassin` radiative transfer code for an experiment with a single star as ionizing source in a homogeneous spherical cloud of molecular hydrogen gas.

The number of settable parameters in a radiative transfer code is quite large. Most of the fundamental code parameters used in radiative transfer can be queried from the helper function of the running code (see Section 1.5.1). In this example, we rely heavily on the code defaults and only set a few parameters explicitly to illustrate how this is done. Two important attributes are the grid resolution and the chemical abundances of the gas in the grid. In our example, we set these as follows:

```
    setup_grid(outer_radius, Ngrid)
    setup_abundancies(radiative_transfer)
```

Abundances of various elements are initialized as fractions relative to hydrogen. The following small snippet sets a few abundances, but printing the entire table will allow you to quickly develop an appreciation for the complexity of the chemical network available in these radiative transfer codes.

```
def setup_abundancies(code):
    table = code.abundancies_table()
    for atom in table.keys():
        table[atom] = 0.0
    table['H'] = 1.0
    table['He'] = 0.1
    table['C'] = 2.2e-4
    table['N'] = 4.e-5
    table['O'] = 3.3e-4
    table['Ne'] = 5.e-5
    table['S'] = 9.e-6
```

After initializing the grid and the abundances, we set the rest of the initial conditions, such as the temperature of the ambient gas and the limits to the frequency mesh:

```
    radiative_transfer.parameters.initial_nebular_temperature = 6000.0 | units.K
```

```
    radiative_transfer.parameters.high_limit_of_the_frequency_mesh \
        = 15 | mocassin_rydberg_unit
    radiative_transfer.parameters.low_limit_of_the_frequency_mesh \
        = 1.001e-5 | mocassin_rydberg_unit
```

We then initialize the run-specific code parameters, such as the total number of photons, the convergence limit, and the number of ionization stages in the experiment:

```
    radiative_transfer.parameters.total_number_of_photons = 10000000
    radiative_transfer.parameters.total_number_of_points_in_frequency_mesh = 600
    radiative_transfer.parameters.convergence_limit = 0.09
    radiative_transfer.parameters.number_of_ionisation_stages = 6
```

Finally we initialize the grid in the radiative transfer code and load the pre-evolved star into the code as the ionizing source:

```
    radiative_transfer.grid.hydrogen_density = grid.hydrogen_density
    radiative_transfer.commit_grid()
    radiative_transfer.particles.add_particle(star)
    radiative_transfer.commit_particles()
```

After all model- and code-specific parameters are set, the grid is initialized, and the ionizing source is defined, then we can start the event loop. The loop is controlled by setting a maximum value to the number of photons and tracking the degree to which convergence has been reached:

```
    max_number_of_photons \
        = 100*radiative_transfer.parameters.total_number_of_photons
    previous_percentage_converged = 0.0
```

The full event loop is shown in Listing 6.5. Figure 6.5 presents the radial temperature profile of the molecular cloud.

```
    step = 0
    while percentage_converged < min_convergence:

        radiative_transfer.step()
        percentage_converged = radiative_transfer.get_percentage_converged()
        print("percentage converged :", percentage_converged, ", step :", step,
              ", photons:",
              radiative_transfer.parameters.total_number_of_photons)

        if previous_percentage_converged > 5 and percentage_converged < 95:
            convergence_increase = (
                (percentage_converged - previous_percentage_converged)
                / previous_percentage_converged
            )
            if convergence_increase < 0.2 \
                and radiative_transfer.parameters.total_number_of_photons\
                    < max_number_of_photons:
                radiative_transfer.parameters.total_number_of_photons *= 2
```

```
169         step += 1
170         previous_percentage_converged = percentage_converged
```

Listing 6.5: Event loop for the radiative transfer code.

Figure 6.4: Degree of ionization $\xi_{\rm ion}$ due to a $100\,{\rm L}_\odot$ star at the center of a 3 pc spherically symmetric Plummer distribution of 10^4 gas particles. The red curve gives the average degree of ionization, binned in intervals of 0.1 pc. The figure was made using the script in List. 6.1 and took 20 seconds on a laptop, see also `${AMUSE_DIR}/examples/textbook/rad_minimal.py`.

6.3.3 Coupling Radiative Transfer with Hydrodynamics

Combining radiative transport with hydrodynamics is not trivial. In principle, we could adopt a linear coupling strategy between two modules, as discussed in Chapter 7, but a bridge-like approach (see Chapter 8) gives superior results, in terms of both better performance and smaller errors. However, as we have not yet discussed bridge [**check if this is still true – Steven**], we adopt a simpler but still effective approach similar to the interlaced temporal discretization discussed in Section 7.2.1.

We have defined a small radhydro class to simplify this function, to be incorporated later into a bridge with (for example) a gravity solver. In order for this class to comply with bridge specifications, it must have a few extra functions: the most important is evolve_model(). In addition, to make life easier, we include a method to copy data from the two worker codes to the framework, and one to clean up after we are done. For safety (and convenience), we start by synchronizing all relevant data between the two worker codes and the framework, giving us a self-consistent data set to work with. For additional safety, albeit probably unnecessary in this example, we perform a similar synchronization after the two worker codes have reached their objectives.

Between the two synchronization operations, we run the hydrodynamical simulation for half a time step, followed by the radiative transfer simulation for a full time step, and end with a second half of a time step for the hydrodynamics (see also Section 7.2.1):

```
1     while t < (tend-timestep/2):
2         self.hydro.evolve_model(t+timestep/2.)
3         update_rad_from_hydro_channel.copy()
4         self.rad.evolve_model(t+timestep)
5         update_hydro_from_rad_channel.copy()
6         self.hydro.evolve_model(t+timestep)
7         t = self.model_time
```

The rest of the radiation–hydrodynamics class is fairly straightforward. In Section 6.6.1, we present an example calculation using this RadHydro class using the source code listing in ${AMUSE_DIR}/examples/textbook/supernova_irradiation.py.

6.4 Examples

6.4.1 Heating of a Protoplanetary Disk

The abundance of several species of short-lived radionuclides (SLRs), such as ^{26}Al and ^{60}Fe, in the solar system has puzzled astronomers for some time (MacPherson *et al.*, 1995). Key questions include: How did these volatile elements, particularly ^{26}Al, enter the solar system? How did they come to be part of meteorites? and how did chondrules (the metallic grains containing these elements) form before chondrites (the meteorites that contain them)? There seems to be a lot we do not really understand.

One possible explanation involves a nearby supernova when the Sun was still part of its parent star cluster. This is a two-stage process. First, the radiation from a bright supernova illuminates and heats the disk. Second, a few decades later, the hot ionized gas from the explosion plows through it. The idea is that it might be possible for the first burst of radiation to heat the disk enough to vaporize the dust, such that the subsequent wave of radioactive material could deposit the SLRs. Later cooling of the disk might subsequently have caused these SLRs to become part of the newly formed dust. This is all speculative, but worthy of further exploration.

Here, we study the first stage of the process: the heating and ionization of a protoplanetary circumstellar disk by the radiation from the supernova. For the gaseous disk, we adopt the protoplanetary disk model discussed in Section 5.3.2. To mimic the thick protoplanetary disk, we adopt a Safronov–Toomre Q parameter of $Q = 25$, rather than $Q = 2$ as in the earlier example of a thin disk (see Section 5.3.2). The disk was constructed as in the earlier example, with 10^4 equal-mass SPH particles at distances between 1 au and 100 au from the Sun, a total mass of 0.01 $M_\odot$ and a power-law density profile with an exponent of -2. We arbitrarily assume that the disk is oriented at an angle of 15° (almost face-on) to the line of sight of the exploding star. The hydrodynamical simulation was run using `Fi` with an isothermal equation of state and an artificial viscosity parameter of $\alpha = 0.1$ (see Section 5.2.4).

For the evolution of the supernova luminosity, we follow Sanders *et al.* (2015) and adopt a simple broken power-law luminosity function with characteristic luminosity and time scales. The script ${AMUSE_DIR}/examples/textbook/supernova_IIp_Lightcurve.py presents code from Sanders *et al.* (2015) to reproduce those light curves. We adopt two distinct supernovae, SN PS1-11aof, with typical luminosities of a few times $\sim 10^9\,L_\odot$ when the peak is reached about 10 days after the onset of the explosion. In our example, we adopt the best-fit parameters for this type IIp supernova, which has a peak luminosity of 2×10^{43} erg/s, and the other supernova PS1-10a is somewhat dimmer. The light curves for two supernovae are shown as the blue curves in Figure 6.3.

Figure 6.5: Radial electron temperature profile of a centrally illuminated H_2 cloud. The blue points indicate the initial temperature, the red points the final temperature. A script to generate this figure can be found at `${AMUSE_DIR}/examples/textbook/electrontemperature_profile_of_H2cloud.py`. It should take roughly xxx minutes to run.

6.5 Validation

A number of standard tests can be used to validate the results of radiative transfer codes in a steady state. Approximate analytic solutions exist for a few problems, and radiative transfer codes are tested exhaustively against them. Every serious radiative transfer code performs well on all of these tests.

These tests are important, and any code must pass them, but the real challenge lies in problems where the medium through which the radiation passes evolves dynamically in response to the radiation. Very few analytic (or even approximate analytic) solutions exist for such problems, so definitive tests in these cases are rare. There is, however, a large research community actively addressing these issues, and they have designed several standard tests to which every code should be expected to yield qualitatively similar solutions. Two excellent examples of large collaborative efforts in this area have been carried out by Iliev *et al.* (2006, 2009). These tests are limited to ionizing radiation passing through an optically thick medium, which is an important process and one of the hardest problems in radiation transfer, as discussed in Section 6.1.1.1. It is difficult, in this context, to compare methods that make different assumptions about the problem, but at least having a suite of tests allows a researcher to validate the results of a radiative transfer code against earlier calculations.

Because testing codes against codes is the principal remaining validation strategy, we show here just one example. This is the last validation example in this book. Any further calculations using multiphysics solvers must be validated to the best degree possible, but we cannot offer any guarantee for the correctness of the results. The importance of radiative hydrodynamics across many theoretical and numerical astronomical problems has led to a wealth of papers on the cross-validation of methods and implementations. A few of the more recent ones include: Teyssier (2002); Leibrandt *et al.* (2005); Agertz *et al.* (2007); Iliev *et al.* (2009); Kolb *et al.* (2013); Ishii *et al.* (2017); Kim *et al.* (2017); Abe *et al.* (2018).

We run test suite number 5 of Iliev *et al.* (2009), and present the resulting temperature evolution in Figure 6.7. This test calculates the expansion of an HII region in a homogeneous medium using four different solvers (see also Pelupessy *et al.*, 2013; for details). We simplify the problem here by adopting a solution with single-frequency radiative transfer to study the thermal evolution of the gas, but the same strategy should work with a full radiative solver. We carry out this test using Fi or Gadget2 for the hydrodynamics solver and SimpleX or SPHRay for the radiative transfer. The coupling of the hydrodynamics code with the radiative-transfer solver is realized using the strategy discussed in Section 6.5.1.

Figure 6.6: Degree of ionization $\xi_{\rm ion}$ due to a $100\,{\rm L}_\odot$ star at the center of a 3 pc spherically symmetric Plummer distribution of 10^4 gas particles. The red curve gives the average degree of ionization, binned in intervals of 0.1 pc. The figure was made using the script in List. 6.1 and took 20 seconds on a laptop, see also `${AMUSE_DIR}/examples/textbook/rad_minimal.py`.

6.5.1 Coupling Radiative Transfer with Hydrodynamics

Combining radiative transport with hydrodynamics is not trivial. In principle, we could adopt a linear coupling strategy between two modules, as discussed in Chapter 7, but a bridge-like approach (see Chapter 8) gives superior results, in terms of both better performance and smaller errors. However, as we have not yet discussed bridge [**check if this is still true – Steven**], we adopt a simpler but still effective approach similar to the interlaced temporal discretization discussed in Section 7.2.1.

We have defined a small radhydro class to simplify this function, to be incorporated later into a bridge with (for example) a gravity solver. In order for this class to comply with bridge specifications, it must have a few extra functions: the most important is evolve_model(). In addition, to make life easier, we include a method to copy data from the two worker codes to the framework, and one to clean up after we are done. For safety (and convenience), we start by synchronizing all relevant data between the two worker codes and the framework, giving us a self-consistent data set to work with. For additional safety, albeit probably unnecessary in this example, we perform a similar synchronization after the two worker codes have reached their objectives.

Between the two synchronization operations, we run the hydrodynamical simulation for half a time step, followed by the radiative transfer simulation for a full time step, and end with a second half of a time step for the hydrodynamics (see also Section 7.2.1):

```
    while t < (tend-timestep/2):
        self.hydro.evolve_model(t+timestep/2.)
        update_rad_from_hydro_channel.copy()
        self.rad.evolve_model(t+timestep)
        update_hydro_from_rad_channel.copy()
        self.hydro.evolve_model(t+timestep)
        t = self.model_time
```

The rest of the radiation–hydrodynamics class is fairly straightforward. In Section 6.6.1, we present an example calculation using this RadHydro class using the source code listing in ${AMUSE_DIR}/examples/textbook/supernova_irradiation.py.

6.6 Examples

6.6.1 Heating of a Protoplanetary Disk

The abundance of several species of short-lived radionuclides (SLRs), such as ^{26}Al and ^{60}Fe, in the solar system has puzzled astronomers for some time (MacPherson *et al.*, 1995). Key questions include: How did these volatile elements, particularly ^{26}Al, enter the solar system? How did they come to be part of meteorites? and how did chondrules (the metallic grains containing these elements) form before chondrites (the meteorites that contain them)? There seems to be a lot we do not really understand.

One possible explanation involves a nearby supernova when the Sun was still part of its parent star cluster. This is a two-stage process. First, the radiation from a bright

supernova illuminates and heats the disk. Second, a few decades later, the hot ionized gas from the explosion plows through it. The idea is that it might be possible for the first burst of radiation to heat the disk enough to vaporize the dust, such that the subsequent wave of radioactive material could deposit the SLRs. Later cooling of the disk might subsequently have caused these SLRs to become part of the newly formed dust. This is all speculative, but worthy of further exploration.

Here, we study the first stage of the process: the heating and ionization of a protoplanetary circumstellar disk by the radiation from the supernova. For the gaseous disk, we adopt the protoplanetary disk model discussed in Section 5.3.2. To mimic the thick protoplanetary disk, we adopt a Safronov–Toomre Q parameter of $Q = 25$, rather than $Q = 2$ as in the earlier example of a thin disk (see Section 5.3.2). The disk was constructed as in the earlier example, with 10^4 equal-mass SPH particles at distances between 1 au and 100 au from the Sun, a total mass of 0.01 $M_\odot$ and a power-law density profile with an exponent of -2. We arbitrarily assume that the disk is oriented at an angle of 15° (almost face-on) to the line of sight of the exploding star. The hydrodynamical simulation was run using `Fi` with an isothermal equation of state and an artificial viscosity parameter of $\alpha = 0.1$ (see Section 5.2.4).

For the evolution of the supernova luminosity, we follow Sanders *et al.* (2015) and adopt a simple broken power-law luminosity function with characteristic luminosity and time scales. The script ${AMUSE_DIR}/examples/textbook/supernova_IIp_Lightcurve.py presents code from Sanders *et al.* (2015) to reproduce those light curves. We adopt two distinct supernovae, SN PS1-11aof, with typical luminosities of a few times $\sim 10^9\,L_\odot$ when the peak is reached about 10 days after the onset of the explosion. In our example, we adopt the best-fit parameters for this type IIp supernova, which has a peak luminosity of 2×10^{43} erg/s, and the other supernova PS1-10a is somewhat dimmer. The light curves for two supernovae are shown as the blue curves in Figure 6.3.

Figure 6.7: Expansion of an ionization front in an initially homogeneous medium (see test #5 of Iliev *et al.* (2009)). We show here the temperature structure as a function of distance (normalized to the box size $L_{\rm box}$) after 10, 30. and 200 Myr. The source has a ionizing luminosity $L = 5 \times 10^{48}$photons s^{-1}. Calculations are performed using `Fi` with `SimpleX` (blue curve), `Gadget2` with SimpleX (green), `Fi` with `SPHRay` (yellow), and `Gadget2` with `SPHRay` (red). A more extensive version of this test can be found in Pelupessy *et al.* (2013), see their Fig.22. Using script `${AMUSE_DIR}/examples/applications/iliev_test5.py`, this figure can be generated in about 10 minutes on a fast workstation.

Both radiative codes show different behavior in the fall-off of the temperature, mainly due to the different ways in which high-energy photons are treated. A similar divergence in the ionization-front profiles is observable in Iliev *et al.* (2009). The choice of hydrodynamic code has a minor effect on the temperature profile. Other differences can be traced to the specifics of the numerical treatments and the parameter settings of individual methods. The coupling with SPHRay, for example, results in more scatter in the temperature in Figure 6.7 than does the calculation with SimpleX. This is caused by using only 2×10^6 rays, which is a rather small number as compared to the rays used in the Monte Carlo solver.

Unresolved results are generally hard to spot in radiative transfer calculations, and can only be validated by repeating the simulation at higher resolution. The calculation

presented in Figure6.3 was performed using from 10^6 up to a maximum of 10^9 rays per hour. The latter calculation is expensive in terms of computer time, but the difference between the runs with 10^8 and 10^9 rays was smaller than the thickness of the line. One might then argue that 10^8 rays is sufficient. Note, however, that changing the resolution of the underlying SPH model would again change the results, and a new convergence test would be needed for validation. Therefore, the response of the results to the resolution depends not only on the resolution of the radiative transfer calculation but also on the resolution of the underlying gas code. This poses interesting challenges for validating coupled radiative-hydrodynamics problems.

6.7 Assignments

6.7.1 Habitability of a protoplanetary disk

The habitability of a protoplanetary disk depends on the local temperature. We can calculate the disk temperature structure using a radiative transfer code acting on a gaseous disk. This assignment therefore consists of two parts: (1) obtaining a static hydrodynamical realization of a protoplanetary disk, and (2) allowing the disk to evolve. The radiative transfer codes available in `amuse` however, are dedicated to ionizing radiation, and calculating the temperature structure in the disk around a low-mass star will therefore not give the correct answer. In this example we consider a disk around a high-mass star, which we can address using the available radiative transfer codes.

1. Generate the initial conditions for a protoplanetary disk of 1% of the central star's mass, extending from 1 au to 100 au. Adopt a Toomre Q parameter of 2 (see Section 6.6.1), which will result in a relatively thin disk. In Section 5.3.2, we discussed how to generate such a disk model. Perform the simulation for a star of 10 $M_\odot$ at an age of 10 Myr (see Section 4.2.2). You should run a stellar-evolution code to obtain the appropriate luminosity of the star.

2. Measure the temperature in the disk for a duration of 1 Myr, and make a plot of local temperature as a function of radius. Determine the extent of the so-called habitable zone in the disk—the region where the temperature lies between 273 K and 373 K (a broad range, but one in which Earth organisms are known to survive and thrive; see Miroshnichenko *et al.*, 2001). Life also seems to be able to flourish in more extreme environments (Rothschild & Mancinelli, 2001), and you may want to extend the range of the habitable zone to include psychrophilic organisms at temperatures below the freezing point of water (Cavicchioli *et al.*, 2002).

3. For a more massive (brighter) star, the habitable zone moves outward. Plot the range of the habitable zone as a function of the luminosity of the central star. What can you say about the radial temperature structure within the habitable zone as the mass of the star increases, other than that it gets bigger and moves

outward? Specifically, make a plot of the mean temperature in the habitable zone as a function of the distance to the star and as a function of the mass of the star.

6.7.2 Bumpy disks

When running the disk with a bump problem in Section 5.3.2, it is clear that the bump shields part of the disk from the central irradiating source. As a consequence, the disk behind the bump remains cooler compared to its surroundings. One might wonder whether it is possible to create a habitable zone behind such a bump, or is the lifetime of the bump too short for life to develop? In order to prevent the bump from dispersing, we put a massive planet or a secondary star in the bump.

In this assignment, you will simulate a protoplanetary disk with a bump, in order to study the temperature evolution of the region behind the bump and compare this with the temperature elsewhere.

1. Generate the initial conditions for a protoplanetary disk with a bump around a 10 $M_\odot$ star. The mass of the disk should be 1% of the central star's mass, ranging from 1 au to 100 au. Adopt a Toomre Q parameter of 10, which will result in a relatively thick disk. Generate a Plummer density distribution with a mass of 10% of the disk and a radius of 10 au, and put it in a circular orbit at a distance of 50 au around the central star (see Section5.3.2). While running, use `hop` (see Section 7.3.1) to keep track of the position, velocity, size, and mass of the bump.

2. Measure the temperature in the disk for as long as the bump has at least 30% of its initial mass. Plot the mean temperature in the bump as a function of time, and of the point diagonally across the disk at the other side and at the same distance from the central star. This will give you a sort of differential measurement of the temperature in the disk with and without bump.

3. Repeat the calculation for a 10 $M_\odot$ star at the terminal-age main sequence, at the beginning of the giant branch, and when it ascends the asymptotic giant-branch.

4. Is there any moment in time at which the bump would be a better place to live than at the other side of the disk?

Bibliography

Abe, Makito, Suzuki, Hiroyuki, Hasegawa, Kenji, Semelin, Benoit, Yajima, Hidenobu, & Umemura, Masayuki. 2018. SEURAT: SPH scheme extended with ultraviolet line radiative transfer. *MNRAS* , **476**(2), 2664–2673.

Agertz, Oscar, Moore, Ben, Stadel, Joachim, Potter, Doug, Miniati, Francesco, Read, Justin, Mayer, Lucio, Gawryszczak, Artur, Kravtsov, Andrey, Nordlund, Åke, Pearce, Frazer, Quilis, Vicent, Rudd, Douglas, Springel, Volker, Stone, James, Tasker, Elizabeth, Teyssier, Romain, Wadsley, James, & Walder, Rolf. 2007. Fundamental differences between SPH and grid methods. *MNRAS* , **380**(3), 963–978.

Altay, Gabriel, Croft, Rupert A. C., & Pelupessy, Inti. 2008. SPHRAY: a smoothed particle hydrodynamics ray tracer for radiative transfer. *MNRAS* , **386**(4), 1931–1946.

Altay, Gabriel, Croft, Rupert A. C., & Pelupessy, I. 2011 (Mar.). *SPHRAY: A Smoothed Particle Hydrodynamics Ray Tracer for Radiative Transfer*. Astrophysics Source Code Library, record ascl:1103.009.

Baba, T. 2008. Slow light in photonic crystals. *Nat Photon*, **465**, 473.

Bohren, Craig F., & Huffman, Donald R. 1983. *Absorption and scattering of light by small particles*.

Cavicchioli, Ricardo, Siddiqui, Khawar S, Andrews, David, & Sowers, Kevin R. 2002. Low-temperature extremophiles and their applications. *Current Opinion in Biotechnology*, **13**(3), 253 – 261.

Chandrasekhar, Subrahmanyan. 1960. *Radiative transfer*.

Delaunay, B. 1934. Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS*, 793–800.

Ercolano, B., Barlow, M. J., Storey, P. J., & Liu, X. W. 2003. MOCASSIN: a fully three-dimensional Monte Carlo photoionization code. *MNRAS* , **340**(4), 1136–1152.

Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W., & Tufo, H. 2000. FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes. *ApJS* , **131**(1), 273–334.

Gerritsen, J. P. E., & Icke, V. 1997. Star formation in N-body simulations. I. The impact of the stellar ultraviolet radiation on star formation. *A&A* , **325**(Sept.), 972–986.

Hernquist, Lars, & Katz, Neal. 1989. TREESPH: A Unification of SPH with the Hierarchical Tree Method. *ApJS* , **70**(June), 419.

Hubber, D. A., Ercolano, B., & Dale, J. 2016. Observing gas and dust in simulations of star formation with Monte Carlo radiation transport on Voronoi meshes. *MNRAS* , **456**(1), 756–766.

Iliev, Ilian T., Ciardi, Benedetta, Alvarez, Marcelo A., Maselli, Antonella, Ferrara, Andrea, Gnedin, Nickolay Y., Mellema, Garrelt, Nakamoto, Taishi, Norman, Michael L., Razoumov, Alexei O., Rijkhorst, Erik-Jan, Ritzerveld, Jelle, Shapiro, Paul R., Susa, Hajime, Umemura, Masayuki, & Whalen, Daniel J. 2006. Cosmological radiative transfer codes comparison project - I. The static density field tests. *MNRAS* , **371**(3), 1057–1086.

Iliev, Ilian T., Whalen, Daniel, Mellema, Garrelt, Ahn, Kyungjin, Baek, Sunghye, Gnedin, Nickolay Y., Kravtsov, Andrey V., Norman, Michael, Raicevic, Milan, Reynolds, Daniel R., Sato, Daisuke, Shapiro, Paul R., Semelin, Benoit, Smidt, Joseph, Susa, Hajime, Theuns, Tom, & Umemura, Masayuki. 2009. Cosmological radiative transfer comparison project - II. The radiation-hydrodynamic tests. *MNRAS* , **400**(3), 1283–1316.

Ishii, Ayako, Ohnishi, Naofumi, Nagakura, Hiroki, Ito, Hirotaka, & Yamada, Shoichi. 2017. Validation of radiative transfer computation with Monte Carlo method for ultra-relativistic background flow. *Journal of Computational Physics*, **348**(Nov.), 612–633.

Kim, Jeong-Gyu, Kim, Woong-Tae, Ostriker, Eve C., & Skinner, M. Aaron. 2017. Modeling UV Radiation Feedback from Massive Stars. I. Implementation of Adaptive Ray-tracing Method and Tests. *ApJ* , **851**(2), 93.

Kolb, Stefan M., Stute, Matthias, Kley, Wilhelm, & Mignone, Andrea. 2013. Radiation hydrodynamics integrated in the PLUTO code. *A&A* , **559**(Nov.), A80.

Leibrandt, D. R., Drake, R. P., Reighard, A. B., & Glendinning, S. G. 2005. A Validation Test of the Flux-limited Diffusion Approximation for Radiation Hydrodynamics. *ApJ* , **626**(1), 616–625.

MacPherson, G. J., Davis, A. M., & Zinner, E. K. 1995. The distribution of aluminum-26 in the early Solar System - A reappraisal. *Meteoritics*, **30**(July), 365.

Miroshnichenko, Margarita L., Hippe, Hans, Stackebrandt, Erko, Kostrikina, Nadezhda A., Chernyh, Nikolai A., Jeanthon, Christian, Nazina, Tamara N., Belyaev, Sergei S., & Bonch-Osmolovskaya, Elizaveta A. 2001. Isolation and characterization of Thermococcus sibiricus sp. nov. from a Western Siberia high-temperature oil reservoir. *Extremophiles*, **5**(2), 85–91.

Paardekooper, J. P., Kruip, C. J. H., & Icke, V. 2010. SimpleX2: radiative transfer on an unstructured, dynamic grid. *A&A* , **515**(June), A79.

Pelupessy, F. I., van der Werf, P. P., & Icke, V. 2004. Periodic bursts of star formation in irregular galaxies. *A&A* , **422**(July), 55–64.

Pelupessy, F. I., van Elteren, A., de Vries, N., McMillan, S. L. W., Drost, N., & Portegies Zwart, S. F. 2013. The Astrophysical Multipurpose Software Environment. *A&A* , **557**(Sept.), A84.

Peraiah, Annamaneni. 2001. *An Introduction to Radiative Transfer*.

Plummer, H. C. 1911. On the problem of distribution in globular star clusters. *MNRAS* , **71**(Mar.), 460–470.

Portegies Zwart, S. F., & Verbunt, F. 1996. Population synthesis of high-mass binaries. *A&A* , **309**(May), 179–196.

Portegies Zwart, S. F., & Verbunt, F. 2012 (Jan.). *SeBa: Stellar and binary evolution.* Astrophysics Source Code Library, record ascl:1201.003.

Ritzerveld, Jelle, & Icke, Vincent. 2006. Transport on adaptive random lattices. *Phys. Rev. E.* , **74**(2), 026704.

Rothschild, Lynn J., & Mancinelli, Rocco L. 2001. Life in extreme environments. *Nat* , **409**(6823), 1092–1101.

Sanders, N. E., Soderberg, A. M., Gezari, S., Betancourt, M., Chornock, R., Berger, E., Foley, R. J., Challis, P., Drout, M., Kirshner, R. P., Lunnan, R., Marion, G. H., Margutti, R., McKinnon, R., Milisavljevic, D., Narayan, G., Rest, A., Kankare, E., Mattila, S., Smartt, S. J., Huber, M. E., Burgett, W. S., Draper, P. W., Hodapp, K. W., Kaiser, N., Kudritzki, R. P., Magnier, E. A., Metcalfe, N., Morgan, J. S., Price, P. A., Tonry, J. L., Wainscoat, R. J., & Waters, C. 2015. Toward Characterization of the Type IIP Supernova Progenitor Population: A Statistical Sample of Light Curves from Pan-STARRS1. *ApJ* , **799**(2), 208.

Teyssier, R. 2002. Cosmological hydrodynamics with adaptive mesh refinement. A new high resolution code called RAMSES. *A&A* , **385**(Apr.), 337–364.

Voronoi, Georges. 1908. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs.

Journal für die reine und angewandte Mathematik, **134**, 198–287.