HIPACC Summer School 2014: Simulating Stellar Collapse

C. D. Ott

July 22, 2014

1 Introduction

In this exercise, we will be using the code GR1D by O'Connor & Ott (2010) [1] to simulate stellar collapse to a neutron star and black hole formation. GR1D is a spherically symmetric (hence, 1D) code that can run on your laptop. It is fully general relativistic and solves the 1D general-relativistic hydrodynamic equations. In its standard version, which you can be obtained from http://www.stellarcollapse.org, can handle a finite-temperature nuclear equation of state and treats neutrinos.

Here, we will be running GR1D in a simplified customized version for HIPACC, which avoids dependence on external libraries and should be very easy to install on your laptop. You can always go back later and get the full version if you like.

Of course, because GR1D is spherically symmetric, no quadrupole gravitational waves will be emitted by the collapsing stars it simulates. The goal of this exercise is to make you familiar with the basic features of stellar collapse and stellar-mass black hole formation.

2 Installing GR1D on gordon

On gordon, download the GR1D source code tarball:
`wget http://www.tapir.caltech.edu/~cott/HIPACC2014/GR1D_HIPACC2014.tar.gz`

Move the tarball to a location of your choice in your home directory and extract it with

```
tar -xzvf GR1D_HIPACC2014.tar.gz
```

Next cd into GR1D_HIPACC2014 and bring up the file make.inc in your editor and replace the entry for F90 with your Fortran 90 compiler. Chances are that the default, gfortran, will be sufficient. You can test this by just typing gfortran on the command line. The result should be

```
airbert:~ cott$ gfortran
gfortran: fatal error: no input files
compilation terminated.
airbert:~ cott$
```

This means you have `gfortran` in your default path. Otherwise you need to find it on gordon (http://www.sdsc.edu/us/resources/gordon/).

Once you have set up `make.inc`, just type `make` and the `GNU make` system will take care of building `GR1D` for you. The results will be an executable called `GR1D` in the `GR1D_HIPACC2014` directory.

3 Stellar Collapse to a Neutron Star and a Prompt Explosion

We will be simulating stellar collaspe to a neutron star and a prompt supernova explosion in a $15 - M_\odot$ presupernova star. For this, take the file `hybrid_collapse.param` from the `sample_parameter_files` directory, copy it into the `GR1D_HIPACC2014` top level directory, and rename it to `parameters`. This file contains the input parameters that tell `GR1D` what to do. There is no need to change this file for the present exercise.

3.1 Equation of State

In this collapse example, `GR1D` will map in a $15M_\odot$ mass presupernova star and collapse it with a simplified, so-called "hybrid" equation of state [2], which mimics the behavior of the true equation of state in stellar collapse. It is, in essence, a combination of two polytropic equations of state with a simple model to account for thermal pressure due to shock heating:

$$P = P_{\rm cold} + P_{\rm thermal} \ , \tag{1}$$

where

$$P_{\rm cold} = \begin{cases} K_1 \rho^{\Gamma_1} & \rho < \rho_{\rm nuc} \ , \\ K_2 \rho^{\Gamma_2} & \rho \geq \rho_{\rm nuc} \ , \end{cases} \tag{2}$$

where $\rho_{\rm nuc}$ is nuclear saturation density, which we set to $2 \times 10^{14}\,{\rm g\,cm^{-3}}$. Γ_1 is set to a value near $4/3$, which is appropriate to describe the collapse phase at densities below nuclear where the pressure is dominated by degenerate relativistic electrons. At nuclear density, the stiffening of the nuclear EOS due to the short-distance repulsion of the nuclear force is captured by choosing $\Gamma_2 = 2.5$. K_1 is chosen for a relativistic electron gas (see, e.g., [3]) and K_2 is determined by ensuring continuity of the pressure at the transition density.

The thermal component is a simple Γ-law of the form

$$P_{\rm thermal} = (\Gamma_{\rm th} - 1)\rho\epsilon_{\rm th} \ , \tag{3}$$

where $\Gamma_{\rm th}$ is usually set to 1.5 to average contributions of relativistic and non-relativistic gases and the thermal specific internal energy $\epsilon_{\rm th}$ is determined by the difference of the polytropic specific internal energy $\epsilon_{\rm cold}$ and the specific internal energy that comes out of the solution of the hydrodynamics equations. Including $P_{\rm thermal}$ is important to get things right when the supernova shock appears, since shocks lead to heat generation (dissipation).

3.2 Carrying out the Simulation

Once you have set up the `parameters` file, execute `GR1D` by using `./GR1D`. This will start up the code. You will get status output that tells you where the simulation is right now of the following

form:

```
    0   0.000000E+00   1.050000E-07   9.726484E+09   9.942381E-01
  100   2.593969E-04   9.321510E-06   9.668233E+09   9.942367E-01
  200   1.190442E-03   9.305936E-06   9.497712E+09   9.942319E-01
  300   2.122026E-03   9.336463E-06   9.389554E+09   9.942270E-01
  400   3.058001E-03   9.381891E-06   9.339601E+09   9.942218E-01
  [...]
```

The format is: timestep number, physical time in seconds, timestep size in seconds, central density in $\mathrm{g\,cm^{-3}}$, and value of the lapse function α. The lapse is a GR quantity that simply relates coordinate time t to proper time τ. In `GR1D`, $d\tau = \alpha dt$. It is a so-called gauge quantity and `GR1D` uses a Schwarzschild-like gauge in which α will be 1 in flat space and will go down to small values in regions where gravity is strong.

The simulation should reach nuclear density within a few minutes (depending on how fast your laptop is) and the code will then declare that bounce has occured. The simulation will happily continue afterwards, but for our purposes it is not useful to simulate longer than about 0.2 s.

3.3 Looking at Simulation Output

All of `GR1D`'s output files are just plain ASCII. You can look at them using the Unix/Linux tool `less` or your favorite editor. You can also plot up data using your favorite plotting package. The examples below are given for `gnuplot`.

Even while the simulation is running, you can open a second terminal and look at the `Data` subdirectory. There should be many files. For the collapse example, it is most useful to look at `rho_c_t.dat`, which contains the central density as a function of time. You could plot it up in `gnuplot` to track the collapse. Here is an example of how to do this:

```
gnuplot
gnuplot> plot "Data/rho_c_t.dat" u 1:2 w l
```

This will bring up a window showing you $\rho_c(t)$. the `u 1:2 w l` means "plot column 2 as a function of column 1 with lines". As your simulation creates more data, just type `replot` into the `gnuplot` window. This updates the display.

Another useful thing to do is to look at the collapse velocity profile. This is a little more tricky. The velocity as a function of radius is stored in the file `v1.xg` and whenever it is output, the entire grid is output. The format of a single output looks like this:

```
 "Time =   0.12281956402371703
  -7.000000000E+04  -1.188964531E+05
  -5.000000000E+04  -7.000586422E+04
  -3.000000000E+04  -1.043850313E+04
  -1.000000000E+04   5.017415002E+05
   1.000000000E+04  -5.017415002E+05
   3.000000000E+04   1.043850313E+04
   [...]
```

Left is radius in cm, right is the radial velocity in cm/s. Note that the first few entries have negative radius. These are boundary points at the center (the hydrodynamics code needs boundary information and the boundary condition here is reflective). In order to plot the velocity at a time shortly after core bounce in `gnuplot`, do

```
gnuplot> set logscale x
gnuplot> plot "Data/v1.xg" i 200 u 1:2 w l
```

Here the 200 means "output 200" (this is the 200th time the velocity has been output to the file). Unfortunately, you won't be able to get `gnuplot` to display the physical time. If you are ambitious, you could write a `python` script that parses the output file and spits out the right time (and perhaps does the plotting via the much niceer `matplotlib` package). You could even generate a movie showing the evolution of the velocity.

If you look at the velocity at different postbounce times, you will find that the shock does not stall and just explodes the star. This is to be expected, because our simple model here neglects the effects of nuclear dissociation and neutrino cooling.

4 Stellar Collapse to a Neutron Star and subsequent Black Hole Formation

Next we will simulate collapse to a black hole. This will overwrite any data that you have generated in the previous example. If you want to keep the data, then make a copy of the `Data` subdirectory. You could also just rename the subdirectory, but then make sure to create a new `Data` directory, because the code needs it to be there to be able to write output.

We will be collapsing a $75 - M_\odot$ star with low metallicity that fails to explode. We "induce" the failure by reducing $\Gamma_{\rm th}$ (which controls the thermal pressure behind the shock) from its fiducial value of 1.5 to 1.25, which mimics the effect of neutrino cooling and dissociation of heavy nuclei. It will make the shock stall and will thus prevent an explosion.

Take the file `hybrid_collapse_black_hole_formation.param` from the `sample_parameter_files` subdirectory, place it in the toplevel directory and call it `parameters`. Then run the code with `./GR1D`.

This simulation will take quite a bit longer than the previous one, because the resolution is set to be higher (which is necessary for simulating black hole formation). By tracking `rho_c_t.dat`, you will note that first a temporarily stable protoneutron star is formed and that subsequently the central density will further increase until a black hole forms and the simulation eventually crashes. This happens when the lapse α approaches zero in the center of the star. You can check how close your simulation is to black hole formation by tracking `alpha_c_t.dat`. It is also interesting to look at `X.xg`, which contains

$$X(r) = \left(\sqrt{1 - \frac{2GM(r)}{rc^2}} \right)^{-1} . \tag{4}$$

This quantity becomes very large near the location of the Schwarzschild radius $r = 2GM/c^2$ when the black hole forms. It is in a `.xg` file, so what we said about plotting collapse velocity in the previous section applies.

References

[1] E. O'Connor and C. D. Ott. *Class. Quantum Grav.*, **27**, 114103, 2010.

[2] H.-T. Janka, T. Zwerger, and R. Mönchmeyer. *Astron. Astrophys.*, **268**, 360, 1993.

[3] L. S. Shapiro and S. A. Teukolsky. *Black Holes, White Dwarfs and Neutron Stars*. John Wiley & Sons, New York U. S. A., 1983.