THE EFFECT OF CONVECTION ON CORE-COLLAPSE SUPERNOVAE

by

Yerassyl Telman

A Thesis Submitted to the Faculty of the

DEPARTMENT OF PHYSICS

In Partial Fulfillment of the Requirements
For the Degree of

BACHELOR OF SCIENCE

In the School of Science and Technology

NAZARBAYEV UNIVERSITY

2024

NAZARBAYEV UNIVERSITY, SCHOOL OF SCIENCE AND TECHNOLOGY

As members of the thesis committee, we certify that we have read the thesis prepared by Yerassyl Telman entitled

THE EFFECT OF CONVECTION ON CORE-COLLAPSE SUPERNOVAE

and recommend that it be accepted as fulfilling the thesis requirement for the degree of Bachelor of Science.

_____ Date: 28 April, 2024
Ernazar Abdikamalov

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to the Department of Physics.

I hereby certify that I have read this thesis prepared under my direction and recommend that it be accepted as fulfilling the thesis requirement.

_____ Date: 28 April, 2024
Thesis Director: Ernazar Abdikamalov

ABSTRACT

Modern Astrophysics, and the study of Core-Collapse Supernovae in particular, require an intense computational power and convenient code for different machines. A portable versatile code IDEFIX is able to deal with these issues. This thesis demonstrates a simulation of Core-Collapse Supernova with stable shock using IDEFIX.

The model has an outer accreting part and inner shock. The code provides an extensive possibilities to efficiently modify the model for specific needs.

Throughout the work, we obtain standing accretion shock instability (SASI) with 3D simulations, as well as observe how an outer convective shell with velocity perturbations affect the shock and SASI for different perturbation profiles. Additionally, we provide a way of analyzing those accreting perturbations which hints towards the future research of this topic.

Contents

1 INTRODUCTION

1.1 Parallel programming and Kokkos

Current astrophysical problems require complex computations and simulations of various processes, including hydrodynamical flows. Simulations of stellar evolution and core-collapse supernovae is one of such complex tasks that cannot be computed on a single computer effectively. Thus, there is a need for parallel programming with supercomputers and large computational nodes and an efficient portable code for solving those tasks on different machines [1], which is a big problem in computational astrophysics.

Computers perform tasks using central processing units (CPUs) and graphic processing units (GPUs). For an average computer, CPUs usually have 6-12 cores, while GPUs have around 1000-5000 cores. Although performance of GPUs and CPUs increase exponentially, a lot of modern astrophysical problems require extensive computation, so one way of speeding up the calculations is to parallelize them on several cores.

The main challenge of parallel programming is a deliberate control of memory and optimization in order to correctly delegate tasks between cores, which may be tedious for large codes. Also, the code must be specific for the supercomputer with many cores depending on its architecture and characteristics which additionally complicates the development and implementation of simulations.

To solve this issue, it is possible to use Kokkos metaprogramming library [2]. Kokkos has tools and algorithms to control memory and data transfer between cores such that different codes and applications written with Kokkos can be implemented on different machines.

1.2 Core-Collapse Supernovae

Core-Collapse Supernovae (CCSNe) occur during the explosion of a massive star with potential energy in the order of 10^{53} erg [3, 4]. There are several scenarios of how it happens, such as an electron capture, pair instability, iron core and so on. In general, the gravitational collapse of a star to Proto-Neutron Star (PNS) happens when an iron core becomes inert and ceases to generate nuclear energy. Then, the collapsing core bounces out of incompressible matter and creates a shockwave, which stalls after a few milliseconds due to the loss of kinetic energy, and revives for hundreds of milliseconds due to neutrino heating. During the shock revival there are inner cooling region and outer heating gain region that creates convections. This normally occurs in a region with equal heating and cooling rate called gain region.

One outcome of the stalled shock is the standing accretion shock instability (SASI). Such instability exhibits large scale oscillations of the shockwave. The emergence of SASI

can be explained by advective-acoustic cycle mechanism [5]. The mechanism of such cycle involves two waves: an entropy and voritcity advective wave, and an acoustic wave outwards. Advective waves towards the PNS causes an acoustic wave that propagates to the shockwave, which in turn creates another entropy and vorticity wave. The waves hitting the shockwave and PNS surface have amplitudes $\mathcal{L}_{sh}$ and $\mathcal{L}_{\nabla}$ respectively. Such cycle becomes unstable if $\mathcal{L}_{sh}\mathcal{L}_{\nabla} > 0$, leading to SASI behaviour.

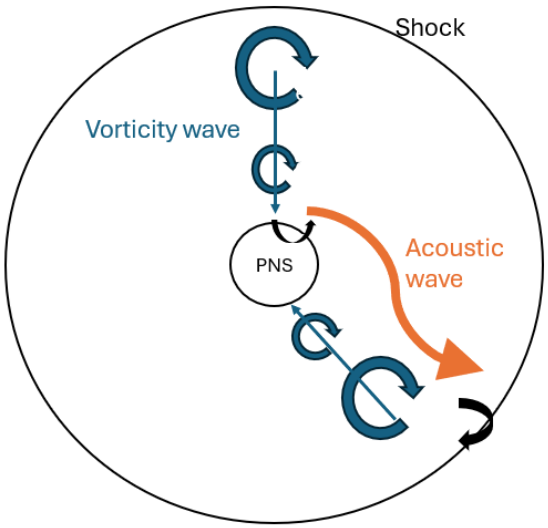


Figure 1: The diagram of the advective-acoustic cycle. Blue vorticity waves move towards a PNS and cause an acoustic wave to shock.

In this work, our task is to investigate the CCSN with their pre-collapse convection and the evolution of shock with an efficient portable code for parallel computing.

2 METHOD

2.1 IDEFIX Code

In this research work, we use new IDEFIX code developed by Lesur et al. [6, 7] to simulate core-collapse supernova. IDEFIX is a portable code that allows to solve various nonrelativistic hydrodynamical (HD) and magnetohydrodynamical (MHD) problems using Kokkos library. The code contains algorithms and modules to create astrophysical flows with realative ease. This section will describe its main tools and features.

2.1.1 Hydrodynamics and Solution

IDEFIX uses continuity and momentum equations:

$$\partial_t \rho + \nabla \cdot (\rho \vec{v}) = 0 \tag{1}$$

$$\partial_t(\rho\vec{v}) + \nabla \cdot (\rho\vec{v} \times \vec{v} + P\mathbb{I} + \Pi) = -\rho\nabla\psi \tag{2}$$

Here ρ is the density, P - pressure, $\vec{v}$ is the flow velocity, ψ is the gravitational potential, and $\mathbb{I}$ - identity matrix. The viscous stress tensor is given as:

$$\Pi = \eta_1(\nabla\vec{v} + (\nabla\vec{v})^T) + (\eta_2 - \frac{2}{3}\eta_1)\nabla \cdot \vec{v} \tag{3}$$

Here η_1 and η_2 are standard and bulk viscosities respectively.

IDEFIX starts the solution of HD equations by setting generalized conserved quantities $U = (\rho, P, \vec{v})$ for the density, pressure and velocity respectively as

$$\partial_t U + \nabla \cdot \vec{F} = S \tag{4}$$

In this equation, $\vec{F}$ is a flux of conserved quantities and S is a source term. The whole solution is based using time-average flux terms $\langle F \rangle$ to find average conserved quantities $\langle U \rangle$.

Different Riemann solvers can be used to solve the needed differential HD equation such as Russanov flux, Harten, Lax and van Leer (HLL), the HLLC, the approximate linear Roe Riemann solver and so on [6, 8]. All these types of Riemann solvers are by itself a complex topic of Computational Physics and lies outside the scope of this work.

2.1.2 Implementing Kokkos

As was stated above, Kokkos library makes the code portable for any device. In case of IDEFIX, any code run has a host as a CPU and device as either CPU or GPU. Kokkos has functions `IdefixArrayND` and `idefix_for` to initialize N-dimensional arrays and for loops on a device. As an example, a typical for loop in three dimensions look like this:

```
void Func(Hydro *hydro){
    IdefixArray4D<real> Vc = data->hydro-Vc;
    idefix_for("Func", 0,
        0,data->np_tot[KDIR],0,data->np_tot[JDIR],0,data->np_tot[IDIR],
        KOKKOS_LAMBDA (int k, int j, int i) {
            //Operations with Vc
    });
}
```

2.1.3 Output

IDEFIX gives output files in `.vtk` format (VTK - Visual Toolkit). It stores all the necessary data about density, pressure and velocity at certain timesteps specified in the input file. This file format can be opened and visualized using Paraview software.

2.1.4 Performance

Lesur et al. completed performance tests on CPUs and GPUs in comparison to another astrophysical code PLUTO [6, 9]. They did the comparison on the same 3D Orszag Tang problem. The performance test (cell/sec/nodes) on AMD Rome CPU demonstrates roughly similar performance compared to PLUTO at lower number of nodes, but IDEFIX is 20-30 % more efficient with over 10000 nodes. Parallel efficiency test shows over 80 % efficiency on CPUs with IDEFIX compared to 60 % with PLUTO code. In addition, IDEFIX reaches over 80 % and 95 % parallelization efficiency with 256^3 subdomains on Nvidia V100 and AMD Mi250 GPUs respectively.

2.2 Simulation Code Contents

Each simulation implemented in IDEFIX uses certain set of codes and modules that specifies the setup, inputs and outputs, and overall implementation of the simulation. The main parts of the needed simulations are in the "Test" directory by files `definitions.hpp`, `idefix.ini` and `setup.cpp`. This section will further elaborate on each of these files as described in the documentation of the code [7].

2.2.1 Definitions file

The `difinitions.hpp` is the main problem header file that defines the components, dimensions, and geometry.

Parameters define the number of vector components for the velocity and magnetic field (for MHD). The options are 1, 2, and 3.

Dimensions sets the number of physical dimensions of the system (1D, 2D and 3D). It is possible to have more components than dimensions.

Geometry sets the general geometry of the system as $(X1, X2, X3)$. The main options are:

1. `CARTESIAN`: A Cartesian coordinate system $(X1, X2, X3) = (x, y, z)$.
2. `CYLINDRICAL`: A cylindrical coordinate system $(X1, X2, X3) = (R, z, \varphi)$. It is used only for 2D cylindrical problems.
3. `POLAR`: A polar coordinate system $(X1, X2, X3) = (R, \varphi, z)$ for 3D cylindrical problems.
4. `SPHERICAL`: A spherical coordinate system $(X1, X2, X3) = (R, \theta, \varphi)$ for all three dimensions.

2.2.2 Setup file

`setup.cpp` contains the code for the physical implementation of the simulation. It gets the initial parameters, arrays and variables, specific functions and outputs. This subsection

will explain some of its features with examples.

Firstly, the `setup` class defines all initial conditions and constructors of the code from an input file. The `setup` constructor of this class is called during the setup to set `Input`, `Grid`, `DataBlock` and `Output` objects. `Input` is used for user-defined parameters. For example, the parameter `rsh` for the initial shock radius can be set as:

$$\texttt{rsh} = \texttt{input.Get} < \texttt{real} > (\text{"}\texttt{Setup}\text{"}, \text{"}\texttt{Rshock}\text{"}, \texttt{ZERO\_F});$$

In addition to this, it allows to enroll a user-defined function so there won't be a need to define lots of empty functions. As an example, if there is a specific user-defined boundary condition function, it must be enrolled as:

$$\texttt{data.hydro} - > \texttt{EnrollUserDefBoundary}(\&\texttt{UserdefBoundary});$$

Next, the `InitFlow` is a method in `Setup` class that defines the initial conditions of the hydrodynamical flow with `Vc` array containing values for density, pressure, and velocities. For instance, in case of a density equal to `rho`, pressure equal to `p`, radial velocity of `vx` and zero angular velocities, the initial flow is written as:

```
void Setup::InitFlow(DataBlock &data) {
    DataBlockHost d(data); // Create a host copy
    for(int k = 0; k < d.np_tot[KDIR] ; k++) {
        for(int j = 0; j < d.np_tot[JDIR] ; j++) {
            for(int i = 0; i < d.np_tot[IDIR] ; i++) {
                d.Vc(RHO,k,j,i) = rho;
                d.Vc(PRS,k,j,i) = p;
                d.VC(VX1,k,j,i) = vx;
                d.Vc(VX2,k,j,i) = ZERO_F;
                d.Vc(VX3,k,j,i) = ZERO_F;
            }
        }
    }
    d.SyncToDevice();
}
```

In the setup file, a user-defined boundary condition can be defined by enrolling `UserdefBoudary` function. Source term for energy, density, pressure or for any other parameter can be specified by `MySourceTerm` function.

2.2.3 Input file

`idefix.ini` is the default problem input file. It contains various entry values for each class of the simulation in the setup file. Those values are separated in each section of

the file, and is read at the start as was shown in the section 2.2.2. This subsection will elaborate on some of entries in the input file.

`Grid` section states the grid size of the problem in 1/2/3 dimensions, domain of each dimension, number of grid points and spacing between grid points, mainly uniform (`u`) or logarithmic (`l`) depending the coordinate as $x_{i-1/2} = x_{start}\alpha^{i/N}$ where $\alpha = \frac{x_{end}+|x_{start}|-x_{start}}{x_{start}}$. As an example, logarithmic scaling in X1 from 0.4 to 10.0, and uniform in X2 and X3 from 0 to π and 2π respectively is written as follows:

```
[Grid]
X1-grid    1  0.4  300 l  10.0
X2-grid    1  0.0  48  u  3.141592653589793
X3-grid    1  0.0  96  u  6.283185307179586
```

Here X1/X2/X3 specifies the coordinate, "1" is the number of grid blocks in the corresponding coordinate and 500/48/96 is the number of grid points.

`TimeIntegrator` section defines the variables for the time integrator method. This section allows to choose, for instance, value of the first time step with `first_dt` variable, and the time at which the run stops with `tstop` variable. Other than that, it also specifies more specific details like number of integration stages (1st order Euler or 2nd/3rd order Runge-Kutta) and CFL value for numerical stability.

`Hydro` section defines the hydrodynamic parameters of the problem. Main two parameters are a solver for hydrodymanic equations and adiabatic constant γ. For example, HLL solver and $\gamma = 4/3$ is defined as:

```
[Hydro]
solver    hll
gamma     1.333333333333333333
```

In addition, this section allows to switch on a specific type of diffusion (Ohmic diffusion, ambipolar diffusion), switch on Hall effect, rotation in z axis etc for specific problems.

`Setup` section defines variables for the setup class of the code for user-defined functions, initial flow, source term and so on. The variables in this section are specific for each type of HD problem.

`Boundary` section specifies the boundary conditions in the beginning and end of each coordinate as `X1 − beg`, `X1 − end`, `X2 − beg`, `X2 − end`, `X3 − beg`, `X3 − end`. The class has several options for boundary conditions:

- `periodic` copies beginning and end periodically as in full circle.
- `axis` defines an axis in spherical geometry for θ from 0 to π.
- `reflective` explicitly reverses normal component of velocity.

- `userdef` sets user-defined boundary condition if it is enrolled in the setup file.

`Output` section defines how the code gives output files. Mainly it can define the time interval between log outputs and VTK outputs in code time units as:

```
[Output]
vtk    5.0e0
log    1000
```

2.2.4 Code configuration

The parallelization can be done using either Message Passing Interface (MPI) or Open Multi-Processing (OpenMP) application programming interface (API). In order to run the simulation either on CPU or GPU, the code must be configured using `Cmake` software. By default, it can be configured with command `cmake` `$IDEFIX_DIR` such that it runs on CPU in series. However, it is possible to run in parallel with OpenMP interface by adding $-$`DKokkos_ENABLE_OPENMP` $=$ `ON` and specifiying number of CPU threads with `export` `OMP_NUM_THREADS` $=$ `N`.

In order to run the code on GPUs, MPI interface can be enabled by adding $-$`DIdefix_MPI` $=$ `ON`. Also, `Cmake` has various commands to choose GPU architecture, enable CUDA API and so on. In this work, we run simulations with only OpenMP.

2.3 Simulation Model

In our simulation, we use a model with two parts: outer subsonic part with Bondi accretion and inner supersonic part with stationary shock.

2.3.1 Stationary shock

The model of the stationary shock is described in Fernandez & Thompson's work [10]. The stationary shock is created with a pre-shock Mach number of $\mathcal{M}_1 = 5$ and the initial shock radius ($r = r_{sh}$). The system uses hydrodynamic equations of state (1)-(3) for an ideal gas. The adiabatic index $\gamma = 4/3$ and the the radius of proto-neutron star (PNS) is defined $r_{pns} = 0.4$.

The nuclear dissociation energy ε contributes to the conservation of energy at $r = r_{sh}$ as:

$$\frac{\upsilon_1^2}{2} + \frac{\gamma}{\gamma - 1}\frac{p_1}{\rho_1} = \frac{\upsilon_2^2}{2} + \frac{\gamma}{\gamma - 1}\frac{p_2}{\rho_2} + \varepsilon \tag{5}$$

In our model, we simplify this by taking $\varepsilon = 0$.

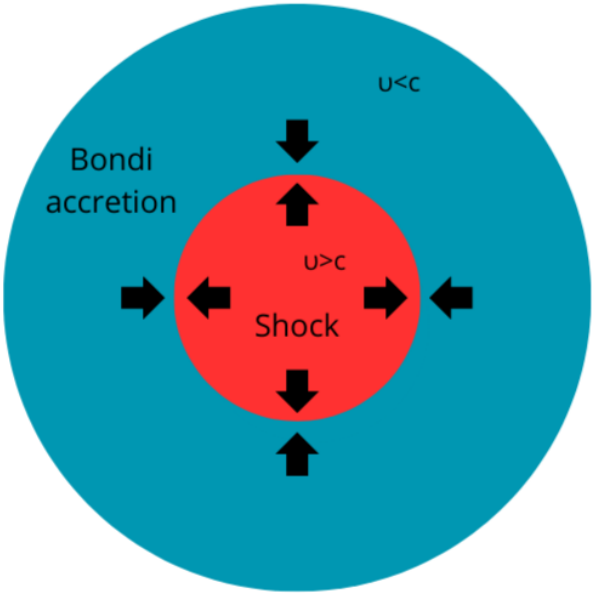


Figure 2: Diagram of the model with supersonic shock inside Bondi accretion outside

The shock compression factor is defined as $\kappa = \rho_2/\rho_1$ and for $\varepsilon = 0$ can be calculated from pre-shock Mach number:

$$\kappa = (\gamma + 1)\left(\gamma - 1 + 2\mathcal{M}_1^{-2}\right)^{-1} \tag{6}$$

For $\gamma = 4/3$ this becomes $\kappa = \frac{7}{3}\frac{\mathcal{M}_1^2}{\mathcal{M}_1^2/3+2}$. This value is the factor by which shockwave changes density and velocity before and after the shock. It is defined by `ShockCompressionFactor` function in the code.

The source term in the equation (4) is represented as the neutrino heating Q_ν [11]:

$$Q_\nu = \left(\frac{B}{r^2} - Ap^{3/2}\right) exp(-(s/s_{min})^2) \tag{7}$$

Here A is the cooling normalization, B is the heating normalization, s is the entropy defined as:

$$s = \frac{1}{\gamma - 1} ln\left[\frac{p}{p_2}\left(\frac{\rho_2}{\rho}\right)^\gamma\right] \tag{8}$$

Where p_2 and ρ_2 are initial pressure and density at $r = r_{sh}$. s_{min} is the minimal entropy at $r = r_{sh}$, so its bigger absolute value leads to less cooling. Also, cooling is corrected by cooling correction factor as $(4\pi\upsilon_1)/(r_{pns}\Delta E_c)$ where ΔE_c is the total change of energy due to cooling. This factor is needed to stabilize the computation of cooling rate.

The values of A, B and s_{min} are defined in the input and setup files as `Acool`, `Aheat` and `smin` respectively. The equation (7) is implemented in `MySourceTerm` function by changing variable `Uc(ENG,k,j,i)` for energy.

The shockwave further evolves and has two stable outcomes: convection-dominated and SASI-dominated sequnce. The two outcomes can be quantified by a convection parameter χ [12]:

$$\chi = \int_{r_g}^{r_{sh}} \frac{|\omega_{BV}|}{|v_r|} dr \tag{9}$$

ω_{BV} is the Brunt–Väisälä frequency expressed in terms of gravitational acceleration g and entropy s as:

$$\omega_{BV} = \sqrt{\frac{\gamma - 1}{\gamma} g \nabla s} \tag{10}$$

$\chi < 3$ are stable and usually leads to SASI-dominance with $\varepsilon = 0$, and $\chi > 3$ leads to convection-dominance.

2.3.2 Bondi accretion

The outer part above the shock is simulated in accordance with Bondi accretion model. The model describes the accretion of a gas cloud of mass M with spherical symmetry into some compact object [13]. The solution of the problem gives us the expression for steady accretion rate $\dot{M}$, gas velocity and how the gas is affected. This section provides a solution of Bondi accretion problem from the textbook by Frank, King and Raine [14].

First of all, with spherical symmetry and steady flow, the continuity equation (1) becomes:

$$\frac{1}{r^2} \frac{d}{dr}(r^2 \rho v) = 0 \tag{11}$$

From this, $r^2 \rho v = const$. Here, $-\rho v$ is the inward flux of the matter, so the continuous influx rate is;

$$\dot{M} = -4\pi r^2 \rho v \tag{12}$$

As the gravitational force $F = -GM\rho/r^2$ is only in radial direction, the Euler equation can be written as:

$$v \frac{dv}{dr} + \frac{1}{\rho} \frac{dP}{dr} + \frac{GM}{r^2} = 0 \tag{13}$$

Since the speed of sound is defined as $c = \sqrt{dP/d\rho}$, the gradient of pressure is

$$\frac{dP}{dr} = \frac{dP}{d\rho}\frac{d\rho}{dr} = c^2\frac{d\rho}{dr}$$

This combined with equation (11) changes the equation (13):

$$v\frac{dv}{dr} - \frac{c^2}{vr^2}\frac{dv}{dr} + \frac{GM}{r^2} = 0$$

Finally, by rearranging terms we get:

$$\frac{1}{2}\left(1 - \frac{c^2}{v^2}\right)\frac{d}{dr}\left(v^2\right) = -\frac{GM}{r^2}\left(1 - \frac{2c^2 r}{GM}\right) \tag{14}$$

The right hand side of the equation (14) must be negative at large distance, and increases as r decreases. Hence, the right hand side must be zero at some distance r_s equal to:

$$r_s = (GM) \,/\, \left(2c^2(r_s)\right) \cong 7.5 \times 10^{13}\, cm \tag{15}$$

In general, equation (14) has six types of solutions for subsonic and supersonic speeds depending on r_s and $\frac{d}{dr}(v^2)$. Type 1 and 2 correspond to $c(r_s) = v(r_s)$ and either $v^2 \to 0$ as $r \to \infty$ or $v^2 \to 0$ as $r \to 0$, so these two types make a transition between supersonic and subsonic velocities at $r = r_s$. Type 3 and 4 are $\frac{d}{dr}(v^2) = 0$ at $r = r_s$ and either $v^2 < c^2$ or $v^2 > c^2$ everywhere, so always in subsonic or supersonic region. Lastly, types 5 and 6 are similarly $\frac{d}{dr}(v^2) = \infty$ at $v^2 = c^2(r_s)$ and always in the region $r > r_s$ or $r < r_s$.

For our model, we are interested in the type 3 solution which is solved as the following equation:

$$\frac{1}{2}v^2 - c^2 \ln v^2 - 2c^2 \ln r - \frac{GM}{r} = 0 \tag{16}$$

The Figure 3 visualizes all types of solutions.

3 RESULTS

3.1 Stabilizing the Shock

Before doing proper 3D simulations, we must get configurations for stable shockwave in the first place. For practical reasons, it is better to find needed parameters and initial values by doing 1D simulation first.

Initially, 1D run with `aheat` = `0.0493818` and reflection coefficient of 1 shows rapid increase of shock radius from $R_{sh} = 3.0$ to $R_{sh} = 14.0$ over 20 time steps.

Thus, our first task is to make shock radius stable in first 10-20 timesteps in 1D. There are several parameters that can be changed to affect shock radius and its evolution: cooling correction, separate heat value for the source term, s_{min} entropy, and reflection coefficient.

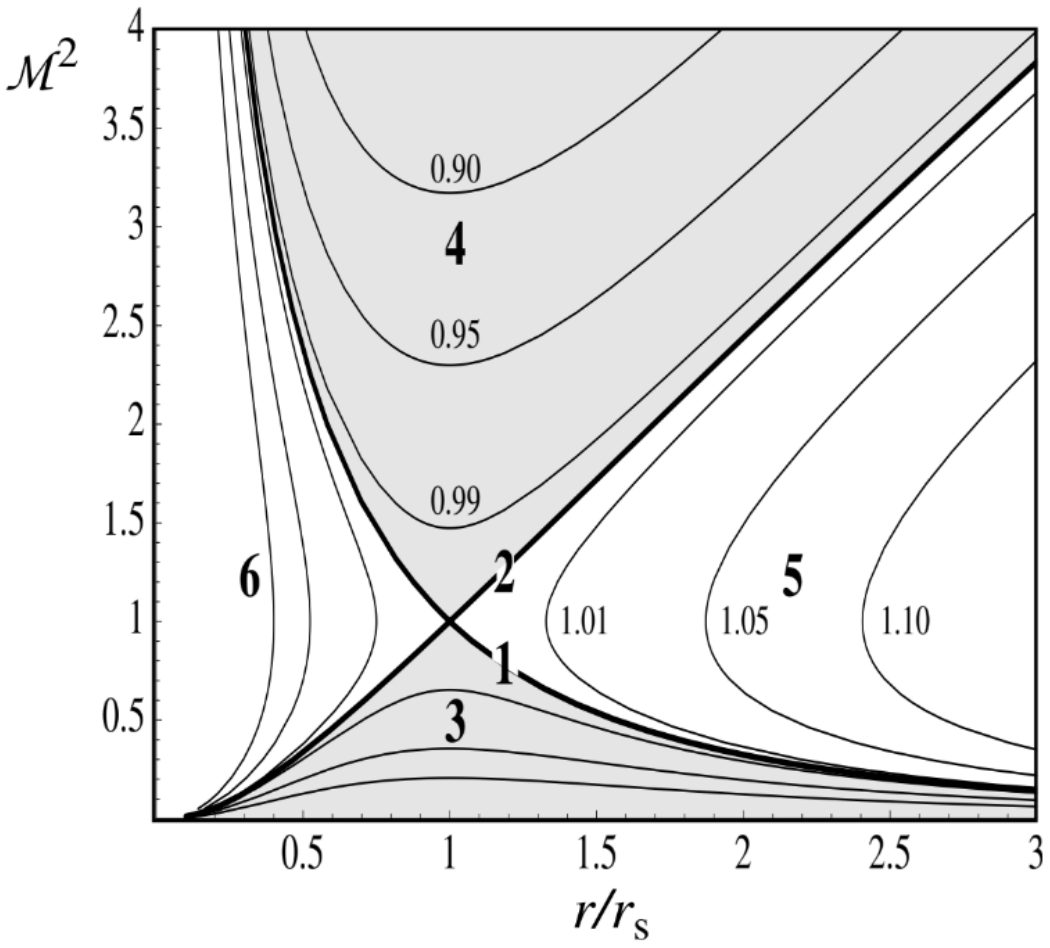


Figure 3: Mach number squared vs r/r_s. Type 3 solution from the equation (14) is shown along with other types of solutions. The graph is taken from the textbook by Frank, King and Raine (2002) [14]

The cooling correction term is by default defined as $(4\pi v_1)/(r_{pns}\Delta E)$, where ΔE is the total energy change due to cooling. This term is needed to stabilize the numerical solution by multiplying it to cooling normalization A from equation (7), and its numerical value is ~ 0.03. Nevertheless, it is possible to set cooling correction to higher constant value such as 1.0, which increases total cooling and decreases outflow.

Another way of decreasing net heating is by introducing a separate heat normalization B from equation (7). In the code, it is possible to change `Aheat` with other variable in the function `MySourceTerm`, so there will be different heat normalization for the initial condition and shock evolution. Since our initial `Aheat` = `0.0493818`, we can set separate heat value $= 0.0 - 0.04$ to decrease heating.

s_{min} is the minimum entropy as shown in equation (7), which prevents runaway cooling. Its smaller absolute value corresponds to more cooling. While initially $s_{min} \sim -20.0$, we can set it to ~ -5.0 to achieve less net heating.

Lastly, it is possible to alter inner boundary condition. The `UserdefBoudary` function defines inner boundary condition for v_r velocity as $v_{ref} = -v_r$. But, the reflection can be changed by changing its coefficient as $v_{ref} = -C_{ref} v_r$. The coefficient C_{ref} here can be varied from 0 to 1. Smaller values correspond to less flux outwards which may prevent blow out of the shock wave.

The figure 4 shows how implementation of these methods decrease the shock radius over time compared to initial values. While some parameters still shows constant increase, some on the contrary shrinks the shock radius.

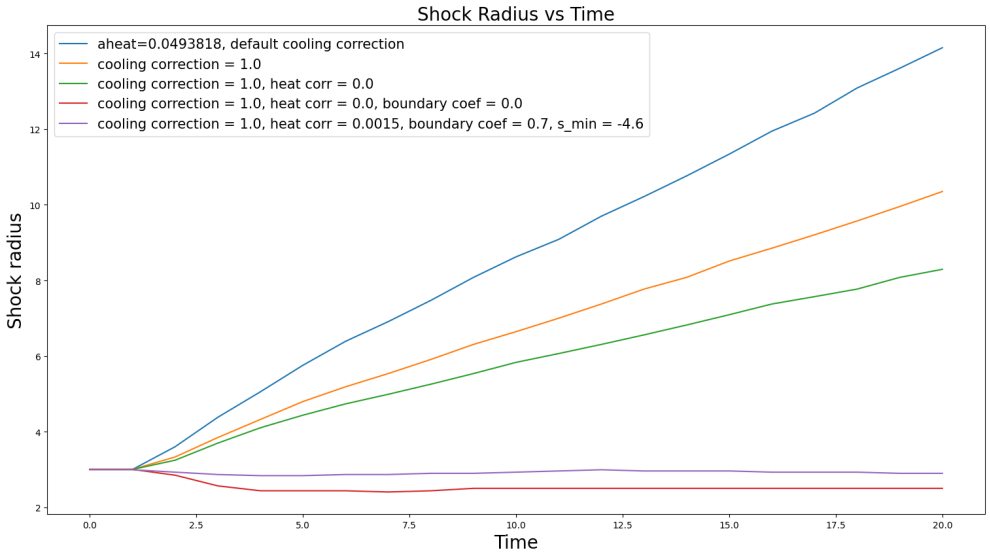


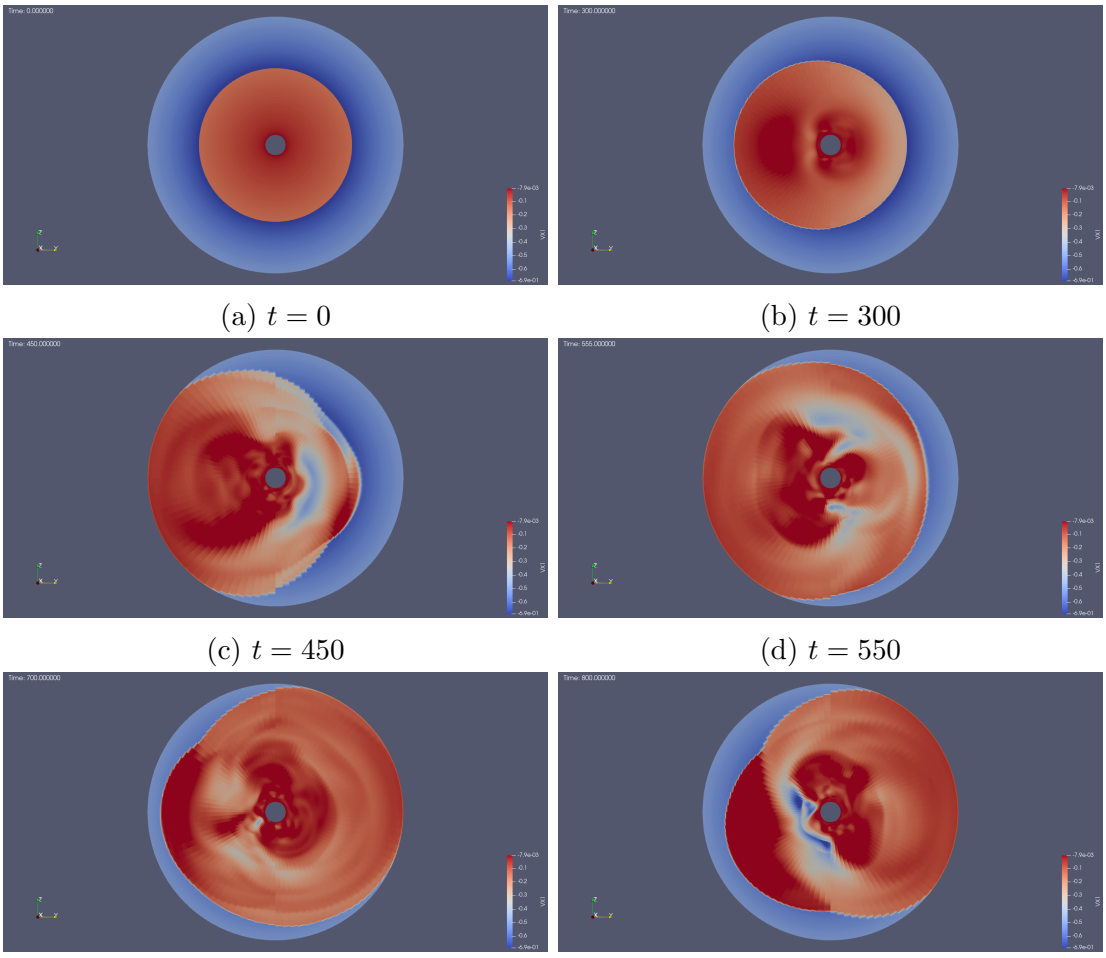
Figure 4: Shock radius over time with different parameters of cooling, heating and boundary condition

A special script was written to find the most optimal parameters. With cooling correction of 1.0, the script runs the simulation in 1D with changing s_{min} from -4.8 to -3.0, heat from 0.001 to to 0.04, and reflection coefficient from 0.0 to 1.0. In total, over 800 quick runs were done, and the script records the ratio of initial and final shock radius and the ratio of initial and final velocity at $r = r_{pns}$. The goal was to find the parameters that result in ratio of 1 for shock radius and v_{pns} closest to 0. After that, it resulted in the clear visualization that shows best parameters.

In the end, the best values ended up to be $\texttt{s}_{\texttt{min}} = -4.6$, $\texttt{heat} - 0.0015$, and reflection coefficient of 0.7.

3.2 3D simulation

After finding the right parameters for stable shockwave, we can perform 3D runs. IDEFIX allows to make multi-dimensional runs simply by setting number of grid blocks in



(a) $t = 0$ (b) $t = 300$

(c) $t = 450$ (d) $t = 550$

(e) $t = 700$ (f) $t = 800$

Figure 5: SASI over time = 800

azimuthal and polar coordinates. For our run, we set it to 48 and 96 for θ and φ angles respectively.

The convection parameter from the eq. (9) is $\chi \sim 0.1$ for these parameters with low heating, so SASI-dominance is expected.

In order to obtain perturbations faster, we can add random perturbations to the density in the `InitFlow` function as $\rho = \rho_0 \pm \delta\rho = \rho_0 \pm 10^{-2}\xi\rho_0$ where ξ is a random number from 0 to 1. The output is every $\Delta t = 5.0$ and the simulation is run up to $t_{stop} = 800$. After the extensive simulation, we obtain the result that shows the SASI behavior starting from around $t = 200$. The shockwave oscillates along y axis with the period of ~ 50. Without random density perturbations, the oscillations start at $t \sim 750$. The images below show the radial velovity as slice in YZ plane at times $t = 0$; 300; 450; 550; 700; 800. The Figure 5 shows the perturbation of the shockwave. This result is a good example of SASI, so we can further modify the model with velocity perturbations.

3.3 Adding Velocity Perturbations

Various studies show that asphericities of the progenitor affect the supernova core-collapse and specifically SASI by causing more aspherical motions in the post-shock region [15, 16]. These asphericites are presented as convective shells with solenoidal velocity perturbations with condition $\nabla \cdot (\rho\vec{v}) = 0$ and Mach number $\mathcal{M}_{prog} \sim 0.1$ [17]. The dominant wave number l can have different values.

The velocity perturbation δv can be added to the initial velocity in some outer (Bondi accretion) region $r_{min} \leq r \leq r_{max}$. The perturbation is defined from the stream function $\vec{\psi}$ as:

$$\delta\vec{v} = \frac{C}{\rho}\nabla \times \vec{\psi} \tag{17}$$

Here C is a dynamic viscosity value and the stream function is defined as:

$$\vec{\psi} = \vec{e_\varphi}\frac{\sin\theta}{r}\sin n\pi\frac{r - r_{min}}{r_{max} - r_{min}}Y_{l,1}(\theta, 0) \tag{18}$$

$$Y_{l,m}(\theta, \varphi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)e^{im\varphi} \tag{19}$$

n and l are number of perturbation cells in radial and angular direction, and P_l^m is associated Legendre polynomial. Spherical harmonic is chosen to be $Y_{l,1}$ so there is no singularity at $\theta = 0$ and $\theta = \pi$. For our model, we are interested in perturbations with $n = 1$ and $l = 1;\ 2;\ 4$. We choose dynamic viscosity value such that maximum Mach number of the convective layer is $\delta v/c \sim 0.1$. Calculating equations (17), (18) and (19) gives expressions for radial and angular velocities as follows:

$$\delta v_r^{l=1} = -\frac{5}{4r^2}\sqrt{\frac{3}{\pi\sin^2\theta}}\cos\theta\sin^{\frac{3}{2}}(\theta)\sin\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{20}$$

$$\delta v_\theta^{l=1} = -\frac{\sqrt{3\pi\sin^3(\theta)}}{2r(r_{min} - r_{max})}\cos\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{21}$$

$$\delta v_r^{l=2} = -\sqrt{\frac{15}{2\pi\sin^2\theta}}\frac{(3 + 7\cos(2\theta))\sin^{\frac{3}{2}}(\theta)}{8r^2}\sin\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{22}$$

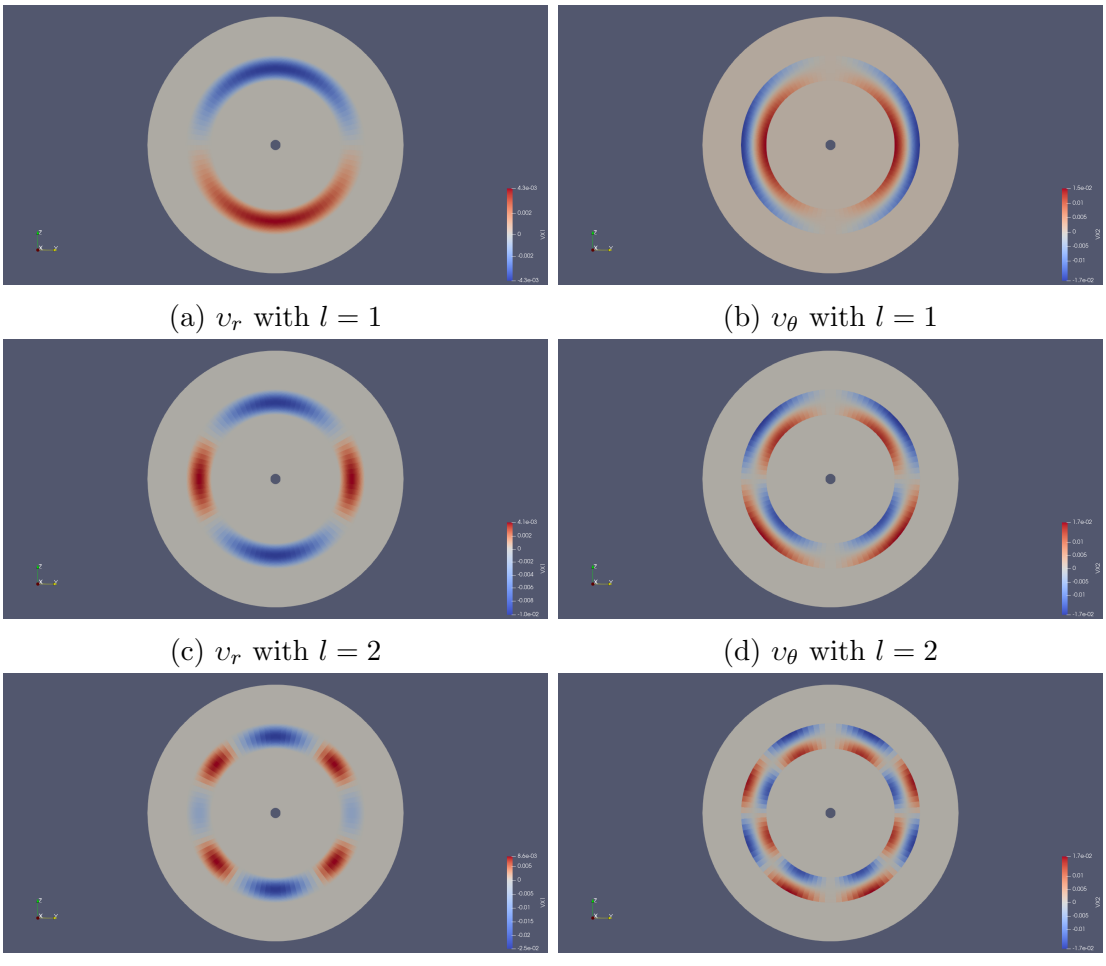
$$\delta v_r^{l=2} = -\sqrt{\frac{15\pi\sin^3(\theta)}{2}}\frac{\cos\theta}{2r(r_{min} - r_{max})}\cos\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{23}$$

$$\delta v_r^{l=4} = -\sqrt{\frac{5}{\pi\sin^2\theta}}\frac{3(27 + 56\cos(2\theta) + 77\cos(4\theta))\sin^{\frac{3}{2}}(\theta)}{128r^2}\sin\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{24}$$

$$\delta v_\theta^{l=4} = \frac{3\sqrt{5\pi \sin^3(\theta)}\cos\theta(3 - 7\cos^2(\theta))}{8r(r_{min} - r_{max})} \cos\left(\frac{\pi(r - r_{min})}{r_{max} - r_{min}}\right) \tag{25}$$

The perturbations are on r and θ planes, while $v_\varphi = 0$ for any l and n.

We can add such velocity perturbations in the `InitFlow` function by changing `d.Vc(VX1,k,j,i)` and `d.Vc(VX2,k,j,i)` variables in a certain region of radius. The Figure 6 show the radial and azimuthal velocity perturbation slice in the region $5.0 \leq r \leq 7.0$.



(a) v_r with $l = 1$ (b) v_θ with $l = 1$

(c) v_r with $l = 2$ (d) v_θ with $l = 2$

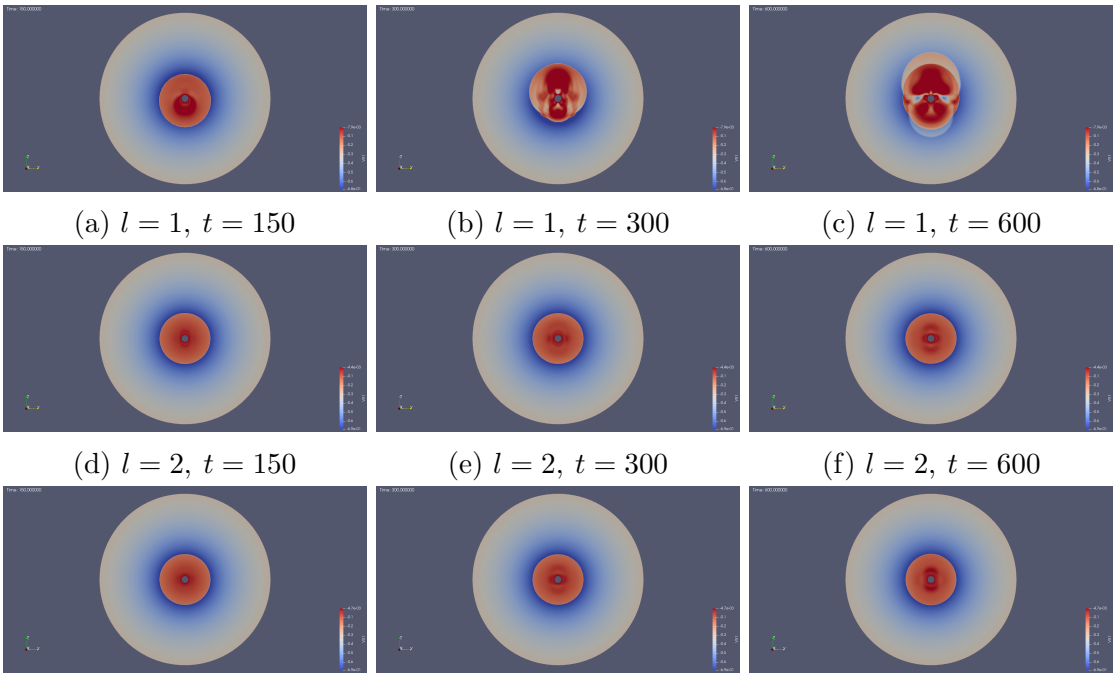
(e) v_r with $l = 4$ (f) v_θ with $l = 4$

Figure 6: Radial and azimuthal velocity perturbations with $l = 1;\ 2;\ 4$

$l = 1$ perturbation demonstrates the occurrence of SASI starting from $t \sim 110$ along Z axis, which is much earlier than without perturbations. Contrary to this, $l = 2$ and $l = 4$ perturbations result in stable shockwave even up to $t = 600$ as shown in in Figure 7.

3.4 Getting Convection Instead of SASI

SASI is expected for cases with lower heating. We observed this in all previous examples because there was a separate small heating for evolution. Despite this, it is possible to get



(a) $l = 1,\ t = 150$ (b) $l = 1,\ t = 300$ (c) $l = 1,\ t = 600$

(d) $l = 2,\ t = 150$ (e) $l = 2,\ t = 300$ (f) $l = 2,\ t = 600$

(g) $l = 4,\ t = 150$ (h) $l = 4,\ t = 300$ (i) $l = 4,\ t = 600$

Figure 7: $l = 1;\ 2;\ 4$ perturbations at $t = 150;\ 300;\ 600$

stable shockwave with more heating, and subsequently less cooling. More heating may result in random convections instead of SASI.

By varying cooling correction, it was possible to get a stable shock without separate heat value i.e. `aheat` = `0.0493818` during the evolution. To achieve this, we must set the reflection coefficient to zero and cooling correction to 0.5. This results in stable shock in 1D. With this value of heat convection parameter is $\chi \approx 5.68$, so convection-dominance is expected instead of SASI [12].

Nonetheless, 3D simulation shows SASI similar to previous results, and not convection that can be expected with higher heating.

One possible solution to get convection was by setting the heating at the inner boundary to zero so the matter accretes less in the center. This can be done by changing the source term Q(eq. (7)):

$$Q = Q\left[1 - \exp\left(-10^5(r - r_{pns})^2\right)\right] \tag{26}$$

The function $\exp\left(-10^5(r - r_{pns})^2\right)$ very quickly decreases at $r = r_{pns}$. Subtracting the source term from itself at the center also stabilizes the shockwave, but 3D run once again shows only SASI.

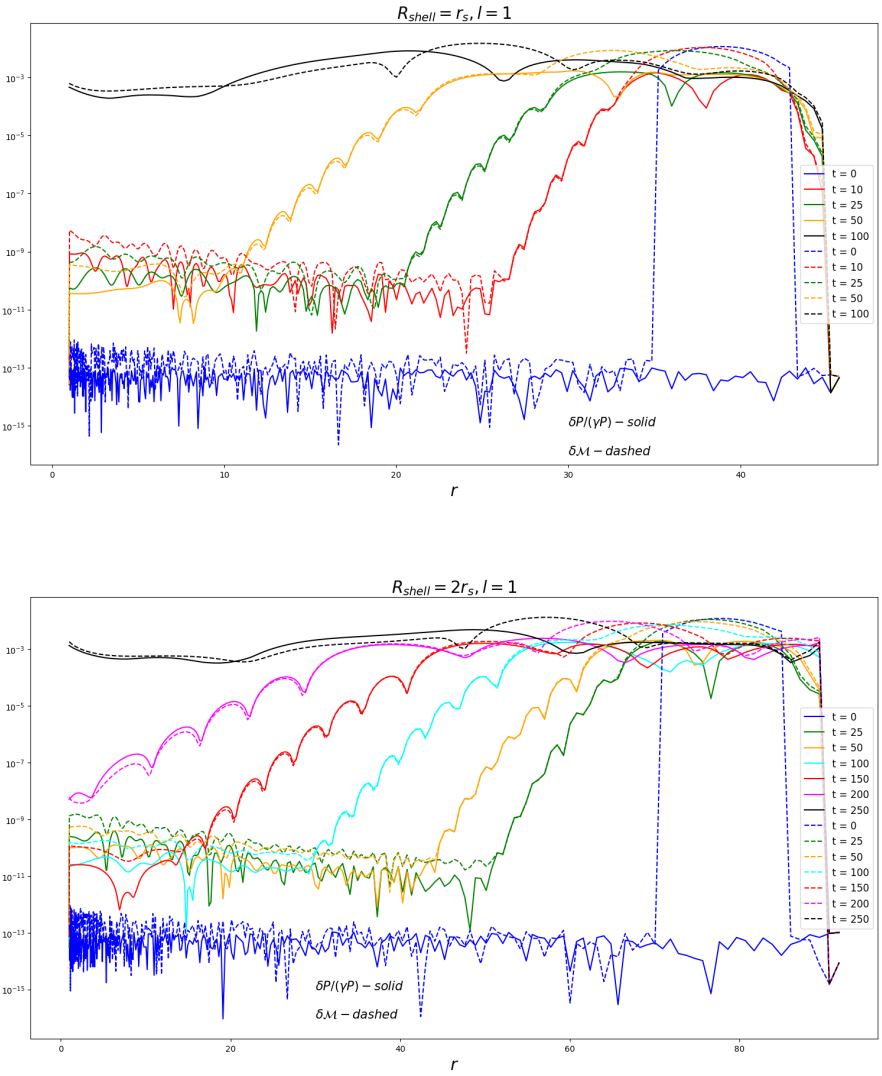


Figure 8: $\delta P/(\gamma P)$ and $\delta\mathcal{M}$ at various times for $R_{shell} = r_s,\ 2r_s, l = 1$

3.5 Analyzing the Accretion of the Velocity Perturbation

We have done multiple CCSN simulations that showed SASI dominance with convective shells and sufficient heating. Now, it is possible to further investigate the effect of velocity convections on the shockwave using IDEFIX code. This is achieved by changing the simulation code to contain only Bondi accretion. The inner boundary is changed from 0.4 (PNS radius) to 1.0 (shock radius) accordingly and the boundary condition is changed to outflow.

Prior to the shock, the acoustic perturbations have a dependence [18]:

$$\delta P/P \sim \delta\rho/\rho \sim \delta\mathcal{M} \tag{27}$$

Here, δP, $\delta\rho$ and $\delta\mathcal{M}$ are deviations of pressure, density and Mach number from their angle average values $\langle P\rangle$, $\langle\rho\rangle$, $\langle\mathcal{M}\rangle$ as:

$$\delta P = \langle\langle P\rangle - P\rangle$$

$$\delta\rho = \langle\langle\rho\rangle - \rho\rangle$$

$$\delta\mathcal{M} = \langle\langle\mathcal{M}\rangle - \mathcal{M}\rangle$$

We place velocity perturbations as described in the section 3.3 at distances $R_{shell} = r_s$ and $R_{shell} = 2r_s$, where $r_s = 39.0$ is a sonic radius ($\mathcal{M} = 1$). The width of these perturbations are from $0.9R_{shell}$ to $1.1R_{shell}$. During the simulations, quantities δP, $\delta\rho$, $\delta\upsilon_r$, $\delta\upsilon_\theta$ and $\delta\mathcal{M}$ are recorded along with angle averaged values of the pressure, density and velocities for further analysis.

The Figure 8 shows the values of $\delta P/(\gamma P)$ and $\delta\mathcal{M}$ for $l = 1$ and $R_{shell} = r_s$; $2r_s$. During the accretion, the identity (27) is visible, but over time both graphs stretch and start increasing to $\sim 10^{-3}$.

With this method, the values of $\delta P/(\gamma P)$, $\delta\upsilon_r/c$ and $\delta\upsilon_\theta/c$ can be analyzed during the accretion to check and compare with previous semi-analytical works of such model. These values are plotted in graphs shown in Figure 9. One noticeable feature is that $l = 1$ values are mostly straight line, whereas $l = 2$ have a significant dip, and $l = 4$ have several dips. At $R_{shell} = 3r_s$ number of such dips increase. On top of that, these results show increasing $\delta P/(\gamma P)$, while semi-analytical solutions predicted a plateau [18].

The question of why these patterns occur is a topic of further research. Future goal is to systematically study the accretion from various distances ($R_{shell} = 5r_s$; $10r_s$ as an example) and effectively visualize pressure and velocities to qualitatively analyze the occurrence of those dips.

4 CONCLUSION

In this thesis, we used new portable Astrophysical flow code IDEFIX designed for high-performance computing. The code solves solves conserved quantities and source terms specified in the `setup.cpp`. The variety of classes and features of the input file and setup file allows to create various astrophysical code for specific needs.

We used a code for Core-Collapse Supernovae that has outer accreting part and inner shockwave. 1D simulations were done to find the parameters of minimal cutoff entropy, heating normalization, and reflection coefficient of the inner boundary condition. The most optimal parameters were proven to be $s_{min} = -4.6$, $C_{ref} = 0.7$, `aheat` = `0.0015`

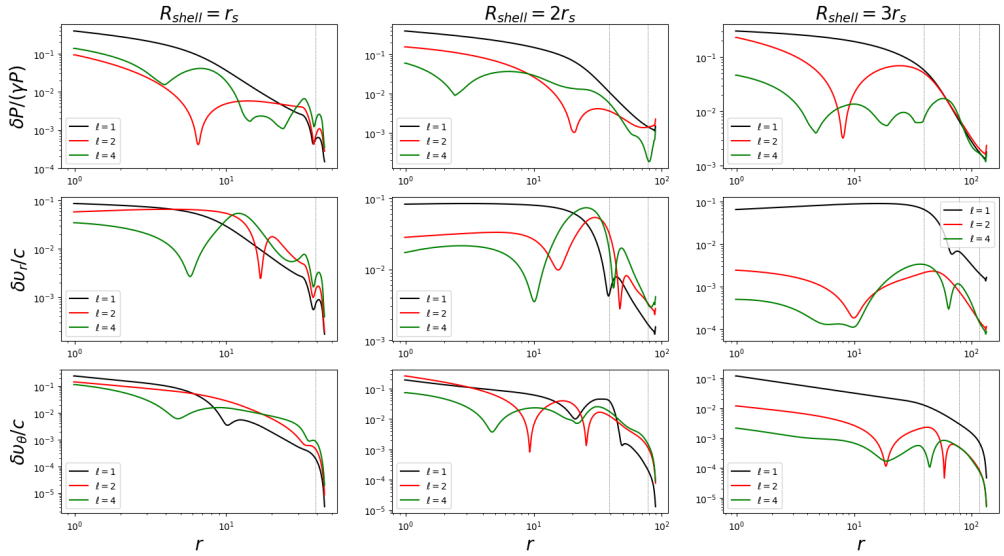


Figure 9: Top row: $\delta P/(\gamma P)$ at $R_{shell} = r_s;\ 2r_s;\ 3r_s$. Middle row: $\delta v_r/c$ at $R_{shell} = r_s;\ 2r_s;\ 3r_s$. Bottom row: $\delta v_\theta/c$ at $R_{shell} = r_s;\ 2r_s;\ 3r_s$

during evolution of the shock and cooling correction of 1. Alternatively, for constantly high `aheat` $=$ `0.0493818`, $C_{ref} = 0.0$ and cooling correction of 0.5 also gives a stable shock. Despite the first parameters predicting SASI ($\chi < 3.0$), and the second one convection ($\chi > 3.0$), 3D simulations of both resulted in standing accretion shock instability.

The effect of convective shells with angular numbers $l = 1;\ 2;\ 4$ on shock was also demonstrated. $l = 1$ perturbations speed up the occurrence of shock oscillations by ~ 6 times, and $l = 2$ and $l = 4$ perturbations did not result in SASI nor convection up to $t = 600$.

Further analysis of such perturbations can be done with pure Bondi accretion by analyzing the data at shock radius. The accretion of perturbations show a clear relationship between the deviation of Mach number and pressure. Consequent analysis of the accretion is a promising topic of further quantitative research.

To sum up, IDEFIX is an efficient code for doing CCSN simulations with relatively smaller computational power. The versatility and portability of the code makes the development and changes in the model convenient, which allowed us to obtain new results.

Bibliography

[1] B. Barney and D. Frederick, "Introduction to Parallel Computing Tutorial," .

[2] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, "Kokkos 3: Programming Model Extensions for the Exascale Era," IEEE Transactions on Parallel and Distributed Systems **33**, 805 (2022).

[3] H.-T. Janka, "Explotion Mechanisms of Core-Collapse Supernovae," Annual Reviews **62**, 407 (2012).

[4] B. Müller, "Hydrodynamics of core-collapse supernovae and their progenitors," Living Reviews in Computational Astrophysics **6** (2020), https://doi.org/10.1007/s41115-020-0008-5.

[5] J. Guilet and T. Foglizzo, "On the linear growth mechanism driving the standing accretion shock instability," Royal Astronomical Society **421**, 546 (2012).

[6] G. Lesur, S. Baghdadi, G. Wafflard-Fernandez, J. Mauxion, C. Robert, and M. Van den Bossche, "IDEFIX: A versatile performance-portable Godunov code for astrophysical flows," Astronomy & Astrophysics **667** (2023), https://doi.org/10.1051/0004-6361/202346005.

[7] G. L. et al., "Idefix code documentation," (2021).

[8] E. Toro, "Chapter 2 - The Riemann Problem: Solvers and Numerical Fluxes," in *Handbook of Numerical Methods for Hyperbolic Problems*, Handbook of Numerical Analysis, Vol. 17, edited by R. Abgrall and C.-W. Shu (Elsevier, 2016) pp. 19–54.

[9] A. Mignone, G. Bodo, S. Massaglia, T. Matsakos, O. Tesileanu, C. Zanni, and A. Ferrari, "PLUTO: A NUMERICAL CODE FOR COMPUTATIONAL ASTROPHYSICS," The Astrophysical Journal Supplement Series **170**, 228 (2007).

[10] R. Fernandez and C. Thompson, "STABILITY OF A SPHERICAL ACCRETION SHOCK WITH NUCLEAR DISSOCIATION," The Astrophysical Journal **697**, 1827 (2009).

[11] R. Fernandez, "Three-dimensional simulations of SASI- and convection-dominated core-collapse supernovae," Royal Astronomical Society **452**, 2071 (2015).

[12] T. Foglizzo, L. Scheck, and H.-T. Janka, "NEUTRINO-DRIVEN CONVECTION VERSUS ADVECTION IN CORE-COLLAPSE SUPERNOVAE," The Astrophysical Journal **652**, 1436 (2006).

[13] H. Bondi, "On spherically symmetrical accretion," Monthly Notices of the Royal Astronomical Society **112**, 195 (1952).

[14] J. Frank, A. King, and D. Raine, *Accretion Power in Astrophysics* (Cambridge University Press, 2002).

[15] S. M. Couch and C. D. Ott, "REVIVAL OF THE STALLED CORE-COLLAPSE SUPERNOVA SHOCK TRIGGERED BY PRECOLLAPSE ASPHERICITY IN THE PROGENITOR STAR," The Astrophysical Journal Letters **778** (2013), 10.1088/2041-8205/778/1/L7.

[16] S. M. Couch, E. Chatzopoulos, W. D. Arnett, and F. Timmes, "THE THREE-DIMENSIONAL EVOLUTION TO CORE COLLAPSE OF A MASSIVE STAR," The Astrophysical Journal Letters **808** (2015), 10.1088/2041-8205/808/1/L21.

[17] B. Müller and H.-T. Janka, "Non-radial instabilities and progenitor asphericities in core-collapse supernovae," Royal Astronomical Society **448**, 2141 (2015).

[18] E. Abdikamalov and T. Foglizzo, "Acoustic wave generation in collapsing massive stars with convective shells," Royal Astronomical Society **439**, 3496 (2020).