Modelling a Type-II Supernova

F.S. Nobels, J. Ubink, H.W. de Vries
Guided by: Prof. Dr. O. Scholten

January 21, 2015

Abstract

A code was created for modelling the hydrodynamics of a type-II core-collapse supernova. Whereas the typical supernova explosion was not observed due to instability of the code, various succesful tests were performed on a more simple, earth-like atmosphere. Observing various shockwave and equilibrium phenomena in the earth-like atmosphere, it can be concluded that the hydrodynamics of the supernova model probably work correctly. Possible improvements to the model would consist of adding radiative transfer and a neutrino-heating mechanism. A code as been written for the former, but this has not been incorporated in the model in a working manner yet.
Contents

1 Introduction .............................. 4
  1.1 Star evolution and mass criterion .............................................. 5
  1.2 Mechanism of the type II supernova .............................................. 5
  1.3 Stellar composition ................................................................. 6
    1.3.1 Neutron star core model ...................................................... 6
    1.3.2 Red giant-like star ............................................................ 6
  1.4 Mean molecular mass ............................................................... 7

2 Physics of supernovae ....................... 8
  2.1 Pressure, energy, and temperature .............................................. 8
  2.2 Luminosity ............................................................................ 9
  2.3 Decay processes ................................................................. 11
  2.4 Opacity ............................................................................. 12
  2.5 Blast Waves ...................................................................... 13

3 Creating a numerical model ................. 15
  3.1 Basic assumptions ................................................................. 15
    3.1.1 Spherical symmetry ............................................................ 15
    3.1.2 Gas composition ............................................................... 15
    3.1.3 Homogeneous distributions .................................................. 15
  3.2 Discretizations ................................................................. 15
    3.2.1 Stellar shells ................................................................. 15
    3.2.2 Shell mass continuity ......................................................... 16
    3.2.3 Time discretization ............................................................ 18
    3.2.4 Variable time step ............................................................. 18

4 Numerical equations ........................... 19
  4.1 Discretization of the physical system ............................................ 19

5 General program structure ..................... 21

6 Test on Earth's atmosphere .................... 22
  6.1 Initial conditions .............................................................. 22
  6.2 Results ............................................................................ 23
    6.2.1 Equilibrium state ............................................................. 23
    6.2.2 Non-equilibrium state ....................................................... 29
    6.2.3 Theoretical soundwave speeds .............................................. 34
    6.2.4 Simulations on shockwaves .................................................. 38
  6.3 Conclusions of atmosphere simulations ........................................ 41

7 Supernova simulation .......................... 42
  7.1 Initial conditions .............................................................. 42
    7.1.1 Neutron star core model ...................................................... 42
    7.1.2 Red giant-like model .......................................................... 42
  7.2 Opacity results ................................................................. 43
  7.3 Results ............................................................................ 46
8 Discussion

8.1 Broken causality ..................................................... 52
8.2 Conservation of energy ........................................... 52
8.3 Comparing this paper's model to the original model of Arnett .................................................. 52
  8.3.1 Homologous expansion ...................................... 52
  8.3.2 Pressure ....................................................... 53
  8.3.3 Presence of radioactive elements ......................... 56
8.4 Light curves ...................................................... 56
8.5 General improvement: smaller discretizations .............. 56

9 Conclusion .......................................................... 57

10 Alternatives and prospects ...................................... 58
  10.1 Other methods of modelling .................................. 58
  10.2 Follow-up: adding the magnetar ............................ 58

A Derivation of the dynamics of an Earth-like atmosphere .... 61

B Full Python script .................................................. 63

C Opacity code ....................................................... 70
1 Introduction

A star’s death may be accompanied by a tremendous explosion, a so called supernova. Being the brightest events in the entire universe, a typical supernova can outshine an entire galaxy, emitting a typical amount of $10^{46} \text{J}$ of energy as radiation\[17\]. Understanding the supernova explosion is of interest for deep space observations. The plateau phase of a type-II supernova light curve can be used as a standard candle, which is a predictable light source that can be used as a gauging point for observations\[18\].

Recently, however, various supernova events were found not to be in accordance with the predicted light curves. Luminosities up to $L \propto 10^{44} \text{erg/s}$ were recorded, which differ up to two orders of magnitude from typical supernovae\[21\].

Thus, a theory explaining the aberrant behavior of these Superluminous supernovae was proposed. It poses that the high luminosities originate from magnetar-powered supernovae\[21\]\[17\]. The magnetar, a rapidly spinning, highly magnetic neutron star, would emit enormous amounts of radiation, which might interact with the ejected matter to increase the luminosity up to a hundredfold.

For this project, the aim was to model a typical supernova explosion, using Python. After succesfully modelling an explosion, a magnetar would be included in the calculations, to create the light curve that would accompany the sought after Superluminous Supernova. This report guides the reader on the journey from the physics of stars to a numerical simulation of the supernova explosion. Along the way, the physical assumptions and methods for discretizing and testing the system will be discussed as well. Before arriving at supernova simulations, a detour is made to numerical simulations of the earth’s atmosphere, which serve to confirm the physical correctness of the model.
1.1 Star evolution and mass criterion

In order for a supernova to occur, a star has to meet certain criteria, of which the most important one for a model will be its mass. If a star is too light, it will not be capable of showcasing a type II supernova. One can illustrate this criterion by having a look at the influence of mass on the fusion processes that occur in the star. Assuming that a star is a spherical object with a density $\rho(r)$, one can find a condition for mass continuity. This is done by dividing the star in infinitesimally thick shells:

$$ \frac{dM}{dr} = 4\pi r^2 \rho(r) . $$

(1)

With $\rho(r)$ a radial density distribution and $M$ and $r$ the mass and radial distance of the shell. This equation only holds if a pressure is provided, which acts against the gravitational forces that pull these shells into the centre\cite{11,16,24}. By integrating the mass continuity equation and plugging it into a standard formula for the gravitational force, one finds:

$$ F_G = - \left[ \frac{GM(r)}{r^2} \right] \rho(r) dr dA . $$

(2)

Where the symbols have their usual meaning. By using the condition that the buoyant force (resulting from the radial dependence of pressure) has to counteract the gravitational force, one can find the equation of hydrostatic equilibrium:

$$ \frac{dP}{dr} = - \left[ \frac{GM(r)}{r^2} \right] \rho(r) . $$

(3)

In this equation, $P$ denotes the pressure and the other variables again have their usual meaning. The consequence of this result is the following: a greater initial star mass will induce a greater pressure gradient. This has a direct influence on the possibility of fusion reactions, as the pressure at the core of the star provides the environment for nuclear fusion to occur. Initially, this will be a reaction of hydrogen into helium. After all the hydrogen has been converted, the core will shrink under influence of its own gravity. This converts the gravitational potential energy into kinetic energy, which creates temperatures great enough to start a new fusion reaction. However, further fusion reactions will stop occurring if the initial mass is not great enough to provide the pressure and gravitational energy necessary to start these fusion reactions\cite{4}.

1.2 Mechanism of the type II supernova

The mechanism that was just introduced does, however, not account for the working of a type-II supernova yet. Things get interesting when the fusion processes have turned the interior of the star into a core of inert iron and other stable elements like nickel. Fusion simply stops when all matter has fused to iron, as iron has the highest binding energy per nucleon, which means that fusion of iron atoms will not yield, but instead cost energy\cite{16}. When the star gets to the point that fusion does not occur anymore, it has to rely on another system for providing the pressure against gravitational collapse. Its last resort is electron degeneracy, which follows from the Pauli exclusion principle. The fact that each state can only be occupied by two electrons of opposite spin will prevent the core from collapsing, as long as it does not exceed the Chandrasekhar mass limit. As expected, this limit is exceeded when the star is heavy enough. To be precise, when a mass criterion of $1.44 M_\odot$ is satisfied, the Chandrasekhar limit will be exceeded and the core will implode under its own gravity\cite{3,14}. After the chain of fusion reactions has reached iron, the outer core will collapse onto the centre of the star. The rapidly shrinking core will emit gamma-rays, which start a photodisintegration reaction that turns the iron and nickel atoms into helium atoms and free neutrons. The core’s density will keep increasing until electrons and protons merge into neutrons, under the emission of a neutrino. The resulting, very dense core will primarily consist of neutrons, which will prevent further collapse through the strong nuclear force and neutron degeneracy. The leftover in-falling matter will rebound and move outward again in a tremendous shock wave. This shock wave is further powered by the capture of thermal neutrino’s, which are emitted by the very dense and very hot($\approx 10^8$ K) leftover core\cite{16}.
1.3 Stellar composition

In the supernova model, two different aspects of stellar composition are considered. The first of these is one in which the star contains a neutron star-like core, surrounded by spherical shells of stellar matter. In the other consideration, the star is similar to a red giant. In the red giant approximation, the star contains an iron core surrounded by layers of elements having lower atomic numbers. In the following section, these two stellar compositions will be treated in detail.

1.3.1 Neutron star core model

The first model used to simulate a core-collapse supernova consists of a star with an uniform chemical position on all radii, and a neutron star at its core. In other words, the star contains only the element hydrogen. At the center of this star there is a core with the properties of a neutron star. The fact that there is a neutron star at the center of the atmosphere, comes from the collapsing of the iron core in the pre-supernova situation. The iron core collapses to a very dense state of matter, comparable to a neutron star[24]. The atmosphere then collapses under gravity and rebounds on the core, resulting in a supernova. In the simulations, the model starts at the moment at which the atmosphere rebounds from the neutron star core.

1.3.2 Red giant-like star

An alternative to the Neutron star core model, is the star composition of a red giant, which is a more realistic one. It is assumed that at the model’s starting moment, the star shell composition is comparable to the composition of a red giant. The same assumption also applies to the mass of the star, which is set at $20 M_\odot$. The stellar material is assumed to consist of approximately 90% hydrogen. The stellar core, which consists of iron, has a mass of 1.4 solar masses, to make sure that the inert core will exceed the Chandrasekhar limit.

In the red giant-like model, the stellar atmosphere is divided into layers of different elements. Data were obtained from [28] for the temperature and the density at the edge of the shells for each particular element. For the used $20 M_\odot$ star, the density- and temperature distributions are given in table 1.

Table 1: Stellar composition of a $20 M_\odot$ star, at the end of its life

<table>
<thead>
<tr>
<th>Element</th>
<th>Main isotope</th>
<th>Total mass</th>
<th>Mean molecular mass (in u)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iron/neutrons</td>
<td>N.A.</td>
<td>$1.4 M_\odot$</td>
<td>N.A.</td>
</tr>
<tr>
<td>Silicon</td>
<td>$^{28}\text{Si}$</td>
<td>0.12 $M_\odot$</td>
<td>$\mu = \frac{28}{16} \approx 1.78$</td>
</tr>
<tr>
<td>Oxygen</td>
<td>$^{16}\text{O}$</td>
<td>0.12 $M_\odot$</td>
<td>$\mu = \frac{16}{8} \approx 1.78$</td>
</tr>
<tr>
<td>Neon</td>
<td>$^{20}\text{Ne}$</td>
<td>0.12 $M_\odot$</td>
<td>$\mu = \frac{20}{12} \approx 1.67$</td>
</tr>
<tr>
<td>Carbon</td>
<td>$^{12}\text{C}$</td>
<td>0.12 $M_\odot$</td>
<td>$\mu = \frac{12}{6} \approx 1.71$</td>
</tr>
<tr>
<td>Helium</td>
<td>$^{4}\text{He}$</td>
<td>0.12 $M_\odot$</td>
<td>$\mu = \frac{4}{2} \approx 1.33$</td>
</tr>
<tr>
<td>Hydrogen</td>
<td>$^{1}\text{H}$</td>
<td>18 $M_\odot$</td>
<td>$\mu = 0.6$ (assumption, from [24])</td>
</tr>
</tbody>
</table>

In the silicon shell, a constant density is assumed. Because the radius of the neutron star core is known, as well as the assumed values for the density and total mass in the silicon shell, it is possible to calculate the radius at which the end of the silicon shell lies. This is done using the following equation:

$$r_2 = \sqrt[3]{\frac{M}{\frac{4}{3}\pi \rho}} + r_1^3 = \sqrt[3]{\frac{0.12 M_\odot}{\frac{4}{3}\pi \rho}} + r_1^3.$$  (4)

The mass per subshell of a certain element can then easily be found by integrating over the volume and normalizing the outcome, as can be seen in the following:

1In the model, the stellar atmosphere is divided into a number of spherical shells. See section 3.2.1 for a detailed discussion.
\[ M = \int_0^\pi \int_0^{2\pi} \int_{r_1}^{r_2} \rho \, dV = 4\pi \int_{r_1}^{r_2} \rho r^2 \, dr = \frac{4\pi}{3} \rho (r_2^3 - r_1^3). \]  

(5)

One can use the same method for calculating the other shell radii. In this way, the thicknesses of each element layer can be determined. The size of the red giant is then found to be \(6.955 \cdot 10^8\) \(\text{km} \approx 7 \cdot 10^8\) \(\text{km}\). The results of these calculations are summarized in table 2.

<table>
<thead>
<tr>
<th>Shell edge</th>
<th>H-He</th>
<th>He-C</th>
<th>C-Ne</th>
<th>Ne-O</th>
<th>O-Si</th>
<th>Si-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density (10^3) (\text{kg m}^{-3})</td>
<td>4.53</td>
<td>0.968 (\cdot 10^3)</td>
<td>1.70 (\cdot 10^5)</td>
<td>3.10 (\cdot 10^9)</td>
<td>5.55 (\cdot 10^9)</td>
<td>4.26 (\cdot 10^7)</td>
</tr>
<tr>
<td>Temperature (10^7) (\text{K})</td>
<td>3.69</td>
<td>18.8</td>
<td>87.0</td>
<td>157</td>
<td>199</td>
<td>334</td>
</tr>
<tr>
<td>Calculated radius (km)</td>
<td>389066</td>
<td>69476</td>
<td>26395</td>
<td>21738</td>
<td>11020</td>
<td>12</td>
</tr>
</tbody>
</table>

1.4 Mean molecular mass

The mean molecular mass \(\mu\) is defined as the average mass (in amu) of the particles in a certain medium. The mean molecular mass can be calculated\[24\] using:

\[ \mu = \frac{\sum_i n_i m_i}{\sum_i n_i}. \]

(6)

In equation 6, \(n_i\) is the number of particles of a certain element or molecule, and \(m_i\) the corresponding mass. Using two simple approximations for protons, neutrons and electrons, one can simplify the calculation of the mean molecular masses. The mass of the proton and neutron are assumed to be equal to 1 amu \((m_H = m_p \approx 1\text{u})\). Another assumption is that the mass of the electrons is negligible \((m_e \approx 0)\).

It is also possible to calculate the mean molecular mass for atoms with an equal amount of protons and neutrons, this results in equation 7 below\[24\]:

\[ \mu_Z \approx \frac{(N_Z 2Ze) + (ZN_Z m_e)}{N_Z + ZN_Z} \approx \frac{(2Z + 0)N_Z}{(Z + 1)N_Z} \approx \frac{2Z}{Z + 1}. \]

(7)
2 Physics of supernovae

2.1 Pressure, energy, and temperature

For this model, it is assumed that the star consists of a mixture of photon gas and ideal classical gas. This assumption will be treated in more detail in section 3.1.2. The pressure of the stellar gas is then given by:

\[ P = \frac{a}{3} T^4 + \frac{\rho k_B T}{\mu}, \]  

(8)

where the two terms are respectively the partial pressures of the photon gas and the classical gas. Furthermore, the quantity \( \mu \) denotes the mean molecular mass of the classical gas, and \( k_B \) the Boltzmann constant.

The internal energy of the mixture is, similarly, given by the sum of the constituents’ internal energies:

\[ U = \frac{3}{2} k_B N T + a V T^4, \]  

(9)

where the first contribution comes from the classical gas and the second from the photon gas. Here, \( N \) denotes the number of particles, and \( a \) the radiation constant. In both equations, the photon gas terms increase faster with higher temperatures than the terms from the classical gas. This is because the photon gas terms are proportional to \( T^4 \), whereas the classical gas terms are linear with temperature. This suggests that for high temperatures, the photon gas will become dominant over the classical gas. It is therefore assumed that at the high temperatures within stars, the classical gas terms will be much smaller compared with the photon gas terms.

As will be described in section 3.2.1, the stellar atmosphere will be divided into a number of spherical shells. For the present discussion, the details of this discretization are not necessary. It suffices to know that the stellar atmosphere consists of thin layers of matter which can move only in the radial direction.

The temperature of a stellar shell changes due to changes in its energy. It is therefore of importance to first find expressions for the total energy of a shell. The first part of the total energy is the internal energy. The internal energy was already defined earlier (see equation 9). The second energy term is the kinetic energy, which is given by:

\[ E_{\text{kin}} = \frac{1}{2} m_{\text{shell}} \left( \frac{v_i + v_i + 1}{2} \right)^2, \]

(10)

where the average velocity of the two surfaces (i.e. the two shell radii, see section 3.2.1) enclosing a shell of stellar material are taken to represent the velocity of the matter inside the shell. The third and final term is the gravitational potential energy, and it has a negative contribution to the total energy. It is given by:

\[ E_{\text{grav}} = -\frac{GM_{\text{encl}} m_{\text{shell}}}{r^2}. \]  

(11)

So, combining the above equations gives the following result for the total energy of a stellar shell:

\[ U + E_{\text{kin}} + E_{\text{grav}} = \frac{3}{2} k_B N T + a V T^4 + \frac{1}{2} m_{\text{shell}} \left( \frac{v_i + 1 + v_i}{2} \right)^2 - \frac{GM_{\text{encl}} m_{\text{shell}}}{r^2}. \]  

(12)

As the various quantities in these expressions change over time, so too does the total energy of a shell change. When one assumes that the shell expands and shrinks adiabatically, its temperature is related to its internal energy. Therefore, one can determine the change in temperature of a shell by means of the change in energy.

The internal energy of a shell can change in two ways: by exchanging heat and by performing work:

\[ dU = dw + dq. \]  

(13)

Next it is assumed that the stellar gas expands adiabatically, so \( dq \) is set to zero. Then:
\[ dU = dw = -PdV. \]  
\[ (14) \]

Filling in equations 8 and 9 for the pressure and internal energy (where \( N = \frac{m_{\text{shell}}}{\mu} \)) respectively gives:

\[ d\left( \frac{3}{2} k \frac{m_{\text{shell}}}{\mu} T + aV T^4 \right) = - \left( \frac{\rho k T}{\mu} + \frac{aT^4}{3} \right) dV, \]
\[ (15) \]

which can be written as:

\[ \frac{3}{2} k \frac{m_{\text{shell}}}{\mu} dT + 4aVT^3 dT + aT^4 dV = - \left( \frac{\rho k T}{\mu} + \frac{aT^4}{3} \right) dV. \]
\[ (16) \]

This can be expressed in terms of the change in temperature as:

\[ dT = - \left( \frac{\rho k T}{\mu} + \frac{aT^4}{3} + aT^4 \right) \frac{dV}{\frac{3}{2} k m_{\text{shell}}}{\mu} + 4aVT^3}. \]
\[ (17) \]

This expression is the final result of this derivation. It describes how the temperature of a shell of stellar matter changes due to changes in its volume.

### 2.2 Luminosity

Luminosity is defined as the energy radiated by a body per unit time. This gives the following, simple relation between an object’s luminosity and its total radiated energy per second:

\[ L = \frac{dE_{\text{radiated}}}{dt}. \]
\[ (18) \]

The luminosity of a star is a very important in the discussion of solar physics. Unfortunately, the model does not include luminosity. Every attempt at incorporating luminosity resulted in enormous temperature increases throughout the entire stellar atmosphere, and thus to instabilities an therefore to a not-usable program. In this section, the unstable temperature expression due to luminosity is derived. It needs to be stessed that the final result is not used in the program, and is only discussed here for completeness’ sake.

So, in order to derive an expression, it is firstly assumed that the star can be treated as a radiating black body. In that case, the luminosity of the star is given by:

\[ L = 4\pi R^2 \sigma T^4, \]
\[ (19) \]

where \( R \) is the radius of the star and \( \sigma \) is the Stefan-Boltzmann constant.

When the matter in two adjacent stellar shells have different temperatures, there is of course an exchange of heat between these two shells. One of the ways through which this exchange takes place, is through radiative heating. The hotter shell of the two emits more blackbody radiation than the cooler one, causing a net inflow of radiation into the cooler shell, and a net outflow of radiation from the hotter shell. This lowers the temperature of the hotter shell, and raises the temperature of the lower shell. In reality, convectional flows play an important role in heat exchange, but due to the enormous mathematical and numerical complexity of these processes, they are not included in the program.

In order to determine how stellar shells exchange heat through radiation, it is necessary to find how differences in luminosity result in a change in temperature. In the following derivation the assumption is made that each shell exchanges radiation only with its two adjacent shells. One can therefore assume that radiation can only travel through one shell. In other words, each shell completely absorbs the radiation it

---

2Note: \( a = \frac{\pi^5 R^3}{15k^3 c^3 N_a^2} \) with \( R \) and \( N_a \) the gas constant and Avogadro’s number, respectively.
receives from adjacent shells. Furthermore, each shell can be thought of as a separate blackbody. One must first find an expression for how the temperature changes due to radiative heating for a shell somewhere in the middle of the star. After this, an expression can be derived for both the outer and innermost shells.

First, consider a shell $i$, which has both an underlying shell and one above it. Shell $i$ emits radiation to, and absorbs it from, the $i-1$th and $i+1$th shell. The net influx of energy of this shell – the net luminosity – is given by:

$$L_{\text{net}} = L_{i-1 \rightarrow i} + L_{i+1 \rightarrow i} - L_{i \rightarrow i-1} - L_{i \rightarrow i+1} \quad (20)$$

Here, the luminosity is positive when it flows into a shell, and negative when it flows out of it. Treating each shell as a separate black body, one gets:

$$L_{\text{net}} = 4\pi \sigma (R^2_i T_i^4 + R^2_{i+1} T_{i+1}^4 - R^2_i T_i^4 - R^2_{i+1} T_{i+1}^4). \quad (21)$$

In the program, finite time steps are used. It is therefore of importance to know by which amount the internal energy of the $i$th shell changes during such a time step. It follows straightforwardly from equation (18) that the increase in energy during a time step $\Delta t$ equals:

$$E = \int_t^{(t+\Delta t)} L dt \approx L \Delta t, \quad (22)$$

where the approximation in the last term is valid for small time steps.

Changes in internal energy and temperature are related to each other through the heat capacity, $C = \frac{dU}{dT}$. The heat capacity of the matter inside the star is therefore needed in order to determine how changes in internal energy relate to changes in temperature. As mentioned earlier, the star is considered to be made up of both a thermodynamic gas and a photon gas. These gases are intermixed and have the same temperature. Assuming that the radiation only affects the photon gas, one can use the heat capacity of a perfect photon gas for determining how each shell’s temperature changes due to changes in energy. The internal energy of a photon gas is given by:

$$U = \frac{\pi^2 k^4}{15c^3h^3} VT^4. \quad (23)$$

If one assumes that the volume stays constant during the time in which radiative transfer of heat takes place (which is a reasonable approximation for small time steps), the heat capacity of a photon gas is equal to:

$$C = \frac{4\pi^2 k^4}{15c^3h^3} VT^3. \quad (24)$$

One can thus find $\frac{dU}{dT} = \frac{4\pi^2 k^4}{15c^3h^3 VT^3}$, which can also be written as:

$$dU = \frac{4\pi^2 k^4}{15c^3h^3} VT^3 dT, \quad (25)$$

or in other words:

$$dT = \frac{dE}{\frac{4\pi^2 k^4}{15c^3h^3} VT^3} \quad (26)$$

Which is set equal to:

$$\Delta T = \frac{\Delta E}{\frac{4\pi^2 k^4}{15c^3h^3} VT^3} \quad (27)$$

Using equation (22) this can be rewritten as:

$$\Delta T = \frac{L \Delta t}{\frac{4\pi^2 k^4}{15c^3h^3} VT^3} \quad (28)$$
Therefore equation \[28\] can be used for calculating by which amount the temperature changes due to the net luminosity of a shell. The net luminosity is taken from equation \[21\] and combine this with equation \[28\] to get the following:

\[
\Delta T = \frac{4\pi \sigma (R_i^2 T_i^4 + R_{i+1}^2 T_{i+1}^4 - R_i^2 T_i^4 - R_{i+1}^2 T_{i+1}^4) \Delta t}{15c^2 k T^3}
\]  
\tag{29}
\]

This equation is the final result of this derivation. It can be used for determining the temperature changes due to radiative heat. Usage of this expression leads to enormous changes in temperature, and is not used in the program. This likely caused by the fact that equation \[19\] describes the luminosity of entire stars, and not spherical stellar shells. This means that it probably cannot be used to describe the luminosity of stellar shells in the way it was done in this derivation.

### 2.3 Decay processes

In reality, a purely kinetic-based light curve does not fit observed light curves very well. This is because there are more sources contributing to the energy in equation \[25\]. One has to also take radioactive decay processes into consideration as a considerable source of energy. A large contributor from these decay processes is the decay of $^{56}\text{Ni}$ to $^{56}\text{Fe}$ via $^{56}\text{Co}$. As before, the expression derived in this section is not included in the program.

The following derivation comes from Nadyozhin (1994)\[19\]:

\[
^{56}\text{Ni} \rightarrow ^{56}\text{Co} + \gamma + \nu_e
\]  
\tag{30}
\]

\[
^{56}\text{Co} \rightarrow \begin{cases} ^{56}\text{Fe} + \gamma + \nu_e \\ ^{56}\text{Fe} + e^+ + \gamma + \nu_e \end{cases}
\]  
\tag{31}
\]

The amount of each isotope is governed by a set of differential equations:

\[
\frac{dN_{Ni}}{dt} = -\frac{N_{Ni}}{\tau_{Ni}}
\]  
\tag{32}
\]

\[
\frac{dN_{Co}}{dt} = \frac{N_{Ni}}{\tau_{Ni}} - \frac{N_{Co}}{\tau_{Co}}
\]  
\tag{33}
\]

Using the starting amount of $^{56}\text{Ni}$-isotopes, $N_{Ni,0}$ and the fact that there are no Cobalt-isotopes at the start of the explosion, one finds the following two solutions, in which $\tau_i$ denote the decay times of the isotopes.

\[
N_{Ni} = N_{Ni,0} e^{-\left(t/\tau_{Ni}\right)}
\]  
\tag{34}
\]

\[
N_{Co} = N_{Ni,0} \frac{\tau_{Co}}{\tau_{Co} - \tau_{Ni}} \left(e^{-\left(t/\tau_{Co}\right)} - e^{-\left(t/\tau_{Ni}\right)}\right)
\]  
\tag{35}
\]

The initial amount of $^{56}\text{Ni}$ can be determined from the ratio of $^{56}\text{Ni}$ in the star and the mass of the nucleons. The only assumption for making a model would thus be a reasonable estimate for the ratio of Nickel mass to star mass.

\[
N_{Ni,0} = \frac{M_{\odot}}{56 m_u} \frac{M_{Ni,0}}{M_{\odot}}
\]  
\tag{36}
\]

The energy deposition, or energy per time unit can be found by multiplying the decays per time unit with their respective energies:

\[
\epsilon = N_{Ni,0} \frac{1}{\tau_{Co} - \tau_{Ni}} \left[ E_{Ni} \left(\frac{\tau_{Co}}{\tau_{Ni}} - 1\right) - E_{Co} \right] e^{-\left(t/\tau_{Ni}\right)} + E_{Co} e^{-\left(t/\tau_{Co}\right)}
\]  
\tag{37}
\]
Where \( E_i \) is the energy from one decay. Substituting the expression for \( N_{Ni,0} \), eq. 10 boils down to:

\[
\epsilon = \frac{M_\odot}{56 m_u} \frac{M_{Ni,0}}{M_\odot} \frac{1}{\tau_{Co} - \tau_{Ni}} \left[ E_{Ni} \left( \frac{\tau_{Co}}{\tau_{Ni}} - 1 \right) - E_{Co} \right] e^{-(t/\tau_{Ni})} + E_{Co} e^{-(t/\tau_{Co})} \tag{38}
\]

This expression could be added to the total luminosity. Input variables would be the star mass and an assumption for the percentage Nickel. Reaction energies and decay times are simply constants. One can check the order of magnitude of \( \epsilon \) by filling in the respective decay energies and reaction times scales. Filling in the values given in [19], one finds:

\[
\epsilon = (6.45 \cdot 10^{43} e^{-t/8} + 1.45 \cdot 10^{43} e^{-t/111}) \frac{M_{Ni,0}}{M_\odot} \left[ \text{ergs}^{-1} \right] \tag{39}
\]

One can integrate this expression over time to find the total emitted energy:

\[
E_{\text{emitted}} = (1.885 \cdot 10^{50}) \frac{M_{Ni,0}}{M_\odot} \left[ \text{erg} \right] \tag{40}
\]

### 2.4 Opacity

In order to create a realistic light curve, one has to consider the mean free path of the photons that try to escape from the shells. The outermost part of the star, of which the mean free path of the photon is large enough for it to escape the star and add to the luminosity, is called the photosphere. In order to find the mean free path and ultimately the photosphere and total luminosity, one needs to find an expression for the opacity first. The opacity in a star can be calculated by approximated analytical formulae and fitted approximations of the known tabulated opacities. In this section the different components of the total opacity will be discussed following the method of Burrows[6]. As discussed earlier in the paper, the existence of luminosity is not taken into account in the model. The opacity will therefore also not be used in the numerical model itself. However, a small program was written to calculate opacities, and although it is not used in the main model, a discussion of its results is included in section 7.2.

The first component of the opacity, which is dominant at high temperatures and densities, is opacity due to electron scattering. This opacity can be described by equation 41, where \( X \) is the mass fraction of hydrogen, \( \rho \) the density and \( T \) the temperature[6]:

\[
\kappa_e = 0.02(1 + X) \left( 1 + 2.7 \cdot 10^8 \frac{K^2 m^2}{kg} \rho \frac{T^2}{T_r^2} \right)^{-1} \left( 1 + \left( \frac{T}{4.5 \cdot 10^8[K]} \right)^{0.86} \right)^{-1} \frac{m^2}{kg}. \tag{41}
\]

The second contribution to the total opacity is Kramers’ opacity, which is the opacity due to free-free, bound-free and bound-bound free electronic transitions\[^3\]. This contribution can be approximated by Kramers’ formula, given by equation 42[6]. Here, \( Z \) is the mass fraction of metals inside the considered material.

\[
\kappa_K \approx 4 \cdot 10^{21} \left[ \frac{m^5 K^{3.5}}{kg^2} \right] (1 + X)(Z + 0.001) \frac{\rho}{T^{3.5}} \frac{m^2}{kg}. \tag{42}
\]

The third component in the total opacity is the opacity of negatively charged hydrogen ions, also called hydrogen anions. These anions are very prevalent in the atmosphere of stars and their contribution to the total opacity is given by equation 43[6]:

\[
\kappa_{H^-} \approx 1.1 \cdot 10^{-27.5} \left[ \frac{m^{3.5}}{kg^{1.5} K^{7.7}} \right] Z^{0.5} \rho^{0.5} T^{7.7} \frac{m^2}{kg}. \tag{43}
\]

\[^3\]All refer to the absorption of a photon by an electron. In the free-free case, a free electron orbiting an ion moves into a higher energy orbit. Bound-free is comparable to bound-bound, the only difference is that a bound electron is ionized in the first case, and simply excited in the latter. [19]
In the low temperature regime, molecules dominate the opacity. This results in equation 44:

$$\kappa_m \approx \frac{Z}{100} \left[ \frac{m^2}{kg} \right]$$  \hspace{1cm} (44)

The total radiative opacity can be approximated by equation 45:

$$\kappa_{rad} \approx \kappa_m + \left( \kappa_{H^-} + (\kappa_e + \kappa_K)^{-1} \right)^{-1}.$$  \hspace{1cm} (45)

When high temperatures are reached, also and electron conductivity becomes important, which can be approximated by equation 46:

$$\kappa_{cond} \approx 2.6 \cdot 10^{-2} \left[ \frac{kg}{K^2 m^4} \right] \langle Z^* \rangle \frac{T^2}{\rho^2} \left( 1 + \left( \frac{\rho}{2 \cdot 10^9} \right)^{2/3} \right) \left[ \frac{m^2}{kg} \right]$$  \hspace{1cm} (46)

The total opacity can then be expressed as:

$$\kappa = (\kappa_{rad} + \kappa_{cond})^{-1}$$  \hspace{1cm} (47)

### 2.5 Blast Waves

To trigger the supernova explosion, one might choose to look at the critical pressure/density at which the degeneracy pressure from electrons is not great enough anymore to counteract the gravitational pressure. The resulting blast wave can then be approximated by using the Sedov-Taylor solution for a strong shock with energy $E$ that is released inside a spherical fluid with density $\rho_0$. The energy can be calculated from the gravitational potential energy that is stored in the ejected mass. Shock waves are, like opacity and luminosity, not built into the model. This section contains a theoretical discussion of blast waves and the way they might be used for simulating supernova explosions.

The Sedov-Taylor solution describes a free expansion that is valid at longer timescales, and is generally used for describing supernova remnants. The solution can be found through dimension analysis. The goal in the dimension analysis method is to find a blast wave radius of dimension $[L]$ (length), given the energy of the explosion $E$, the density $\rho_0$, and the elapsed time. Combining these variables will yield the following solution for the radius of a rapidly moving shock front:

$$R(t) = \eta_s \left( \frac{E^{1/5}}{\rho_0} \right)$$  \hspace{1cm} (48)

This result will also give a starting point for determining the propagation velocity of the shock, $v_s$:

$$v_s = \frac{dR(t)}{dt} = \frac{2}{5t} \eta_s \left( \frac{E^{1/5}}{\rho_0} \right) = \frac{2}{5t} \frac{R(t)}{t}$$  \hspace{1cm} (49)

These Sedov-Taylor solutions are self-similar in the sense that for a larger value of $t$, the shape of the shock front will be the same as at lower values for $t$. I.e. one can regard the shock wave as a pressure-peak that travels through the medium and declines in amplitude as time passes. The self-similarity is captured in the scaling parameter $\eta_s$. The scaling parameter consists of various properties of the medium (including sound speed), and has to be found through the boundary conditions of the expansion phase.

According to Cook, a solution for $\eta_s$ can be found when a simple point explosion occurs in a uniform sphere of air:

$$R(t) = \left( \frac{75E(\gamma - 1)t^2}{8\pi\rho_0} \right)^{1/5}$$  \hspace{1cm} (50)

However, in practical applications (read: supernovae) determining $\eta_s$ from the given physical system proves to be difficult, according to [25] and [10]. The general method for finding $\eta_s$ consists of the following steps, as once carried out by Sedov himself [25] [10].
• Solve for pressure $P(r,t), \rho(r,t)$ and $v(r,t)$ of the swept-up material, respectively. The exact procedure of solving is given in [10], the results are displayed in the three equations below. The three resulting equations will all have to showcase the self-similar behavior that the original solution also possesses. This means that each equation requires the introduction of a self-similar scaling function $A(\eta)$, $B(\eta)$ or $C(\eta)$, respectively.

$$\rho(r,t) = \rho_{ps} A(\eta)$$  \hspace{1cm} (51)

With $\rho_{ps}$ the post-shock density of the fluid, which is given by $\frac{\gamma + 1}{\gamma - 1} \rho_0$.

$$P(r,t) = P_{ps} \left(\frac{\eta}{\eta_s}\right)^2 B(\eta)$$  \hspace{1cm} (52)

With $P_{ps}$ the post-shock pressure in the fluid.

$$v(r,t) = \frac{2}{\gamma + 1} v_s$$  \hspace{1cm} (53)

• Once the three equations are found, the respective variables can be substituted in the Euler equations for conservation of mass, momentum and energy. The used Euler equations are the following:

$$\frac{\partial \rho}{\partial t} + \frac{1}{r^2} \frac{\partial}{\partial r} (r^2 \rho v) = 0$$ \hspace{1cm} (54)

$$\frac{\partial v}{\partial t} + v \frac{\partial v}{\partial r} = - \frac{\partial P}{\partial r}$$ \hspace{1cm} (55)

$$\frac{\partial}{\partial t} \left[ \rho (\epsilon + \frac{1}{2} v^2) + \frac{1}{r^2} \frac{\partial}{\partial r} \left[ r^2 \rho v (\epsilon + \frac{P}{\rho} + \frac{1}{2} v^2) \right] \right] = 0$$ \hspace{1cm} (56)

with $\epsilon = P/[(\gamma - 1)\rho]$

• After substitution, three coupled first order non-linear differential equations are found. After a change of variables, from $(r,t)$ to $(\eta,t)$, the differential equations can all be solved numerically. Using the total explosion energy as a boundary condition, one can guess a value for $\eta_s$, calculate the functions $A(\eta)$, $B(\eta)$ and $C(\eta)$, and iterate until the boundary condition is met through the used $\eta_s$-value.

This general method was proposed and carried out by Sedov himself. Considering how the program (see section earth atmosphere) showcases shockwave behavior in some cases, one could choose to incorporate the Sedov-Taylor solution and the accompanying numerical methods. Because of time- and skill constraints, it was decided to omit this solution from the supernova simulations.
3 Creating a numerical model

3.1 Basic assumptions

Before one can start creating a numerical model, one has to make a number of assumptions and approximations regarding the physical system. Often, physical systems are too complex to be modeled in their entirety, and some simplifications are necessary. For this supernova model, three main assumptions were made. These will be detailed in the following section.

3.1.1 Spherical symmetry

The first main assumption is spherical symmetry. Spherical symmetry means that the star’s properties are assumed to have no angular dependence. In other words, the star is regarded as a perfect sphere. Each radial direction is therefore considered equivalent. This has the effect of turning a three-dimensional problem into a one-dimensional one, since only one radial direction suffices for describing the entire star. A further consequence of this assumption is that the star cannot be allowed to rotate. Rotation about an axis would, namely, give a preferential direction to the system, which would break spherical symmetry. Thus, the star is considered to be spherically symmetrical, and therefore only one radial direction needs to be evaluated.

3.1.2 Gas composition

The second assumption involves the composition of the stellar matter. It is assumed that the star consists of a mixture of photon gas and ideal classical gas. The pressure of the stellar gas given by equation 8, and the internal energy is given by 9, as can be found in section 2.1.

In both equations, the photon gas terms increase faster with higher temperatures than the terms from the classical gas. This is because the photon gas terms are proportional to $T^4$, whereas the classical gas terms are linear with temperature. This suggests that for high temperatures, the photon gas will become dominant over the classical gas. It is therefore assumed that at the high temperatures within stars, the classical gas terms will be much smaller compared with the photon gas terms.

3.1.3 Homogeneous distributions

The final major assumption is that shell matter is regarded to be spread homogeneously throughout each respective shell. Because of this, the same value for the density, pressure, and temperature hold at each radial position inside a certain shell. In the model, these quantities are evaluated in the middle of shells, and these values are then applied to all respective shell matter. This assumption is valid as long as the shell thickness is small.

3.2 Discretizations

3.2.1 Stellar shells

The first way in which the system needs to be discretized, is in its space coordinates. Quantities such as temperature and density cannot be evaluated at every value for the radius. Instead, the stellar material needs to be divided into a number of discrete intervals at which the physical quantities can be calculated. In this model, the star is divided into a number of spherical shells. These shells contain stellar material with a certain density, temperature, pressure, and internal energy (see figure 3.2.2). The matter within a shell is regarded as being isolated from the matter within other shells. There is therefore no convection of stellar material between shells. The dynamics of the star can be simulated by studying the movement of these spherical shells. By looking at the way in which the radial position of each shell changes due to gravity and pressure, gives an understanding of the dynamics of the star as a whole. This necessitates a more precise definition of shell radii and the matter within these shells.
There are a number of different ways in which one can define the shell radius with respect to the matter contained inside the shell. The most straightforward way of these is taking the radius to be the position of either the outer or the inner surface of the shell. Both of these definitions are equivalent, but lead to different problems at the star’s boundary. For instance, choosing the inner surface of a shell to denote the shell radius means that a shell’s matter is located above the shell radius. The first radius then denotes the surface of the neutron star core. This radius has to be kept fixed because the core should not change in size. Located above this radius is the first shell of stellar matter. At the boundary of this shell lies the second radius, and above this radius the second shell of stellar matter. This numbering continues without problems until the star’s outer layer is reached. The final stellar radius should, namely, denote the boundary of the star’s atmosphere, but because stellar radii are located at the inside of shells, there is still some matter above the outermost radius. This radius therefore does not, in fact, denote the radius of the star’s atmosphere, since there is still mass outside of it.

The other definition of stellar radii, i.e. letting the outer surface of a shell denote the radius, leads to a similar problem. Using this definition, the outermost radius does denote the radius of the star’s atmosphere, as the outermost shell of matter lies beneath it and no further mass lies above it. There is therefore no problem with the outer radius here. Instead, one encounters a problem for the innermost radius. The bottom radius should denote the surface of the neutron star core, however, by choosing the radius to lie at the outer surface of a shell, there is always a shell of stellar material below the lowest radius. Therefore, the innermost radius cannot denote the surface of the neutron star.

Thus, both definitions for the stellar radii have inconsistencies. Either of these can be used, but in order to use either shell discretization correctly, a solution to these inconsistencies needs to be found. The numerical model of this paper defines the stellar radii to lie at the inner surfaces of shells. Therefore, a solution needs to be found for the inconsistency of the outermost stellar shell. This inconsistency can be solved by looking at what the outermost shell would represent physically. The outer layer of a star contains very little matter. The temperature, pressure, and density of that matter is consequently very low. The outer shell is therefore regarded as near vacuum. It is assumed that the quantities mentioned earlier do not change for the outermost shell, as they were very low to begin with.

3.2.2 Shell mass continuity

Another important thing to consider is that in this numerical model, shells are not allowed to overlap with each other. By assuming there is no convection of mass throughout the star, molecules should remain in the same shell as the one they started in, and so any overlap of shells is in violation of this assumption. The assumption that there is no convection of mass is made for two reasons. The first being that it greatly simplifies the model because one does not need to keep track of quantities like density gradients and mass flux, which would be necessary to calculate for a model featuring convection. The second reason for this assumption is that it gives a clear meaning to the concept of shell radii. By keeping the mass inside a shell fixed, each shell radius always corresponds to a certain amount of matter. In a model with convection, however, shells are merely artificial divisions of a stellar atmosphere, with no particular physical quantity ascribed to them. This is not the case for a model without convection, in which each shell will always have the same mass ascribed to it. The shell may change in volume, but the same mass remains inside of it. Therefore, the assumption that there is no convection within the star allows for a simpler numerical model and gives each shell a precise physical interpretation.

To summarize, in this model the star’s atmosphere is discretized in a number of spherical shells, each containing a certain amount of stellar matter. The model will calculate how quantities such as temperature, density, and pressure change within these shells, and use these changes to determine the dynamics of the radii of each shell (see section 4.1).
Figure 1: This diagram (not to scale) illustrates the discretization of a stellar atmosphere into spherical shells. The shaded center of the diagram denotes the neutron star core. The dashed circle above radius $R_N$ represents the mass that is located above that shell.
3.2.3 Time discretization

The next quantity requiring discretization is time. Since the numerical model will start from a certain initial situation and calculate how the system evolves over time, it is necessary to choose a value for the size of the time steps, \( \Delta t \). On the one hand, a small value for \( \Delta t \) will ensure accurate simulation, especially when physical quantities are changing rapidly over small periods of time. However, using a small time step will also result in more calculations being made per unit simulated time. A larger time step size is therefore more suitable when one wants to simulate longer periods of time. The drawback of using larger time steps is that it results in less accurate calculations. This can become a major problem when shell radii are very close to one another. As described in the previous section, stellar shells are not allowed to overlap in this model. Therefore, during a convergence of shells, it becomes important that the rapid changes in temperature and density that accompany such a convergence (see section 2.1) are accurately calculated and used to adapt the velocity and acceleration of shell radii. When the time scale on which these temperature and density spikes operate is smaller than the time step, this will usually result in the overlap of shells. In order to prevent this from happening, the time step needs to be suitably small during shell convergences.

3.2.4 Variable time step

The most ideal way to discretize time would be in such a way that one has both a fast simulation time and a high numerical accuracy when necessary. This cannot be done by using a fixed time step. Instead, the value for \( \Delta t \) must be adjusted during the simulation. It should become small whenever the simulation is at risk of becoming unstable, and become larger when the system is stable. For the supernova model, high numerical accuracy becomes necessary whenever shells are beginning to converge. When shell radii are far apart from each other, overlap of shell matter is not likely to happen, and the time step can become large. Therefore, a logical way to adapt the time step size, is by taking the distance between shells a criterion. In the model, the thickness of each shell is evaluated at each time step. The program stores the smallest shell thickness, \( \Delta r_{min} \) and the average shell thickness, \( < \Delta r > \). Then, a scaling factor, \( \delta \), is used to adjust the time step size:

\[
\Delta t_{\text{new}} = \delta \cdot \Delta t_{\text{initial}},
\]

with \( \Delta t_{\text{initial}} \) the base value of the time step as inputted by the user, and where \( \delta \) is defined by:

\[
\delta = \left( \frac{\Delta r_{min}}{< \Delta r >} \right)^n,
\]

with \( n \) an integer. After performing some test runs of the model with different values for \( n \), it was decided to set it equal to two. This value was deemed the best compromise between responsiveness to shell convergence (for which a larger value of \( n \) is needed) and simulation speed (for which the time step size should not become too low). At each time step, the value of the scaling factor is determined, and compared with a minimal value. The scaling factor is only applied to the time step size when it is larger than this minimal value. This is done to prevent the time step size from becoming unpractically small. If no minimal value would have been defined, the time step size would converge to zero as shells moved ever closer to each other, causing the model to stop working properly.

In the case of the red giant-like element layer division, the shell thicknesses differ in size throughout the stellar atmosphere. Because of this, the minimal shell thickness will, even in stable circumstances, be small due to the lower lying layers having much smaller shell spacing than the upper layers. Therefore, the time scaling factor is defined in a slightly different way for this situation. When using the red giant-like element layer division, the stellar atmosphere is divided into two parts. The first of these contains the layers with elements up until neon, and the second one contains the layers from oxygen up to the outer boundary of the star. The model then evaluates a scaling factor using (58) for both parts of the atmosphere (using \( n = 2 \) as well). It then compares these two values, and the smallest one is used for adjusting the time step size.
4 Numerical equations

4.1 Discretization of the physical system

As described in section 3.2.1 a stellar radius denotes the position of the bottom surface of a stellar shell. The radial position of the outer surface of a shell is therefore given by the radius of the shell above it. The \( i \)’th shell of stellar matter therefore lies between the \( i \)’th and \( i + 1 \)’th radius. Because of this, the temperature, pressure, and density of a shell are determined in the middle of two shells. More explicitly:

\[
r_{m,i} = \frac{1}{2}(r_i + r_{i+1}),
\]

where the subscript \( m, i \) denotes that the radius belongs to the average position of the stellar matter inside shell \( i \). Special consideration has to be given to the outermost shell. As described in section 3.2.1 the outermost shell of matter does not have a well-defined outer surface. Because of this, it was decided to set \( r_{m,final} \) equal to \( r_{final} + \frac{1}{2} \Delta r \). This is an arbitrary definition and it could have been phrased in a number of different ways. This definition was deemed the most straightforward and is therefore used in the model. As described in section 3.1.3 the density, pressure, and temperature are considered homogeneous throughout the matter inside a shell. Because of this, these quantities will be determined in the model at the center of stellar shells. It is then assumed that the values these quantities have at the center hold for all matter inside a shell. This approximation is valid for small shell thicknesses. So, the density, pressure, and temperature are evaluated at the center of shells, that is, at each \( r_{m,i} \). In the following discussion, quantities evaluated at the centers of shells will have an \( m \) included in the subscript.

The next step in discretizing the system is going from quantities defined in between shells radii to the acceleration and movement of shell radii themselves. As derived in section 2 the acceleration of a spherical shell due to gravity and pressure is given by:

\[
a = -\frac{G \cdot m_{encl}}{r^2} - \frac{1}{\rho} \frac{dP}{dr}.
\]

Before this equation can be used for determining the acceleration of shell radii, it is necessary to define some of its terms in more detail. The first of these is the mass enclosed by a certain radius, \( m_{encl} \), is equal to the mass of all shells lying below it. Since matter is not allowed to flow between shells in this model (see section 3.2.1), the mass inside shells remains constant, and therefore \( m_{encl} \) is a constant as well. The second term needing clarification is the density. As described earlier, densities are evaluated at the center of shells. What is needed, therefore, is the density at a shell radius, instead of at the shell center. The density at a radius is approximated by taking the average between the densities evaluated at the center of the surrounding stellar shells. In other words:

\[
\rho_i = \frac{1}{2}(\rho_{m,i-1} + \rho_{m,i}).
\]

The final term needing clarification is the pressure gradient, \( \frac{dP}{dr} \). This derivative has to be approximated numerically, which has been done in the following way:

\[
\frac{dP}{dr} = \frac{P_{m,i} - P_{m,i-1}}{r_{m,i} - r_{m,i-1}},
\]

where the two radial positions are determined by equation 59. This model therefore takes the pressures at the centers of two adjacent shells, and divides this by their difference in radial position. This gives an approximation for the pressure gradient over the shell radius, because this radius is situated in between the two shells.

Combining the above three discretized terms gives the following result for the acceleration of shell radii:

\[
a_i = -\frac{G \cdot m_{encl,i}}{r_i^2} - \frac{1}{2(\rho_{m,i-1} + \rho_{m,i})} \frac{P_{m,i} - P_{m,i-1}}{r_{m,i} - r_{m,i-1}}.
\]
Now that the acceleration of each shell is defined, one can determine how this acceleration affects the velocity and position of each shell. The change in velocity can be derived easily. Since the acceleration is merely the time derivative of the velocity, the velocity can be approximated by multiplying the instantaneous acceleration \( a \) by the time step size \( \Delta t \):

\[
v_i(t + \Delta t) = v_i(t) + a_i(t) \Delta t. \tag{64}
\]

The expression for the change in radius can be found in a comparable way. The acceleration is, of course, the second time derivative of the radial position, \( r \). Multiplying the velocity with the time step size \( \Delta t \) gives:

\[
r_i(t + \Delta t) = r_i(t) + v_i(t) \Delta t + a_i(t) \Delta t^2. \tag{65}
\]

Lastly, the equation for the change in temperature (equation 17) needs to be discretized. This discretization is fairly straightforward, and yields:

\[
\Delta T = - \left( \frac{\rho_i k T_{\text{old}}}{\mu} + \frac{a T_{\text{old}}^4}{3} + a T_{\text{old}}^4 \right) \frac{\Delta V}{2} + 4 \alpha \frac{4}{3} (r_{i+1}^3 - r_i^3) T_{\text{old}}^3. \tag{66}
\]

In which \( \Delta V \) is the volume difference of the shell between two time steps:

\[
\Delta V = \frac{4\pi}{3} \left( r_{i+1}(t + \Delta t)^3 - r_i(t + \Delta t)^3 - (r_{i+1}(t)^3 - r_i(t)^3) \right). \tag{67}
\]

Every value for the temperature will then be calculated by adding \( \Delta T \) to the value of \( T \) from the time step:

\[
T_{\text{new}} = T_{\text{old}} + \Delta T. \tag{68}
\]
5 General program structure

The program, which simulates either an earth-like atmosphere or a supernova consists roughly of four parts:

- Block of functions
- Initial conditions
- Time-loop
- Data collection

The first part of the program consists of definitions. In the previous sections, various discretizations of variables were introduced. Variables that are necessary for different parts of the code were defined as functions, to make sure that the program would stay clearer. Below is an example of a pressure function, which asks for previously defined variables (such as constants \( a \) and \( k_B \) or temperature \( T \)) and returns a value. Instead of using the discretized pressure formula repetitively, one can simply recall the function at various places in the program to perform calculations.

```python
def pressure_gen(T, rho, molecularmass):
    return a/3*T**4 + rho*boltz*T/molecularmass
```

The second part of the code sets the initial conditions, which is done by using an interactive interface. The program asks the user for multiple inputs, which consist of shell numbers, simulation types and runtimes. The input steers various conditional statements, which activate different parts of the initial conditions. According to the given input, all starting parameters are then calculated, such that the program can proceed to the time-loop.

In the time loop, the time evolution of the different quantities in the star or atmosphere is calculated. This is done by applying the pre-defined functions repetitively to the set initial conditions. When the new quantities are calculated, these are calculated and stored in an array. Everytime the loop is completed, one row of data is added to the array. An overview of the data storage can be found in figure 5. Aside from the array that consists of physical quantities, a time-vector is also stored.

![Figure 2: Stacking of data rows in the array](image)

Ultimately, the data from time loop is plotted. For plotting, the library `matplotlib` was used. The color of the lines in the plot scales from dark to light, according the height of the respective shell. Darker colors accompany the more inwardly situated shells and vice versa.
6 Test on Earth’s atmosphere

Before simulations on supernovae can be done, it is necessary to know if the dynamics of the numerical model work correctly. The best way to test this, is by comparing the numerical results with (analytically) known solutions. Because the supernova model largely consists of gas dynamics, it can be tested with simulations of the Earth’s atmosphere, which is a much easier physical system. One can easily adjust initial parameters to match those of the Earth instead of those inside an unstable star. This section will detail how the specific parameters and initial conditions for the Earth-like atmosphere were arrived at.

In the previous sections, the discussion focused on the physics of stars and supernovae. Most of the equations and physics can be translated straightforwardly to the situation of an Earth-like atmosphere. First of all, the spherical shell discretization can be used for modelling the Earth’s atmosphere in much the same way it is used in case of the star. The two main differences in this discretization are that instead of containing stellar matter, the shells contain nitrogen molecules, and that the shells do not enclose a neutron star core but that they enclose the Earth. Secondly, the assumption that the shell matter consists of a mixture of a photon gas and an ideal gas can also be used for describing the Earth’s atmosphere. This is because the photon gas term scale with the temperature, which is much lower in Earth atmosphere simulations. At room temperature (293K), the photon gas pressure is $1.86 \times 10^{-6}$ Pa, which is negligible. Therefore, the photon gas can remain incorporated in the model when evaluating the Earth’s atmosphere. Lastly, the assumption that the outermost shell is nearly a vacuum holds for the Earth as well, so no quantities will be updated for that shell in this case either.

6.1 Initial conditions

Initially, the temperatures of each shell are set equal to the value of 293 K. The initial density distribution will be chosen in such a way that this constant initial temperature distribution yields an equilibrium state. The derivation for this equilibrium situation of an Earth-like atmosphere situation starts, as with the supernova, with hydrostatic equilibrium:

$$\frac{\partial p}{\partial r} + g\rho = 0$$

(69)

where $g$ is the average gravitational acceleration on Earth. This yields the same expression for the acceleration of shell radii as in the supernova situation described in section 4.1. Consequently, the same equations hold for the velocity and change in radius in both situations.

In appendix I, the equilibrium pressure and density distributions are derived. Of these, only the density distribution will be used directly. The pressure distribution as derived in the appendix will not be used explicitly in the model. Instead, the pressure will be calculated through the general equation $\rho = \frac{4}{3}VT^4 + \frac{\rho kT}{\mu}$. This was done to prevent large changes in pressure between the first and second time step: because the pressure is calculated by using the general equation within the time loop, using another method of calculation for the first time step would cause the pressure to greatly change in value between the first and second time step. Therefore, the initial pressures are calculated by using the general equation. These values should not be too different from the pressure distribution from the appendix because the photon gas term is negligible at Earth-like temperatures. So, the pressure distribution is not used, but the density distribution will be used, and is given by:

$$\rho(r) = \rho(0) \exp\left(-\frac{r}{H}\right).$$

(70)

In this formula, the density $\rho(0)$ was set equal to 1.251 kg/m$^3$, which is the density of air at standard temperature and pressure.

Based on these initial conditions it is possible to make predictions of how the simulation will evolve. Because the density distribution has been defined in such a way that the initial temperature distribution yields hydrostatic equilibrium, it is expected that the shell radii will not change position, since the shells should be at their equilibrium position already. Instead, the shells will likely oscillate somewhat about their
starting position. This oscillation is caused by the fact that the starting parameters are likely not exactly equal to the equilibrium situation, and therefore the system will move towards it, overshoot a small amount, and start oscillating. The numerical model can thus easily be checked by comparing its behavior to the predicted outcome for constant- and variable temperature solutions.

<table>
<thead>
<tr>
<th>$R_{\text{earth}}$</th>
<th>6400 km</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\text{earth}}$</td>
<td>5.9721986·10^{24}$kg$</td>
</tr>
<tr>
<td>$R_{\text{atm}}$</td>
<td>50000 m</td>
</tr>
<tr>
<td>$\Delta r$ (10 shells)</td>
<td>5000 m</td>
</tr>
<tr>
<td>$\Delta r$ (100 shells)</td>
<td>500 m</td>
</tr>
</tbody>
</table>

Table 3: Parameters used for the simulation of the Earth-like atmosphere.

6.2 Results

6.2.1 Equilibrium state

The first simulation that will be discussed here is a 15 minute simulation with the aforementioned conditions, using ten shells, a basis time step of 1 second, time step scaling, and no damping. As can be seen in figure 3, the shell radii oscillate around an initial position. This suggests that the initial situation corresponds to an equilibrium state. A closer look at the movement of one shell radius (see figure 4) confirms that the radii oscillate about their starting value. Figure 5 shows that the temperatures of shells oscillate greatly about the starting value of 293 K. The horizontal line one can see in this figure corresponds to the temperature of the outer shell, which stays constant by definition. The final quantities of interest here are the pressure gradients. The time evolution of these gradients are shown in figure 6. One can see that the gradients oscillate around their initial value as well, which is another indicator of equilibrium. Another interesting quantity to consider is the temperature of each shell. A zoomed-in view of this temperature evolution can be seen in figure 7. One can see from this graph that the temperatures oscillate about their initial equilibrium value, which is the behavior one would expect from stable states.
Figure 3: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 293K. It shows how the radii of shells evolve over time.

Figure 4: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 293K. It shows a close-up of a section of the evolution of the fourth shell’s radius.
Figure 5: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 293K. It depicts the evolution of the pressure gradient across each radius in Pa/m. The darker the shade of grey, the lower the shell lies in the atmosphere.

Figure 6: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 293K. This graph shows a close up of the temperature evolution of all shells. The darker the shade of grey, the lower the shell lies in the atmosphere. The horizontal, unmoving line represents the outermost shell, the temperature of which is kept fixed.
For this simulation, the density distribution for which the atmosphere is supposed to be stable was derived using a constant initial temperature distribution. This temperature was not assigned a specific value, so the atmosphere should be stable for any constant initial temperature. To test this, another simulation was run with a constant initial temperature of 393K. As seen in figure 7, the atmosphere remains constant at this temperature as well. Another important quantity to consider is the energy of each shell. This energy is the sum of a shell’s internal, kinetic, and gravitational energy, and is calculated using equation 12. The time evolutions of the energies are shown in figure 8. One can see that the energy of each shell remains constant over time. Since the shells do not expand or shrink, this is in accordance with theory.

In deriving the equilibrium density distribution, the only assumption was that the temperature of the atmosphere had the same value everywhere. No specific value was given, however, and therefore the atmosphere should remain stable for any given initial temperature. This was tested, and it was found that the simulation remained stable for temperatures as high as 10000K. At higher temperatures, the program did become unstable. This is likely cause by numerical instabilities due to the fact that shells move faster at higher temperature.

It is odd that the atmosphere is stable for temperatures around 10000K. At high temperatures, the photon gas terms start to become dominant. However, in the derivation of the stable density distribution (see the appendix), only the ideal gas pressure was taken into account. The photon gas was neglected during this derivation. It is therefore strange that the simulation remains stable when the photon gas terms become important.

So, the results from these simulation confirm that the derived pressure distribution indeed corresponds with a stable atmosphere, and that the model is capable of maintaining stability.
Figure 7: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 393K. It shows the time evolution of each spherical shell.
Figure 8: Graph showing the results of an Earth-like atmosphere simulation with an initial temperature of 393K. This graph shows how the energy of each shell evolves over time.
6.2.2 Non-equilibrium state

The next thing that needs to be tested is whether the model can converge to an equilibrium when it starts off in an unstable state. Therefore, a simulation was run where the innermost shell was given a temperature of 350K while keeping the other shells at 293K. One would expect the lowest shell to expand, thereby lowering its temperature. The expansion of the innermost shell causes each subsequent shell to contract, which will raise their temperature. This should cause the outlying shells to expand as well. This expanding and contracting is expected to continue until a new equilibrium situation is reached. In this equilibrium, the shells should have a larger volume than in the initial situation. Because the shells are expected to move with respect to each other, overlap is more likely. Therefore, a lower basis time step of 0.1 seconds was chosen for this simulation.

The results are depicted in figure 9. It shows that the shells do indeed expand one after the other. However, another interesting phenomenon is visible here, which will be described qualitatively in this section. There appears to be a kind of shockwave travelling from the innermost shell, up to the edge of the atmosphere (see figure 10). Once the shockwave reaches the edge of the atmosphere, the outermost layers burst upwards but quickly come back down and move to a position that is lower than their initial one (see figure 11). This shrinking appears to send a shockwave downwards which moves towards the surface of the Earth. Once it reaches the bottom shell, the shock 'reflects' back up and again seems to cause the outer shells to burst upward. After this, the outer shells contract again and appear to send another shockwave downwards. That these movements are indeed caused by shockwaves can be seen from figure 12. This graph shows how the pressures inside the shells change during the passage of the first shockwave (from $t = 0$ to 260 s). One can see how the pressure of the innermost shell - which is the highest line in the graph - peaks and then drops, upon which the pressure of the second shell starts to spike. The pressure of this shell then drops like the first one, after which the third shell’s pressure peaks. This goes on until the lowest shells, but these shells are not clearly visible in this figure. One can see how a kind of pressure wave travels throughout each shell. The pressure peaks become lower as the wave travels further. After the pressure wave has passed a shell, the shells’ pressure oscillates for a time and then settles to a constant value. It remains at this value until the reflected wave travels back down. When the wavefront reaches the shell, its pressure drops and then peaks again. Once the second wavefront has passed and its reflection travels upwards again, the pressures oscillate much more wildly. The back and forth reflections of shockwaves continue as the simulation goes on. The movements of the shell radii become more oscillatory and chaotic as time goes on, which can be seen clearly in figure 11.

29
Figure 9: Graph showing the results of an Earth-like atmosphere simulation using 100 shells with the first shell at 350K. It shows how the shell radii evolve over time.
Figure 10: Graph showing the results of an Earth-like atmosphere simulation using 100 shells with the first shell at 350K. It shows a close up of the time evolution of the shell radii during the first shockwave.
Figure 11: Graph showing the results of an Earth-like atmosphere simulation using 100 shells with the first shell at 350K. It shows the time evolution of the outermost shells.
Figure 12: Graph showing the results of an Earth-like atmosphere simulation using 100 shells. It shows a close up of the pressure evolution during the first shockwave.
Pressure waves travel through a medium at the speed of sound for that medium. Therefore, the shockwaves in these results should propagate at the speed of sound in air. From figure 10 it can be seen that the shockwave travels through the 50000 meters of atmosphere in about 117 seconds. The propagation velocity of this shockwave is then $v_{\text{wave}} = \frac{50000\text{m}}{117\text{s}} = 427\text{m/s}$. The speed of sound in air of 20°C is about 343 m/s, which is about 20% smaller than the value in the figure. Given that the density and temperature of the simulated gas are different from the Earth’s, and that the pressure wave can only move upwards in the model, the found discrepancy is not too large. The fact that the model is able to simulate shockwaves was quite a surprise. Before the model is discussed any further, a theoretical discussion of shockwaves and sound velocities is necessary, and is described in the following section.

6.2.3 Theoretical soundwave speeds

Sound waves are longitudinal waves that propagate in various media. In this section, sound waves in gases are considered. Sound waves can be classified in two ways, depending on the thermal properties of the gas. These are isothermal- and adiabatic sound waves. Before considering the difference between the two types, one needs to consider the general equation for the speed of sound which is given by the following equation, in which $B$ is the bulk modulus and $\rho$ the density:

$$v_s = \sqrt{\frac{B}{\rho}} \quad (71)$$

The general expression for the bulk modulus is given by:

$$B = -V \frac{\partial p}{\partial V} = \frac{\partial p}{\partial \rho} \quad (72)$$

**Isothermal sound waves** Considering waves under isothermal conditions (i.e. assuming that the temperature is constant), one obtains the following expression for the bulk modulus:

$$B_T = -\rho \left( \frac{\partial p}{\partial \rho} \right)_T = p \quad (73)$$

Using the isothermal Bulk modulus means that the speed of sound becomes:

$$v_s = \sqrt{\frac{p}{\rho}} \quad (74)$$

**Adiabatic sound waves (not entirely complete)** If, on the other hand, one looks at waves under adiabatic conditions ($pV^\gamma = \text{constant}$), it can be seen that the following holds for the relation between $\gamma$ and $\frac{dp}{p}$:

$$\frac{dp}{p} = -\gamma \frac{dV}{V} \quad (75)$$

One can now define the Bulk modulus under adiabatic conditions:

$$B_S = -V \left( \frac{\partial p}{\partial V} \right)_S = \gamma p \quad (76)$$

In the same fashion as for the isothermal sound velocity, the adiabatic sound velocity is found to be:

$$v_s = \sqrt{\frac{\gamma p}{\rho}} \quad (77)$$

Now, both type of sound waves can be applied to three relevant gas regimes, consisting of either an ideal or a photon gas, or a mixture of both, respectively. Regarding the simulations of the earth-like atmosphere
it is assumed that shells are adiabatic, which means that only the adiabatic sound speed for these three regimes will be discussed.

First of all, the combination of ideal and photon gas is considered. In this regime, the pressure is given by \( p = \frac{\rho k_B T}{m_w} + \frac{aT^4}{3} \). Substituting this in equation 77 gives that the speed of sound for this general situation is equal to:

\[
v_s = \sqrt{\frac{k_B T}{\mu} + \frac{aT^4}{3\rho}}
\]  
(78)

In the ideal gas regime, the pressure is much larger than the photon gas pressure. At the temperatures encountered in earth-like atmosphere simulations, the gas behaves as an ideal gas. Then, \( \frac{\rho k_B T}{m_w} \gg \frac{aT^4}{3} \), which means that in this regime the pressure is given by \( p = \frac{\rho k_B T}{m_w} \). The speed of sound can then be found by substituting the previous expression in equation 77, which gives:

\[
v_s = \sqrt{\frac{k_B T}{\mu}}.
\]  
(79)

An important feature of this sound speed expression, is that it scales with the square root of the temperature. This temperature dependence makes it possible to test whether the shockwaves found in the earth atmosphere simulations are, in fact, adiabatic pressure waves. Raising the temperature of a group of shells with a certain factor should increase the velocity of the resulting shockwave by the square root of that factor.

Lastly, the photon gas regime is covered. In this regime, the photon gas pressure dominates, and therefore \( \frac{aT^4}{3} \gg \frac{\rho k_B T}{m_w} \). This means that in this regime the pressure is given by \( p = \frac{aT^4}{3} \), substituting this in 77 gives:

\[
v_s = \sqrt{\frac{aT^4}{3\rho}}.
\]  
(80)

The velocity of sound now scales with the square of the temperature.

It is of interest to find out at which temperature the mixture of ideal and photon gas transitions from being dominated by the ideal gas to a being dominated by a photon gas. To study this transition, one could equate the two terms of the pressure equation. This yields:

\[
\frac{T}{\rho^{1/3}} = 3 \sqrt{\frac{3k_B}{a\mu}}
\]  
(81)

When this is plotted in a log \( T \) – log \( \rho \)-plot the result will be a line with slope \( \frac{1}{3} \). For the following part of this subsection of this report \( \xi \) is define as \( \xi = P_{\text{gas}}/P_{\text{total}} \), this parameter indicates what fraction the ideal gas contributes to the pressure, so \( \xi = 1 \) means that the ideal gas is dominant and a \( \xi = 0 \) means that photon gas is dominant.

Now a contour plot is made to test the speeds of sound waves in the simulation with the theoretical sound speed, see figure 13. As can been from this countour plot, it is very clear that there are 3 different regimes, one regime where the lines are constant, a regime where the the lines have a slope of \( \frac{1}{3} \) and an intermediate regime. In the following part also an example temperature is plotted and the corresponding values of \( \xi \).
Figure 13: Contour plot of the log $v_s$ in a log $T$ – log $\rho$ - plane, with the mean molecular mass of dinitrogen, where it is also assumed that dinitrogen is monoatomic, speeds above log 7 need to be ignored, because relativity is not taken in account. The violet line indicates where the different pressures are dominant, above the line the photon gas is dominant, below the line ideal gas is dominant.
Figure 14: In this graph the log $v_s$ is plotted against log $\rho$, what clearly can be seen in this graph are the 3 regimes in combination with graph 15, which shows the value of $\xi$ by different log $\rho$. In the first part it is clearly visible that there is a slope of $\frac{1}{2}$, which between -3 and 0 changes to a slope of zero.
6.2.4 Simulations on shockwaves

For the next simulation, the temperature of the lowest shell is again set to 350 K, but now the 40'th shell has a temperature of 800 K. The basic time step size is kept at 0.1 seconds. A simulation was run for 15 minutes, the results of which are depicted in figures 16 to 19.

From figure 16, it can be seen that both the first, and the 40'th shells expand due to their increased temperature. Both of these shells send out shockwaves. As in the previous simulation, the bottom shell emits a shockwave that travels upwards. The 40'th shell sends out shockwaves as well. Since this shell is not bounded by the Earth’s surface (like the bottom shell), it can expand in both directions. A zoomed in view of shell 40 can be seen in figure 17. From this figure, it can be seen that the shell expands for about an equal amount in both directions. Because the two shells immediately adjacent to shell 40 have roughly the same density, they are about equally compressible. The fact that shell 40 expands for the same amount in both directions is therefore logical. Lastly, the boundaries of shell 40 appear to rise upwards. The reason for this is not clearly known, but a possible explanation might be that some part of the shockwave from the bottom shell affects the 40'th shell before coming into contact with it.

As described in section 2.1 shells expand adiabatically. Therefore, shells should drop in temperature as they expand. The temperature of shell 40 should, for this reason, drop from its initial value of 800 K to a lower value. From figure 18, one can see that, disregarding the direct effects of passing shockwaves, the volume of shell 40 eventually becomes about constant. Its temperature should, consequently, become constant as well. In figure 18, a zoomed in view of the temperature evolution of the 40'th shell is depicted. One can see that the initial expansion of the shell corresponds to a drop in temperature, but as the shell reaches a stable volume, the temperature takes on a constant value. This result further supports the validity of the model.

Another interesting result from this simulation can be seen around $t = 30s$. At that point, the upward-travelling shockwave from the bottom shell, meets the downward-travelling wave of shell 40. The shockwaves
do not seem to interact with one another. Instead, they cross each other unhindered. This is in agreement with what one would expect based on the principle of the superposition of waves.

The final thing to note is that the shockwave tends to curve upwards as it travels up through the atmosphere. This can be seen especially well for the first upward-travelling shockwave coming from the 40'th shell. This suggests that the velocity of the shockwave becomes larger as it travels upwards through the atmosphere. The temperature evolution of each shell during the first 75 seconds is plotted in figure 19. In this plot, the lighter the shade of grey, the higher the shell corresponding to that line. From this graph, it can be seen that, as the shockwave from shell 40 progresses upwards, the temperature spikes of the shells it encounters become larger. The increase in peak temperature can be explained by the fact that higher shells have lower densities, and therefore have a lower heat capacity so when higher shells are compressed their temperature will increase more then lower shells this means that their speeds will increase according to equation 79. Which results in the upwardly curving line from figure 16.

Figure 16: Graph showing the results of an Earth-like atmosphere simulation using 100 with the temperature of the bottom shell initially set to 350 K and the 40'th shell's temperature at 800 K.
Figure 17: Graph showing a close up of the results of an Earth-like atmosphere simulation using 100 shells with the temperature of the bottom shell initially set to 350 K and the 40'th shell’s temperature at 800 K. The graph shows part of the time evolution of the shells around shell 40.

Figure 18: Graph showing part of the temperature evolution of the 40'th shell.
6.3 Conclusions of atmosphere simulations

The first simulation has shown that the model is capable of maintaining an equilibrium state when it initially begins in an equilibrium state. This demonstrates that there are no large numerical errors that introduce instabilities to the simulation. If these errors would exist, it would cause the system to drift away from equilibrium, which does not happen in the found simulations. Therefore, the model is numerically stable.

The second simulation has shown that the model is capable of simulating more complex gas dynamics, namely the phenomenon of shockwaves. The shockwaves travel with a velocity that is close to the velocity of sound. The travelling velocity of the shockwaves was found to increase with altitude. Lastly, simulations have shown that two shockwaves are able to cross each other unhindered.
7 Supernova simulation

Now that it has been demonstrated that the model can accurately simulate gas dynamics, it is finally possible to start simulating supernovae. The following section will contain a description of the initial physical conditions inside the star, the results from the simulation, and the conclusions that can be drawn from these.

7.1 Initial conditions

7.1.1 Neutron star core model

As described in section 1.3, two initial situations will be considered. First, the simplest of the two will be considered: the neutron star core surrounded by layers of stellar material. This situation can be arrived at by adjusting various parameters of the earth-like atmosphere simulations. The shell spacings and radii need to be changed because of the larger size of stellar atmospheres. Furthermore, the mass and the radius of the static ‘earth’ below the bottom shell need to be changed to fit the density and size of a neutron star. The values of all these parameters are given in table 4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_{\text{core}}$</td>
<td>12 km</td>
</tr>
<tr>
<td>$M_{\text{earth}}$</td>
<td>$1.4 \cdot M_{\odot}$</td>
</tr>
<tr>
<td>$R_{\text{atm}}$</td>
<td>500000000 m</td>
</tr>
<tr>
<td>$\Delta r$ (10 shells)</td>
<td>50000000 m</td>
</tr>
<tr>
<td>$\Delta r$ (100 shells)</td>
<td>50000000 m</td>
</tr>
</tbody>
</table>

Table 4: Parameters used for the simulation of the neutron star supernovae.

The initial density distribution is calculated by a function very similar to the one used in the earth simulations (see equation 70). For the neutron star core model, the same equation was used, but with a multiplicative factor $\alpha$ in the exponent:

$$\rho(r) = \rho(0) \exp(-\alpha \frac{r}{H}).$$  \hspace{1cm} (82)

Without this factor $\alpha$, the densities dropped off too quickly with increasing shell radius. The factor was set equal to 0.0001, which ensured a more evenly spread density distribution. The density just above the core, $\rho(0)$, was set to $1.251 \cdot 10^{15} \text{kg/m}^3$.

7.1.2 Red giant-like model

Another, more complex, initial situation in which the stellar atmosphere is like a red giant’s was considered as well. In this situation, the stellar atmosphere is divided into a number of layers of different elements. The upper layers of the atmosphere consist of hydrogen. The further down one goes, the heavier the elements become. The sizes, masses and temperatures of these element layers were discussed in section 1.3.2, and can be found in tables 1 and 2.

The first step towards implementing these initial conditions is to discretize the atmosphere properly. For the red giant atmosphere, it is not possible to divide the atmosphere into ten or a hundred equally spaced shells. This is because the hydrogen layer is much thicker than the layers of all the other elements, which would result in most shells containing only hydrogen. Only one or two shells would contain other elements. Therefore, the initial shell spacings are not uniform. The silicon and oxygen layers contain ten shells each, the neon layer consists of five shells, the carbon layer is divided into fifteen shells, the helium layer into twenty, and lastly, the hydrogen layer contains forty shells. Within each layer, the shells are equally spaced.

In section 1.3.2 only temperatures on the boundaries of element layers were discussed. However, the shells inside layers need to have a temperature value as well. Assuming that the the boundary temperature at the bottom of an element layer declines exponentially to the boundary temperature at the outside of the layer, one can determine the temperature values for each shell inside a certain layer. Take $T_1$ and $T_2$ to be
the boundary temperatures at the bottom and at the outside of an element layer, respectively. The bottom of this layer is located at a radial distance of \( r_1 \) from the core. Similarly, the layer’s outside is situated at \( r_2 \). Then, the temperature of a spherical shell located at position \( r \) can be interpolated using:

\[
T(r) = T_1 e^{-\frac{\ln \frac{r}{r_1}}{r_2/r_1}}.
\]

(83)

So, the shell temperatures for all element layers are calculated using this equation.

The final quantity that needs to be discussed is the initial density distribution. Table 1 gives the masses of the various element layers. As discussed, each of these layers is divided into a number of spherical shells. The amount of mass inside these shells is then taken to be equal to the mass of its respective element layer divided by the number of shells inside that layer. Inside a layer, the shell thickness is constant in the initial situation. The outer shells therefore have the same thickness as the inner ones, but because the outer ones are located at greater distances from the core, their volume is larger than that of the inner shells. So, because the outer shells inside an element layer have larger volumes but the same amount of mass as more inwardly lying shells, the density of the outer shells is smaller. Therefore, there is a density gradient inside each element layer.

So, with this, the initial conditions of the red giant-like stellar atmosphere model have all been detailed, and can be implemented in the simulations.

7.2 Opacity results

In order to find the photosphere and ultimately the luminosity of the supernova, a short code\(^4\) was also written for determining the values for the opacity (\( \kappa \)). The intention was to include this code into the final program, to determine both photosphere properties and the luminosity. Whereas modelling the photosphere ultimately failed, the code for the opacity could be tested, and could be compared to literature values. The results of this small program are discussed in this section.

As can be seen in figures 20 and 21, the program does roughly follow the different regimes of the opacity, but compared with the real values the curves do not exactly match. Instead, there are small deviations from the real values. According to [22], this means in practice that in order to find good computer simulations, opacity tables have to be used and interpolated, in order to calculate opacities at every position.

\(^4\)See appendix C
Figure 20: Plot of the found theoretical opacities, using the opacity code from this paper. Composition is $X = 0.70$ and $Z = 0.02$. 
Figure 21: Plot of the true opacity, at the same densities as figure 20 \cite{20} (Note that in this plot the units are in CGS and NOT SI) \cite{22}.
7.3 Results

As noted earlier in this paper, no successful supernova explosions have been simulated. Simulations have been run for a wide range of initial conditions. For instance, a situation in which the temperature was initially constant throughout the star, or one in which the bottom shell had a much larger temperature than the rest of the shell has been tested. Another case that was studied had an initial temperature gradient across the stellar atmosphere. Furthermore, a more realistic red giant-like atmosphere - as described in section 1.3.2 - was used as input.

In all cases, the simulations ended due to errors with shells overlapping one another. When the bottom shell had lower temperature values, the bottom shell collapsed to the core and overlapped with it. At higher temperature values, the bottom shell expanded much too rapidly and overlapped with the shells lying above it. So, depending on the particulars of the respective initial condition, the overlap errors were caused by either the collapse of the bottom shell (such as the result depicted in figure 22), or by a much too rapid expansion (like the one shown in figure 23). The red giant-like initial situation resulted in an overlap error due to rapid expansion of the bottom layer. These unstable, rapid expansions might be made stable by lowering the time step size considerably, but this would increase the simulation time as well, up to the point where a simulation on a feasible timescale is not possible.

![Graph showing a zoomed-in view of the lowest shells of a supernova simulation using 100 shells. Initially, the temperature of each shell was set to 237000 K. The base time step size was set to 0.001s. The simulation ended in overlap.](image)

Figure 22: Graph showing a zoomed-in view of the lowest shells of a supernova simulation using 100 shells. Initially, the temperature of each shell was set to 237000 K. The base time step size was set to 0.001s. The simulation ended in overlap.

The fact that, for simulations with lower temperatures, the bottom shell collapsed to the core does not match theoretical expectations. The temperature of a shell changes with the volume as described by equation 17. So, as the bottom shell shrinks further and further, the temperature should rise enormously until it reaches a value at which the pressure in the bottom shell is large enough to cause the shell to expand again. This does not happen in the simulations. Instead, the bottom shell actually accelerates downward, reaching velocities on the order of $10^7 \text{ms}^{-1}$. Even though the temperature in the bottom shell rises, the pressure does not become large enough to counteract the collapse.

Another way in which the results do not match with what one would expect from equation 17 is that the second-lowest shell does not cool down. Because the lowest shell collapses to the core, the volume of the shell above it increases rapidly. According to equation 17, the temperature of this shell should decrease. However, this...
As in the simulation of figure 22, the initial temperature of each shell was set to 237000 K, but with the temperature of the bottom shell set to 3000 times 237000 K. The base time step size was set to 0.001s. The simulation ended in overlap. This overlap could be prevented by lowering the base time step size considerably. However, this would result in unpractically large simulation times.

does not happen in the simulation. Instead, the temperature of the second-lowest shell rises slightly. A solution to this problem would be to decrease the basis time step size considerably. By lowering the time step size, the bottom shell travels a smaller distance per time step, and can therefore get closer to the core before collapsing. The volume of the bottom shell can therefore shrink further with a smaller time step size, which allows the temperature to rise even higher. This could make it possible for the pressure to rise to a value high enough for expanding the bottom shell. However, lowering the time step size would greatly increase the simulation runtime, and is therefore not feasible for this project.

Another way in which this problem can be solved is by letting the temperature scale more with volume changes. For example, the expression for the change in temperature can be multiplied by a scaling factor $\alpha$:

\[ T_{\text{new}} = T_{\text{old}} + \alpha \Delta T, \]

where $T_{\text{new}}$ and $T_{\text{old}}$ are the temperatures at the new and the previous time step respectively. This increased scaling is introduced to prevent the bottom shell from overlapping with the core, and therefore this scaling should only be used when the bottom shell is at risk of collapse. Therefore, the scaling factor $\alpha$ is only applied to shells that are within a distance of a 50'th of the initial shell spacing $\Delta R$ of the core. It should be mentioned that the code for this scaling factor is not included in appendix B.

The results of the simulations with scaling factors can be found in figures 24 to 26. It was found that for values of $\alpha$ below 400, the collapse of the bottom shell is indeed stopped, but the shell does not expand much. Instead, it falls back down again after a short period of expansion. This leads to a kind of 'bouncing' behavior, which can be seen in figure 24.

When the scaling factor was set above 450, the bottom shell heated up very rapidly once it came close enough to the core for the scaling to be applied. This caused the bottom shell to immediately expand enormously, which resulted in overlap with the higher shells. An example of this can be seen in figure 25. So $\alpha$’s that are higher than 450 are not usable.

For values of $\alpha$ around 425, the bottom shell does expand, but slowly enough to prevent overlap. The
results of this simulation can be found in figure 26. The bottom shell rebounds from the core and shoots upwards, causing a shockwave to propagate throughout the stellar atmosphere. When two shells come close to each other and the shockwave is transmitted, the temperature and density take on peak values on the order of $10^{10} \text{K}$ and $10^6 \text{kg/m}^3$, respectively.

Sadly, it is not feasible to simulate what happens when the shockwave reaches the edge of the atmosphere. This simulation took two hours, so in order to simulate what happen as the shockwave progresses, a very large runtime is needed. This is unfortunate, because a shockwave caused by a rebound of stellar matter on the core is one of the theoretically expected causes of supernovae.

It must be stressed, however, that this scaling factor does not have a physical justification. The scaled-up temperature increase introduces energy to the system from nothing, and therefore violates the conservation of energy. The non-physicality of this scaling factor causes some of the results to be non-physical as well. For instance, after the shockwave has passed a shell, the temperature of that shell drops to extremely low values (a few tens of kelvins). According to the simulation data, the temperature of the bottom shell even becomes negative, even though the plots did not show it. Furthermore, the simulations with a scaling factor are much more dependent on the value of the base time step size. The previously described results all had a base time step of 0.01s. However, if a base time step size of 0.001s is chosen, all the results were similar to the ‘bouncing’ scenario, even for large values of $\alpha$. Normally, the results of the simulation do not depend too much on the time step size. In simulations without the scaling factor, a different time step size can slightly change the precise values of the physical quantities. Also, the base time step size can be the key factor in determining whether shells overlap or not. However, the base time step size should not completely alter the outcome of a simulation. In the scaled-temperature simulations, the time step size, among other variables, determines whether the bottom shell ‘bounces’ of the core (figure 24), explosively rebounds (figure 25), or sends a shockwave through the atmosphere (figure 26). The time discretization should not make such a big difference. The temperature scaling factor $\alpha$ is therefore not physical, and only serves to show what the program is capable of. These results should thus not be considered physically correct.

So, the introduction of a scaling factor for the change in temperature prevents the bottom layer from collapsing. However, the factor needs to be high (about 425) in order for the bottom shell to expand.
Figure 24: Graph showing the results of a simulation with an initial temperature of 237000K in every shell, a base time step of 0.01s, and a temperature scaling factor $\alpha$ of 10. This graph shows how the bottom shell 'bounces' of the core.
Figure 25: Graph showing the results of a simulation with an initial temperature of 237000K in every shell, a base time step of 0.01s, and a temperature scaling factor $\alpha$ of 450. It shows how the bottom shells expands explosively after reaching a radius at which the scaling factor is applied. The simulation ended in overlap.
Figure 26: Graph showing the results of a simulation with an initial temperature of 237000K in every shell, a base time step of 0.01s, and a temperature scaling factor $\alpha$ of 425. It shows how the bottom shell falls down to the core, rebounds, and then sends a shockwave through the lower part of the stellar atmosphere.
8 Discussion

8.1 Broken causality

The assumption that stellar material is spread homogeneously throughout a shell can lead to inaccuracies. As shell radii change over time, the gas inside shells needs to disperse in order to maintain homogeneous distributions. Because spherical symmetry has been assumed, stellar gas redistributes only in the radial direction. When, for instance, the radius of the surface at the inside of a shell changes, the gas inside that shell will distribute in such a way that its density is constant throughout the shell. This means that information about the new volume of the shell has travelled from the gas at the inside to that at the outside of the shell. This information travels at a velocity of:

$$v = \frac{\Delta r}{\Delta t},$$  \hspace{1cm} (85)

where \(\Delta r\) is the shell’s radial width, and \(\Delta t\) the time step size. This poses no physical problem provided that \(v\) is smaller than the speed of sound inside the shell matter, as it is that speed with which pressures waves are transmitted. For the simulation of the Earth’s atmosphere, using ten shells, the values of \(\Delta r\) oscillate around the initial value of 5000 m. The time step size is, at its smallest, around 0.00001s. This leads to an information velocity of \(v = \frac{5000m}{0.00001s} = 5 \cdot 10^8 m/s\), which is larger than the velocity of light. Information transfer at this rate is obviously non-physical. In order to prevent faster-than-light information transfer like this, the time step size needs to be chosen in accordance with the largest value of \(\Delta r\) at a given moment. The problem with this is that the most separated -and therefore the most stable- shells would determine the time step size, instead of the closest and least stable ones (see section 3.2.3). Adjusting the time step size in this way would therefore result in large numerical instabilities. Furthermore, the violation Special Relativity does not seem to matter for this model: the results from earth-like atmosphere simulations were physically correct, and could even model shockwaves travelling through the atmosphere. Because of these reasons, it was decided to disregard the faster-than-light communication occurring in the model.

8.2 Conservation of energy

The total energy was found for simulations of both the earth-like atmosphere and the supernova explosion, as derived in section 2.1. Remarkably, it was found from the before-and after values that whereas the former showcased a conservation (give or take a fraction of some percents due to rounding off errors) of energy, the latter did not seem to obey the energy conservation laws. The reason for this violation is likely due to the kinetic energy term of the total energy (see equation 12). In order to calculate the kinetic energy of a shell, the velocity of the stellar material inside that shell needs to be known. However, only the velocities of the shell radii are calculated in the model. Therefore, an approximation for the velocity of the shell matter needed to be made. In this case, the average of the velocities of the two surfaces enclosing a shell was taken to represent the velocity of the stellar matter in the middle of that shell. This velocity, in turn, represents the velocity of all the mass inside a shell. Because this method is a coarse approximation, use of it results in numerical errors. These errors are, apparently, small for earth atmosphere simulations, where the velocities involved are quite small. Energy is therefore conserved for earth atmosphere simulations. For supernova simulations, however, the velocities of every shell are much larger. This might cause the error in the approximation to be larger as well, which results in a violation of the conservation of energy.

8.3 Comparing this paper’s model to the original model of Arnett

8.3.1 Homologous expansion

In the original article on modelling the lightcurves of type-II supernovae\[1\], assume that a supernova explosion can be modelled through homologous expansion\[1\]. This assumption boils down to the following relation: \(\dot{R}/R = C\). As the derivative of the position of the outer layer of the star divided by the position itself is a constant\[2\]. In the model of \[1\] a slightly different type of homologous expansion was used. In this model
it is assumed that $\dot{V}/V = 3v_{sc}/R(t)$. Where the constant $v_{sc}$ is a constant which sets the velocity scale\textsuperscript{1}. With this assumption, the assumption that work is ignored and the assumption of that Thomson scattering dominates, it is able to analytical solve the differential equation that describes a supernova type II\textsuperscript{1}. it proved to be difficult to realise this behavior without violating physics behind the model. To realise this kind of expansion in the used model, gravity needs to be shut down, as the star would otherwise simply contract again after expanding homologously, in the model of \textsuperscript{1} no gravity is taken in to account in any way.

8.3.2 Pressure

In the model of \textsuperscript{1}, it is assumed that radiation pressure is dominant, ignoring the ideal gas pressure. In this paper’s model however, the ideal pressure is not ignored and a mixture of both a photon gas and an ideal gas is used to describe the pressure. Incorporating the ideal gas pressure in the model of this paper seems to be of importance when there are close encounters between shells, for example when a shockwave is transmitted through the (stellar) atmosphere. As shells converge and the volume between them becomes smaller, the temperature and the density of that shell increase. This causes both the radiation pressure, and the ideal gas pressure to increase. It was observed in the model of this paper that upon close encounters between shells in the earth-like atmosphere simulations, the density and temperature take on values for which the ideal gas pressure is dominant. This is shown in figure \textsuperscript{27} This figure consists of a log-scaled density-temperature plot, containing the boundary line that separates the density-temperature region in which the ideal gas pressure is dominant, from the radiation dominated region (below and above the line, respectively). This is not surprising, because at earth temperatures, the ideal gas pressure is always dominant over the radiation pressure. The red dots in this plot represent the temperatures and densities of shells that are compressed during the transmission of a shockwave in an earth-like atmosphere simulation. One can see that all dots lie in the ideal gas regime. This is not surprising, because the ideal gas pressure is always dominant for earth-like temperatures.

Under supernova conditions, however, the temperature is large enough for the radiation pressure to become dominant. Because of this, Arnett neglected the ideal gas pressure in his supernova model \textsuperscript{1}. In this paper’s model, both the ideal gas pressure and the radiation pressure were used. To see whether this mixture of gases was necessary for the model to work correctly, one can again look at the temperatures and densities of shells during shockwave transmission. For this, the scaling factor $\alpha$ was set to 425, and the resulting shockwave was examined. The results of this can be found in figure \textsuperscript{28} For this plot, three random shockwave transmission points were chosen. Two of the red dots lie in the radiation-dominated regime, but one of them is in the ideal regime. Since these shockwaves transmission points were taken at random, it is safe to assume that there are more than one shells for which the ideal gas pressure is dominant. Therefore, one can conclude that, for this paper’s model, it is necessary to use a mixture of both a radiation and an ideal gas, since both gas terms are important for the correct transmission of shockwaves. In this way, the model of this paper differs from the one from \textsuperscript{1}, which only uses a radiation gas.
Figure 27: The equation of state for a gas of free particles in the log $T$-log $\rho$ plane. The violet line is the boundary between the areas where, respectively, radiation gas pressure (above the line), and ideal gas pressure (below the line) dominate. The red dots represent the temperatures and densities of shells that are compressed during the transmission of a shockwave in the earth-like atmosphere. All dots lie below the line, and therefore the ideal gas pressure is dominant in the transmission of shockwaves in the earth-like atmosphere.
Figure 28: The equation of state for a gas of free particles in the log $T$-log $\rho$ plane. The violet line is the boundary between the areas where, respectively, radiation gas pressure (above the line), and ideal gas pressure (below the line) dominate. The red dots represent the temperatures and densities of shells that are compressed during the transmission of a shockwave in the supernova simulation with scaling factor $\alpha = 425$. Two of these points lie in the radiation regime, and one of them lies in the ideal regime.


8.3.3 Presence of radioactive elements

In [1] it is assumed that a certain amount of $^{56}\text{Ni}$ is present in the ejected matter. In the used model, radioactive elements were ultimately ignored, as the simulation would only last for the order of seconds at best, with radioactivity-powered luminosity only providing significant contributions after days.

8.4 Light curves

One of the goals was to ultimately create a plot of the total luminosity against time, also known as a light curve. For a supernova, this allows for determining characteristic behavior, of which the plateau phase of a type II supernova is an example. Two components of the luminosity were programmed: radioactive decay and temperature exchange through radiation. These terms, together with the kinetic contribution would give the full expression for an eventual light curve. The first problem with creating a light curve was that the radioactive term was not numerically stable for the given star parameters. The radioactive decay for enormous amounts of radioactive $^{56}\text{Ni}$ was calculated by the code, and resulted in instability, as the values started exceeding the program limit very soon. Second, the temperature exchange between shells would provide a valuable addition to both the supernova mechanism and the light curve, if it had not showcased the same problems as the radioactive term. Also for the radiative temperature exchange mechanism, the luminosity, numerical instability was observed. This instability was likely caused by the fact that the luminosity expression used in the derivation probably only holds for entire stars, instead of spherical stellar shells. With the luminosity code included, shells would overlap after the first time step, even if this time step was taken enormously small ($\Delta t = 0.0001$). An option was therefore added to put the radiative temperature exchange on or off in the simulation.

8.5 General improvement: smaller discretizations

One of the biggest improvements, but also one of the biggest challenges for improving the code would be creating a much smaller discretization. This includes both the timestep and the physical discretization. Smaller timesteps would result in less overlap errors, as each respective timestep would induce a smaller difference in the output variables. At present, the enormous changes that sometimes occur in the radii of different shells cause overlap errors at timescales as small as 0.01 seconds. Upon making the times sufficiently small enough, one could expect the various problems that arise in the luminosity calculations to disappear. Lack of computer power however prevents the discretizations from becoming much smaller.
9 Conclusion

The initial purpose of the project was to model a magnetar-powered type-II supernova, to see whether a so-called 'superluminous supernova' could be replicated. Whereas the modelling of the magnetar had to be left out as a physically feasible supernova was not observed from the simulations, various successful tests were performed on an easier to model 'trial system', namely an earth-like atmosphere. The tests of the hydrodynamics on an earth-like atmosphere showcased various interesting phenomena, including the transmission of shockwaves. These shockwaves propagated with velocities in the correct order of magnitude, according to the derived sound speeds as a function of shockwave temperature. The behavior of the earth-like atmosphere indicates that the hydrodynamics for the model in general is likely correct. The problems surrounding the supernova simulations consisted of multiple aspects. The first aspect that negatively influenced the quality of the simulations were the used discretizations. At some points, the code would simply experience too large fluctuations in values to function properly, which resulted in that the luminosity and luminosity-based heat transfer could not be used. When using these options, one would run into overlapping shells, even when using small timesteps. This was combated by using a scaling timestep, but restraints in computing power resulted in that stable simulations with small timesteps would take too long. A second factor consisted of the choice of the physical system. In order for a proper explosion to occur, fitting initial conditions, consisting of correct density-, pressure- and temperature profiles as well as a core-envelope transition had to be included properly. It is very likely that whereas the assumptions that were made were not per se wrong in and of themselves, but that the combination of them was not a fitting approximation for the circumstances at the beginning of the supernova explosion.
10 Alternatives and prospects

10.1 Other methods of modelling

There are two main approaches that could have been used instead to model a supernova explosion, both are based upon readily available codes. The first alternative consists of ‘skipping’ the derivation of the hydrodynamics as was done in the model of this paper. One can choose to incorporate the FLASH-code\cite{12}, which is considered the golden standard for calculating hydrodynamical phenomena like supernovae. Potter et al.\cite{23} have illustrated this method by using FLASH to incorporate the hydrodynamics of supernova SN1987A, after which they were capable of including their components of interest. These included, for example, various recorded phenomena like an hourglass-shaped asymmetrical radio morphology, or a certain ring-shaped interaction layer between interstellar matter and the supernova shockwave. Also, using the FLASH-code would make the incorporation of various initial conditions easier, as Potter et al.\cite{23} could use a blue supergiant instead of a red supergiant, which in the model from this paper would have posed a much greater challenge.

Another way in which other simulation methods differ from this paper’s model, is the fact that they are far more elaborate. The supernova model of this paper practically only uses hydrodynamics as a building block for the shockwave and the resulting expansion, whereas this is in reality just one of three important mechanisms. Bruenn et al.\cite{5} reported very feasible results using the CHIMERA-code package, which departs from a different theoretical framework than the model of this paper. In CHIMERA, the emphasis rests on the neutrino-heating system as a trigger for the supernova explosion. This is coupled with hydrodynamics to provide a framework for the expansion, after which a radioactive/radiative component is added to complete the dynamics of the supernova. In this paper’s model, expressions were found for the radiative and radioactive components, but these did not seem to work properly and were therefore not included in the code.

An interesting similarity between the two professional and the created model is that both FLASH and CHIMERA also experience minor instability issues at the start of the simulations. However, whereas FLASH and CHIMERA will remain numerically stable, the model of this paper would create physically impossible situations such as overlapping star shells, which is another confirmation of the paper’s model being incomplete in some sense.

When compared to the current standards of supernova modelling, one can conclude that the created model lacks two of the three essential parts for a full supernova simulation; a working radioactive and radiative interaction and a neutron-heating mechanism to power the explosion. One could in the future choose to use either FLASH or CHIMERA to ease the hydrodynamic modelling, after which hypotheses surrounding phenomena outside of the hydrodynamics of the supernova can be tested. These would include the original intention of the project, namely the simulation of the influences of a magnetar on the behavior of the supernova explosion.

10.2 Follow-up: adding the magnetar

For a supernova, one can calculate the luminosity based upon the energy that accompanies the explosion. Incorporating a magnetar in the lightcurve thus requires a similar approach.

As a rapidly spinning magnetic object, one can expect a magnetar to radiate enormous amounts of energy. To add a reasonable, time-depentent expression for the magnetar-component of the total luminosity, one needs to find:

- Size and mass of the leftover core
- Magnetic field strength and angular frequency based upon star properties, these can be used for calculating the energy of the emmitted radiation \cite{17}
- Spin-down time of the magnetar for determining the time dependency of the energy. \cite{17}
References


Pols, O.R., *Stellar structure and evolution*


A Derivation of the dynamics of an Earth-like atmosphere

Consider an infinitesimally small cylinder of air, with area $\Delta A$ and height $\Delta r$. If the pressure at the bottom of the cylinder is $p_{\text{bottom}} = p(r)$, then the pressure at the top is:

$$p_{\text{top}} = p(r) + dp$$

(86)

This allows for $dp$ to be written as:

$$dp = \frac{\partial p}{\partial r}dr$$

(87)

Besides the expression for $dp$, the mass of the cylinder is also needed in terms of finite differences of its volume:

$$M = \rho dAdr$$

(88)

The atmosphere is supposed to be in a hydrostatic equilibrium, so its resulting force ought to be zero:

$$F_{\text{total}} = F_{\text{top}} + F_{\text{bottom}} + F_g = 0$$

(89)

$$-gM - (p + dp)dA + pdA = 0$$

(90)

$$-g\rho dAdr - (p + dp)dA + pdA = 0$$

(91)

Substituting (87) into (91) and simplifying yields the following differential equation for a earth-like atmosphere:

$$\frac{\partial p}{\partial r} + g\rho = 0$$

(92)

This differential equation can be solved for an atmosphere with constant temperature. Before this is done, it is useful to define an expression for $\rho$.

$$\rho = p \cdot R \cdot T$$

(93)

This expression is derived from the ideal gas law. Inserting $\rho$ in the differential equation gives:

$$\frac{\partial p}{\partial p} = -\frac{g \cdot P}{R \cdot T}$$

(94)

In which one can define a constant $H$, which links a constant temperature $T_0$, gravitation and the gas constant:

$$H = \frac{R \cdot T_0}{g}$$

(95)

Equation (94) can now easily be solved by separating variables. The solution for $\rho$ follows from the solution for $p(r)$, which is given in (96)

$$p(r) = p_{\text{surface}} \exp\left(-\frac{r}{H}\right)$$

(96)

$$\rho(r) = \frac{p_{\text{surface}}}{RT_0} \exp\left(-\frac{r}{H}\right)$$

(97)

For solving the atmosphere for a non-constant temperature distribution, one has to redefine $H$. Instead of a constant temperature $T_0$, one now chooses a radially dependent function for $T$. The resulting solutions differ from the constant-temperature in that they exhibit an integral in the exponential, which can be evaluated algebraically.

$$p(r) = p_{\text{surface}} \exp\left(-\int_0^r \frac{dr'}{H(r')}\right)$$

(98)
\[ \rho(r) = \frac{p_{\text{surface}}}{RT_0} \exp\left(-\int_0^r \frac{dr'}{H(r')} \right) \]  

(99)

\[ H(r) = \frac{R \cdot T(r)}{g} \]  

(100)

In the model, the values for the temperature are the same inside each shell. Therefore, the constant temperature results (equations 97 and 98 are valid. Of these two, only the density distribution will be used. The reasons for this are detailed in section 6.1.
B Full Python script

```python
#!/usr/bin/env python
# This program simulates a supernova explosion or an Earth-like atmosphere
# made by Folkert Nolte, Jeroen Ubink and Henry de Vries

# The following lines are for importing a number of libraries
from __future__ import division
import numpy as np
import scipy as sc
import scipy.constants as scc
from matplotlib.pyplot import figure, show
from scipy.interpolate import interp1d, splrep, splev
np.set_printoptions(threshold='nan')
import math as math
import time

# This variable is used for labeling the output files of the simulation
stime = 7090710

# FUNCTIONS
# Density distribution at t=0
def rhof(r, Rearth, molecularmass):
    a = beginrho * np.exp(-exponentfactorrho * (r - Rearth - deltaR) / Hconstant(r, Rearth, molecularmass))
    return a

# Temperature distribution at t=0, in this case an equal distribution
def temp(r, Rearth):
    return begintemp

# Pressure distribution at t=0
def pres(r, Rearth, molecularmass):
    return beginpres * np.exp(-exponentfactorpres * (r - Rearth - deltaR) / Hconstant(r, Rearth, molecularmass))

# The function for the H-constant
def Hconstant(r, Rearth, molecularmass):
    return gascon * temp(r, Rearth) / (gravacc * molecularmass * avogadro)

# DERIVATIVE FUNCTION
# The following function takes two arrays and approximates the derivative
# at each element
def der(a, b):
    size = len(a)
    k = np.zeros(size)
    for i in range(1, size):
        k[i] = (a[i] - a[i - 1]) / (b[i] - b[i - 1])
    return k

# DYNAMICS FUNCTIONS
# Acceleration
def acc(mencel, r, rho, dp_over_dr):
    if r == 0:
        return 0
    else:
        return -G*mencel / (r*r) - 1/rho * dp_over_dr

# Pressure
def pressure_gen(T, rho, molecularmass):
    return a/3*T**4 + rho*bolts*T/molecularmass

# This function is used for interpolating the temperature between element layers
# in the situation of stellar distribution with elements
def tempdistre(r, T1, T2, R1, R2):
    return T1*np.exp(-(1/(R2-R1)))*np.log(T1/T2)*r

# ENERGY FUNCTIONS
# Gravitational energy
def gravenergy(mass, massencel, r):
    return mass*massencel*G/r

# Internal energy
def internalenergy(volume, temperature, mass, molecularmass):
    return enerec*volume*temperature**4 + (3/2)*bolts*mass/molecularmass*temperature

# Kinetic energy
def kineticenergy(mass, velocity):
    return 0.5*mass*velocity**2
```
# DATA STORAGE FUNCTION#

```python
def storage(v_total, r_total, T_total, rho_total, dp_over_dr_total):
    v_total = np.vstack((v_total, v))
    r_total = np.vstack((r_total, r))
    T_total = np.vstack((T_total, T))
    rho_total = np.vstack((rho_total, rho))
    dp_over_dr_total = np.vstack((dp_over_dr_total, dp_over_dr))
    return v_total, r_total, T_total, rho_total, dp_over_dr_total
```

# PLOT FUNCTIONS#

```python
def plotf(array, timevar, r, ylabel, title, name, show1):
    steps = len(r)
    fig = figure()
    frame = fig.add_subplot(1, 1, 1)
    for i in range(1, steps):
        var = 0.09 * shellnumber * i
        p1 = frame.plot(timevar, array[:, i], color=str(var))

    frame.set_xlabel('Time (in s)', style='italic')
    frame.set_ylabel(ylabel, style='italic')
    frame.set_title(title)
    frame.grid(True)
    fig.savefig(name + '.png')
    if show1 == True:
        show()
```

# THE INITIAL CONDITIONS#

```python
Molar = 1.99 * 10 ** 30
Munit = 1.660538921 * 10 ** -27
Rsolar = 6.955 * 10 ** 8

# USER INPUTS#

atmosim = input('Do you wish to simulate an unstable star or an Earth-like atmosphere? Input 1 for Earth, anything else for the star: ')
wanthundred = input('Do you wish to use 100 or 10 shells? Input 1 for hundred, anything else for 10: ')
wantds = input('Do you want to use time step scaling? 1 for yes, 0 for no: ')
tmax = input('Enter a max runtime in minutes: ')
```

```python
if atmosim == '1':
    # Use earth conditions
    # Set the shell number
    shellnumber = 1
    if wanthundred == '1':
        shellnumber = 10
    # Set the shell thickness
    deltaR = 5000 * shellnumber
    # Radius of Earth:
    Rearth = 6400 * 10 ** 3 - deltaR
    # Set parameters for use in the initial distribution functions:
    beginrho = 1.251
    beginpres = 653
    beginh = 203
    # It is 1 here because here no adjustment is necessary
    exponentfactorrho = 1
    exponentfactorpres = 1
    # Set the title of the output graph
    graphtitle = 'Numerical_simulation_of_the_atmosphere_of_the_earth'
    # Radius of the atmosphere:
    Ratm = 50000 + Rearth
    # Set the mass of the Earth (the name Mstar is still used for this variable)
    Mstar = 5.9721986 * 10 ** 24
    # Set that the division in element layers as used in the star situation is not used
    wantdensity = 0
    # This variable is the molecular mass of nitrogen
    mol = 4.65173 * 10 ** (-26)
else:
    # Use star conditions
    # Initially set that the element distribution is not used
    wantdensity = 0
    # Adjust the number of shells:
```

64
shellnumber=1
if wanthundred==1:
    shellnumber=0.1
deltaR = 50000000*shellnumber
# Set the radius of the star core (the term Rearth is a remnant of earlier versions of the model and has been kept in. The variable does in fact denote the neutron star core's radius. Rearth = 12000-deltaR
# Set the radius of the stellar atmosphere
Ratm = 500000000+Rearth
# Set the parameters for the initial distribution:
exponentfactorrho=5
exponentfactorpres=5
begintrho=1.251e5
beginpres=237000
beginmep=237000
# Set the mass of the stellar core:
Mstar=Msolar*1.4
# Set the title for the output graphs:
# C h e c k w h e t h e r t h e u s e r h a s c h o s e n 1 0 0 s h e l l s
# Ask whether the user wants to use the stellar atmosphere distribution where it is divided in a number of layers containing certain elements.
# A k s t h e u s e r t o i n p u t s o m e t i m e v a r i a b l e s :
# Calculation of the temperature array
T = np.array([temp(x, Rearth) for x in r], dtype='d')

# Initialization of another array
delta_T = np.zeros(r_length, dtype='d')
delta_rho = np.zeros(r_length, dtype='d')

if wantdensity == 1:  # check if the user wants to use the red giant-like element layer division
    # Set the boundary radii of the element layers
    r[0] = 12 e3
    r[10] = 10120 e3
    r[20] = 21738 e3
    r[25] = 26385 e3
    r[40] = 60476 e3
    r[60] = 389066 e3
    r[99] = 1000 Molar

    # Set the temperatures at the boundaries
    T[0] = 334 e7
    T[10] = 199 e7
    T[20] = 157 e7
    T[25] = 87 e7
    T[40] = 18.8 e7
    T[60] = 3.69 e7
    T[99] = 3500

    # Define the radii of shells in between layers and interpolate the temperatures
    for k in xrange(1, 10):
        r[k] = r[0] + (r[10] - r[0]) / 10 * k
        T[k] = tempdistr(r[k], T[60], T[99], r[60], r[99])
        mass[k] = mass[0] + mass_shell[k]
        molecularmass[k] = 20 / 11 * Munit
    for k in xrange(11, 20):
        r[k] = r[10] + (r[20] - r[10]) / 10 * (k - 10)
        T[k] = tempdistr(r[k], T[10], T[20], r[10], r[20])
        mass[k] = 16 / 9 * Munit
    for k in xrange(21, 25):
        r[k] = r[20] + (r[25] - r[20]) / 5 * (k - 20)
        T[k] = tempdistr(r[k], T[20], T[25], r[20], r[25])
        mass[k] = 12 / 7 * Munit
    for k in xrange(26, 40):
        r[k] = r[25] + (r[40] - r[25]) / 15 * (k - 25)
        T[k] = tempdistr(r[k], T[25], T[40], r[25], r[40])
        mass[k] = 4 / 3 * Munit
    for k in xrange(41, 60):
        r[k] = r[40] + (r[60] - r[40]) / 20 * (k - 40)
        T[k] = tempdistr(r[k], T[40], T[60], r[40], r[60])
        mass[k] = 18 / 10 * Molar

    # Define the mass of the matter inside each shell
    for k in xrange(0, 21):
        mass_shell[k] = 0.12 Molar / 10
    for k in xrange(21, 26):
        mass_shell[k] = 0.12 Molar / 5
    for k in xrange(26, 40):
        mass_shell[k] = 0.12 Molar / 15
    for k in xrange(41, 61):
        mass_shell[k] = 0.12 Molar / 20
    for k in xrange(61, 99):
        mass_shell[k] = 18 Molar / 40

    # Create the array containing enclosed masses
    for i in xrange(0, r_length - 1):
        mass[i + 1] = mass[i] + mass_shell[i]

    # Define the mass arrays when the red giant-like element layer division is not used
    if wantdensity == 1:
        for i in xrange(0, r_length - 1):
            mass_shell[i] = 4 * pi / 3 * (r[i + 1]**3 - r[i]**3) * rho(r[i] + 5 * delta_R, Rearth, molecularmass[i])
            mass[i + 1] = mass[i] + mass_shell[i]
        for i in xrange(35, r_length - 1):
            mass_shell[35] = 10 * mass_shell[35]
        for i in xrange(35, r_length - 1):
            mass_shell[i] = mass[i] * r[i] / r_shell

    # Calculate the densities of the stellar matter in case of the red giant-like element layer division
    if wantdensity == 1:
        for k in xrange(0, r_length - 1):
            loop, rho[k] = mass_shell[k] / (4 * pi / 3 * (r[k + 1]**3 - r[k]**3))
        for k in xrange(1, r_length - 1):
            rho[k] = loop, rho[k - 1] * loop, rho[k] / 2
            rho[k] = loop, rho[k - 1] * rho[k - 2] / 10
        # Initialize the energy arrays
        energybegin = np.zeros(r_length - 1, dtype='d')
        for k in xrange(0, r_length - 1):
            energy[k] = energybegin[k] + pressure_gen(T, rho, molecularmass)
Calculation of the dp over dr array
rvorder contains the positions of the center of each shell
rvorder = np.ones(r_length, dtype='d')
for k in range(0, r_length-1):
    rvorder[k] = (r[k]+r[k+1]-r[k])/2
rvorder[r_length-1] = r[r_length-1] + 0.5*deltaR

Create the initial speed array, we assume this is m/s at all radial coordinates
v = np.ones(r_length, dtype='d')

Initialize the acceleration array
accel = np.zeros(r_length, dtype='d')

# Data storage#
# Store the initial situation
v_total = v
r_total = r
T_total = T
rho_total = rho
dp_over_dr_total = dp_over_dr
erel_total = accel
rho_new = np.zeros(r_length-1, dtype='d')

# define error as 0, meaning no shell overlap has occurred yet
error = 0
# Initialize the time storage
tottime = []
tottime.append(0)

# Initialize the number of iterations of the program
iterations = 0
# Save the old time step for use in adjusting it later on
dtold = dt
# Set the first element of the array containing the values of the radii at a previous time step
r[0] = r[0]

# Initialize some other time variables
timevar = dt
tftime = time.time()
tmax = max*60

# Set the initial two shell distances for the red giant-like element layer division situation
if wantdensity == 1:
    deltaR1 = r[15] - r[14]
    deltaR2 = r[35] - r[34]

# Calculate the initial total energy
for k in range(0, r_length-1):
    energybegin[k] = internalenergy((4/3)*pi*(r[k+1]**3 - r[k]**3)) * T[k] * mass_shell[k] * molecularmass[k]
    gravenergy = mass_shell[k] * mass[k] * r[k] * kineticenergy(mass_shell[k] * T[k] * (v[k+1] + v[k]) / 2)

# Initialize the total energy array that will be changed over the course of the runtime
energytotal = energybegin

if atmetmim == 1 and wanthundred == 1:
    T[0] = 1e9
    T[1] = 8.5e8
    T[2] = 7e8
    T[3] = 5.6e8
    T[4] = 5e8
    T[5] = 3e8
    T[6] = 1e8
    T[7] = 1e7
    T[8] = 1e6
    T[9] = 1e5
    rho[0] = 10*rho[0]
    rho[1] = 9*rho[1]
    rho[8] = 2*rho[8]
    print T, rho

# Start the time loop in which all quantities are evaluated step by step
i = 0
while timevar <= t_final and error == 0: # Keep running while max. time has not been exceeded and no overlap error has occurred yet
    tottime.append(timevar) # update the time array
    # Calculation of the new r and v array
damping = 1 - dampfactor * dt # calculate the damping
    for k in range(1, r_length):
dr[k]=r[k]  # store the old radii
accel[k]=acc(mass[k], r[k], rho[k], dp_over_dr[k])
r[k]=r[k]+v[k]*dt+accel[k]*dt**2
t[k]=t[k]+v[k]*dt+damp+accel[k]*dt

# Check whether shells overlap
for k in range(1, r_length-1):
    if r[k] < r[k-1]:
        print 'overlap, shell:', k
        print overlap_error
        error = 1
        break

# storage of v, r, T, rho and dp_over_dr
v_total, r_total, T_total, rho_total, dp_over_dr_total =
storage(v_total, r_total, T_total, rho_total, dp_over_dr_total)
accel_total = np.vstack((accel_total, accel))

# Update the temperature and pressure
for k in range(0, r_length-1):
    x = mass_shell[k]/(4*pi/3*(r[k]+1)**3 - r[k]**3)  # temporarily store the density in the variable x
    # Calculate the terms used in the delta T equation
    D1=(3/2)*boltz_mass_shell[k]/molecular_mass[k]
    C1=(a/3)*boltz_mass_shell[k]/molecular_mass[k]
    E1=4*pi**3*(r[k+1]**3 - r[k]**3)*T[k]**3
    delta_T[k]=(-1*(D1+B1+C1)*DeltaV)/(D1+E1)  # calculate delta T

    if time() - t1 >= tmax:
        print 'max time exceeded'
        error = 1
        break
    if (time() - t1) > error:
        error = 1
        print 'error is used to end the loop when max time has been exceeded'

    # storage of v, r, T, rho and dp_over_dr
    v_total, r_total, T_total, rho_total, dp_over_dr_total =
storage(v_total, r_total, T_total, rho_total, dp_over_dr_total)
accel_total = np.vstack((accel_total, accel))

    # Update the temperature array
    T=T+delta_T

    # Update the temperature array
    v_avg=abs(np.average(v))
    newdelta_R=r_shift-(r[k]-r[k-1])
    if newdelta_R > 0.000001:
        if time() - t1 >= tmax:
            print 'max time exceeded'
            error = 1
            break
        if (time() - t1) > error:
            error = 1
            print 'error is used to end the loop when max time has been exceeded'

    # Update the temperature array
    v_total, r_total, T_total, rho_total, dp_over_dr_total =
storage(v_total, r_total, T_total, rho_total, dp_over_dr_total)
accel_total = np.vstack((accel_total, accel))

    # Update the temperature array
    T=T+delta_T

    # Update the pressure array
    pressure[k] = pressure_gen(T[k], loop_rho[k], molecular_mass[k])
    print timevar # print the time step to show the user how far the program is

    # Initialize the energy array
    energy=np.zeros(r_length-1, dtype='d')
    # Calculate the energy of each shell
    for k in range(0, r_length-1):
        energy[k]=internal_energy(((4/3)*pi*(r[k+1]**3 - r[k]**3)), T[k], mass_shell[k], molecular_mass[k])
    energy[k]=gravenergy(mass_shell[k], mass[k], r[k])+kineticenergy(mass_shell[k], v[k+1]+v[k]/2)

    # Update the energy storage
    energy_total=energy_total+energy

    # Calculate the density at shell radii
    for k in range(1, r_length-1):
        rho[k]=loop_rho[k-1]+loop_rho[k]/2
    # Create the array containing the positions of the centers of the shells
    for k in range(0, r_length-1):
        reorder[k]=r[k]+r[k+1]-r[k-1]

    rho[k]=loop_rho[k-1]+loop_rho[k]/2
    # Calculate the pressure gradients
    dp_over_dr=der(pressure, reorder)

    # Adjust the time step size
    if wantdensity!=1:
        if when the red giant-like element layer division is not used:
            r_loss_factor=1
            v_shift[v]=r[k+1]-r[k-1]
            rcheck=min(newdelta_R)
            vcheck=max(v)
            vavg=np.average(v)
            deltaRavg=average(newdelta_R)
            dtfactor=(rcheck/ deltaRavg)**2
            if dtfactor > 0.000001:
                if time() - t1 >= tmax:
                    print 'max time exceeded'
                    error = 1
                    break
                if (time() - t1) > error:
                    error = 1
                    print 'error is used to end the loop when max time has been exceeded'

                # storage of v, r, T, rho and dp_over_dr
                v_total, r_total, T_total, rho_total, dp_over_dr_total =
storage(v_total, r_total, T_total, rho_total, dp_over_dr_total)
accel_total = np.vstack((accel_total, accel))
else:
    print 'dt reached minimum'  # Tell the user that the time step size has reached its lowest value
    # indicating that the program is highly unstable and liable to have shell overlap
if want_density==1 and want_dt_scaling==1:
    # Adjust the time step size when the red giant-like element layer division is used
    rshift=r[1:rl]  # Tell the user that the time steps size has reached its lowest value
    newdeltaR=rshift-r[0:rl-1]
    rcheck1=min(newdeltaR[25:99])
    for k in xrange(1,26):
        dtfactor1=(rcheck1/deltaR1)**2
    for k in xrange(26,99):
        dtfactor2=(rcheck2/deltaR2)**2
    dtfactor=min(dtfactor1,dtfactor2)
    if dtfactor>0.00001:
        dt=dtold*dtfactor
    else:
        print 'dt reached minimum'
        # Increase the total time by dt
        timevar=timevar+dt
        iterations=iterations+1
        i+=1
# Store the data in text files
np.savetxt('data_radius'+str(stime)+'.txt', rtotal, fmt='%i', delimiter=' ')
np.savetxt('data_temp'+str(stime)+'.txt', Ttotal, fmt='%f', delimiter=' ')
np.savetxt('data_rho'+str(stime)+'.txt', rhototal, fmt='%f', delimiter=' ')
np.savetxt('data_pressure'+str(stime)+'.txt', ptotal, fmt='%f', delimiter=' ')
np.savetxt('data_over_dr'+str(stime)+'.txt', dpoverdrtotal, fmt='%f', delimiter=' ')
np.savetxt('data_acceleration'+str(stime)+'.txt', acceltotal, fmt='%f', delimiter=' ')
np.savetxt('data_time'+str(stime)+'.txt', tottime, fmt='%f', delimiter=' ')
np.savetxt('data_energy'+str(stime)+'.txt', energytotal, fmt='%f', delimiter=' ')
# Print the initial and final energies so the user can compare them
print 'The energy was initially:', np.sum(energybegin)
print 'At the end it is:', np.sum(energy)
# Show the runtime
print 'runtime:', (time.time()-t1)/60, 'minutes'
# Plot the time evolution of physical quantities
plot(rtotal, tottime, r, 'Radius(in pm)', graphtitle, plot_str(stime), True)
plot(Ttotal, tottime, T, 'Temperature(in K)', graphtitle, plot_str(stime), True)
plot(veltotal, tottime, v, 'speed(in m/s)', graphtitle, plot_str(stime), True)
plot(dpoverdrtotal, tottime, dpoverdr, 'dp/dr', graphtitle, plot_str(stime), True)
plot(rhototal, tottime, rho, 'rho', graphtitle, plot_str(stime), True)
plot(ptotal, tottime, pressure, 'Pressure(in Pa)', graphtitle, plot_str(stime), True)
plot(acceltotal, tottime, accel, 'Acceleration(in m/s^2)', graphtitle, plot_str(stime), True)
plot(energytotal, tottime, energy, 'Energy(in J)', graphtitle, plot_str(stime), True)
C  Opacity code

```python
#!/usr/bin/env python
# calculation of kappa with the help of a function
from __future__ import division

def kappa( X, rho, T, charge):
    # Calculate heavy elements abundance
    Z = 1 - X
    # Calculate electron scattering opacity
    ke = 0.2*(1+X)/((1 + 2.7*e11*(rho/(T**2)))*(1+(T/4.5*e8)**0.86))
    # Calculate the Kramers opacity
    kk = 4*e25*(1+X)*(Z+0.001)*rho/(T**3.5)
    # Calculate opacity due to hydrogen ions
    kh = 1.1e-25*(Z**0.5)*(rho**0.5)*(T**7.7)
    # Calculate low temperature opacity due to H2O and CO
    km = 0.1*Z
    # Calculate the radiation opacity
    krad = km + 1/(ke + kk) + 1/(kk + kh)
    # Calculate the conductive opacity
    kcond = 2.6e-7*charge*(T+2)/(rho**2)*(1 + (rho/2e6)**(2/3))
    # Calculate the total opacity
    if kcond == 0:
        kap = krad
    else:
        kap = 1/(1/krad + 1/kcond)
    # return opacity in SI units
    return kap/10
```