Computer lab exercises: Simple stellar models and the TOV equation

I. Hawke, S. Husa, B. Szilágyi

September 2004, January 2013

Contents

1 Introduction

Static, spherically symmetric perfect fluid models in general relativity, and thus models of stars, are described by the Tolman-Oppenheimer-Volkoff equation.

The purpose of this exercise is to get some hands-on experience with the numerical integration of ODE systems and with relativistic stellar models. We will point at some subtleties and potential pitfalls one should be aware of even when dealing with the numerics of relatively simple ODEs, but our main focus here is to illustrate some simple physics concepts.

Our example code comes in two versions, both coded in Fortran 90: a short and simple version which is targeted only to solve the concrete problem at hand, and a production strength modular code that can also be used to solve PDE initial value problems in 1+1 dimensions. In case of doubt, restrict yourself to work with the simple code during the course, and have a look at the more complicated code later, whenever you feel like it.

The text starts with a brief introduction to the TOV equation in Sec. 2, and continues with some remarks about the numerical integration of ODEs in Sec. **??**. In Sec. 3 we make a few remarks about how the example codes are organized, and how to work with them. The exercises are collected in Sec. 4.

2 Background: the TOV equation

The equations for a TOV star [1, 2] are usually given in Schwarzschild coordinates. For a textbook discussion see Chpt. (6.2) in the excellent book by Wald [3]. The notation for the fluid quantities follows [4]. Here we are assuming a perfect fluid matter model, i.e. that the stress energy tensor is given by

$$T^{\mu\nu} = (\mu + P)u^\mu u^\nu + P g^{\mu\nu}, \tag{1}$$

where μ is the total energy, P the pressure and u^μ is the fluid four velocity. Regarding equations of state, we will focus on the simple family of polytropic equations of state (feel free to implement something more realistic):

$$\begin{aligned} \mu &= \rho(1+\epsilon), & (2)\\ P &= (\Gamma - 1)\rho\epsilon, & (3)\\ \rho\,\epsilon &= \frac{p}{(\Gamma-1)}, & (4)\\ P &= K\rho^\Gamma. & (5) \end{aligned}$$

Here we have introduced the rest-mass density ρ and the specific internal energy ϵ.

The metric in Schwarzschild coordinates is given by

$$ds^2 = -e^{2\phi(r)}\,dt^2 + (1 - \frac{2Gm(r)}{rc^2})^{-1}\,dr^2 + r^2\,d\Omega^2. \tag{6}$$

Here $m(r)$ is the gravitational mass inside the sphere of radius r, and ϕ is the logarithm of the lapse and can also be interpreted as the Newtonian gravitational potential.

In the exterior vacuum region we have

$$P = 0, \tag{7}$$
$$m = M, \tag{8}$$
$$\phi = \frac{1}{2}\log(1 - 2GM/rc^2). \tag{9}$$

This allows us to match ϕ at the surface of the star. Note that for numerical hydrodynamical evolutions, the pressure is often set to a small nonzero value ('atmosphere') in the exterior for technical reasons.

The Einstein equations then imply the TOV system of ODEs (for a derivation see e.g. [3] or go ahead and try yourself whenever you have time):

$$\frac{dP}{dr} = -G(\mu + P/c^2)\frac{m + 4\pi r^3 P/c^2}{r(r - 2Gm/c^2)}, \tag{10}$$
$$\frac{dm}{dr} = 4\pi r^2 \mu, \tag{11}$$
$$\frac{d\phi}{dr} = \frac{m + 4\pi r^3 P}{r(r - 2Gm/c^2)}. \tag{12}$$

Note that the last equation decouples from the first two! The initial data at $r = 0$ is (for the numerical integration) $P_0 = K\rho_c^\Gamma$, $m_0 = 0$, $\phi_0 = 0$. Note that the equation is singular at $r = 0$, this may require some attention in the numerical code!

Choose units as appropriate for your code and input/output. In SI units, $c = 299792458.0\, m/s$, $G = 6.673 \times 10^{-11}\, m^3/kg\, s^2$, $\hbar = 1.05457148 \times 10^{-34}\, m^2 kg/s$, $m_\odot = 1.989 \times 10^{30}\, kg$.

3 Code description

3.1 Code organization

Our example codes are prepared in Fortran 90, which is a modern language, but simpler to get started with than C or C++. If you feel fit for the job, feel free to recode the problems in C or C++ (but think like a physicist: try to spend more time doing physics than coding).

We provide two working (that does not mean bugs are impossible) example programs: `SimpleTOV` and `MoLoTOV`. `MoLoTOV` is based on a simple code to solve PDEs in 1+1 dimensions with the *method of lines*. Of course this code-base can also be used to solve ODEs. The code introduces some ideas of modular programming in Fortran 90. It should not be hard to extend this code to solve simple parabolic or hyperbolic PDEs.

`SimpleTOV` is a stripped-down version of `MoLoTOV`, which has been compressed into a single file.

We tried to present a few nice features of Fortran 90, which many users are not aware of, e.g. Namelists, derived types, or `kind` parameters to define the precision of floating point variables. We also make use of Fortran 90 modules.

Floating point variables are defined as

```
real(kind=wp) ::  x,
```

where e.g. the kind parameter `wp` is defined as

`integer, parameter ::  wp = kind(1.0d0),`
which makes `x` a double precision variable.

A namelist is a facility for grouping variables for I/O. The NAMELIST statement is used to define a group of variables as follows:
`NAMELIST / group-name / variable-name-list`
The variables must be declared before appearing in the NAMELIST group. The keyword NML= may be used in place of the format specifier in an I/O statement. For example, the following code is used to read paramters from a file in the `SimpleTOV` code:

```
Namelist /TOV_Input/ dr, r_final, gamma, central_pressure, method, &
     &               every_t_step

Open (unit = iounit, file = "TOV.par", status = "old" )
Read (unit = iounit, nml = TOV_Input)
Close(unit = iounit)
```

Derived types are used heavily in the `MoLoTOV` code, e.g. to define an object that contains all physics-parameters:

```
! The definitions of the different parameter types
type physics_type
   character(len = 42):: equation
   character(len = 42):: eos_type
   character(len = 42):: compute_units_type, output_units_type
   real(kind = wp)    :: l_scale, m_scale, t_scale
   real(kind = wp)    :: central_density
end type physics_type
```

3.2 How to run the code, tips & tricks ...

In order to compile `MoLoTOV`, change you working directory to `TOV/MoLoTOV` and type `make`. Since the `SimpleTOV` code is contained in a single file, a Makefile is not required, compile the code by typing f90 -o tov tov.f90. Both codes come with an example parameter file called `TOV.par`, to run the code type

```
shell> ./tov TOV.par
```

For convergence tests, gnuplot, xmgr and ygraph are convenient. Use what you are most familiar with. The output format of the `SimpleTOV` code is easiest to use with gnuplot, e.g. try

```
gnuplot> plot 'tov.dat' using 2:3
```

to plot the pressure versus the radius. For a convergence test try something like

```
shell> paste tov1.dat tov2.dat > tov12.dat; paste tov2.dat
tov4.dat > tov24.dat
gnuplot> plot 'tov12.dat' using ($2):($3-$9), \
'tov24.dat' using ($2):(4*($3-$9)).
```

4 Exercises

As a "reference model" try the following (this corresponds to the TOV star used for the long term evolutions in [7]): a $N = 1$ ($\Gamma = (N + 1)/N = 2$) polytrope, central rest-mass density $\rho_c = 1.28 \times 10^{-3}$ and $K = 100$. The choice of $K = 100$ is connected to completing the choice of our units to $G = c = M_\odot = 1$ (Think about what issues haveto be considered in connection to the choice of unuits!) The resulting gravitational mass should be of $M = 1.4 M_\odot$, and circumferential radius $R = 14.15$km.

4.1 Exercise 1

Check that the G's and c's are correct. How are we going to convert lengths to kilometers when using $G = c = M_\odot = 1$?

4.2 Exercise 2

Run the code for the Euler and RK2 (optionally also RK4) method. Is the code stable and convergent? Do the results look qualitatively correct?

If the order of convergence is not as predicted, what could be the reason?

4.3 Exercise 3

Extend the code to also compute the metric quantity ϕ.

4.4 Exercise 4

Run the code for the "reference model". Does this look like a NS?

4.5 Exercise 5

Neglect the relativistic terms, and check how the qualitative behavior changes. Which regime of central densities can safely be treated in a Newtonian way?

4.6 Exercise 6

What happens if you modify the polytropic index? Which values make sense?

4.7 Exercise 7

Is the radius of our star always finite? Do neutron stars have an atmosphere (according to our simple polytropic model)?

4.8 Exercise 8

Plot total mass vs. radius for a wide range of central densities. What can you see?

Acknowledgements

We thank Erik Schnetter for helpful comments.

References

[1] R. C. Tolman, Phys. Rev. **55**, 364 (1939).

[2] J. R. Oppenheimer and G. Volkoff, Phys. Rev. **55**, 374 (1939).

[3] R. M. Wald. General Relativity, University of Chicago Press, Chicago, 1984.

[4] J. A. Font, M. Miller, W. Suen and M. Tobias, Phys. Rev. **D61**, 044011 (2000).

[5] R. W. Hamming. Numerical Methods for Scientists and Engineers, Second Edition, Dover Publications, New York, 1973.

[6] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling. Numerical Recipes, Cambridge University Press, Cambridge, England, 1986.

[7] J. A. Font, T. Goodale, S. Iyer, M. Miller, L. Rezzolla, E. Seidel, N. Stergioulas, W. Suen and M. Tobias, Phys. Rev. **D65**, 084024 (2002).