Constructing the Equation of State for Compact Stars

Bachelor thesis

Jan Röder

Contents

1 Introduction **2**

2 Preparations **3**
2.1 The Tolman-Oppenheimer-Volkoff equation 3
2.2 Numerical solution 3
2.2.1 Method 3
2.2.2 Implementation 4

3 Code development **4**

1 Introduction

To this day, the nature of the equation of state [EOS] for neutron stars is highly debated. Previously discussed EOSs have emerged from various topics and models of physics. Each one has different parameters determining its impact on calculations.
For the past years, it has been common to derive an EOS from a theory and then use it to construct a mass-radius relation [MRR]. This method is obviously constrained by the fact that one cannot put an arbitrary number of theories into one EOS. That would result in a problem impossible to solve, even for computers. Therefore, the motivation is to minimize the number of parameters and constraints in the future.
The approach chosen in this thesis is inverting the above process. Instead of using an EOS to calculate masses and radii, those shall now take over the role as input parameters. For compact stars up to a certain mass, the EOS will be assumed to be known well enough. Above that mass a numerical reconstruction will take place, determining the EOS from a variety of MRRs.

2 Preparations

2.1 The Tolman-Oppenheimer-Volkoff equation

Before getting into the actual method, one has to set up a theoretical basis for it. This involves a derivation of the equation used for determining the structure of compact stars. Here, that will be the Tolman-Oppenheimer-Volkoff [TOV] equation.
To make analytical and numerical calculations easier, one can choose the unit system to be $c = G = 1$, so that every unit is a power of length.
The derivation is based off the assumption that the star matter can be described as a perfect/ideal fluid. Further, the system shall not evolve in time, therefore staying spherically symmetric. In terms of the energy-momentum tensor we are left with:

$$T_{\mu\nu} = (\epsilon + P)\, u_\nu u_\mu - P g_{\mu\nu} \tag{1}$$

Spherical symmetry leads to a certain form of the metric, which then gives the stress-energy tensor components:

$$T_{\mu\nu} = diag(\epsilon e^{\nu(r)},\ Pe^{\lambda(r)},\ Pr^2,\ Pr^2 sin^2(\theta)) \tag{2}$$

Imposing hydrostatic equilibrium,

$$\nabla_\nu T^{\mu\nu} = 0 \tag{3}$$

and some calculation, one obtains the expression

$$content... \tag{4}$$

To determine what $\nu(r)$ looks like, and using the previously defined metric, one can calculate the non-zero components of the Ricci tensor:

$$content... \tag{5}$$

The Einstein equations then yield more equations containing $\nu(r)$ and it's derivatives. After plugging some of the equations into one another, one obtains $\nu(r)$ and therefore, in conclusion, the full TOV equation:

$$\frac{dP}{dr} = \frac{(\epsilon + P)(m + 4\pi r^3 P)}{2mr - r^2} \tag{6}$$

Together with a second equation for the mass:

$$\frac{dm}{dr} = 4\pi r^2 \epsilon \tag{7}$$

2.2 Numerical solution

2.2.1 Method

The TOV equations come in the form of a first-order ordinary differential equation system (ODE). In order to obtain sufficient precision, a 4th-order Runge-Kutta algorithm shall be

used to solve said ODE. By looking at both equations, our ODE system is given in the form:

$$\dot{y}(t) = f(y(t), t) \tag{8}$$

where $\dot{y}(t)$ is a two component "vector":

$$\dot{y}(t) = \begin{pmatrix} dP/dr \\ dm/dr \end{pmatrix} \tag{9}$$

$f(y(t), t)$ then contains the right hand side of equations 6 and 7.
Numerically, solving the ODE is achieved by approximation of a step τ in the function variable t (in TOV, this is the radius r) with derivatives of the function that is the ODE solution (y).

$$y(t + \tau) = y(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{10}$$

with increments k_i:

$$\begin{aligned} k_1 &= f(y(t), t) \cdot \tau \\ k_2 &= f(y(t) + k_1/2, t + \tau/2) \cdot \tau \\ k_3 &= f(y(t) + k_2/2, t + \tau/2) \cdot \tau \\ k_4 &= f(y(t) + k_3, t + \tau) \cdot \tau \end{aligned}$$

All k_i are of course also two component "vectors".

2.2.2 Implementation

In the actual program (written in C), the left hand side of our ODE system is implemented as a two-dimensional array of form y[N][x]. N will be two, for the whole program, as we will not add other equations to the system. The k_i are all one-dimensional arrays; that way only x is reflecting the step count. x will therefore be in range zero up to the number of iteration steps.

3 Code development

The goal is to reconstruct the EOS with as few biasing and imposing of mathematical form as possible. Therefore, the only pre-determined input is an EOS corresponding to a given mass in the MRR (a). From there, the EOS can be constructed without further constraint. Here, a straight line is added to the end of the EOS, that is, in the high density region (b). The ODE solver starts at a point near the end of the given EOS and will calculate a mass with the (now small) piece of straight line and the EOS. If the mass does not fit the corresponding one in the MRR that was read in previously, the starting point on the line will be shifted to a higher pressure by a small step. At some very high pressure, a cut-off can be applied due to un-physical energy densities. If the starting point reaches the cut-off and the correct mass was not found, the slope of the line is varied. The process then starts over at a point near the given EOS and shifting stepwise from there.

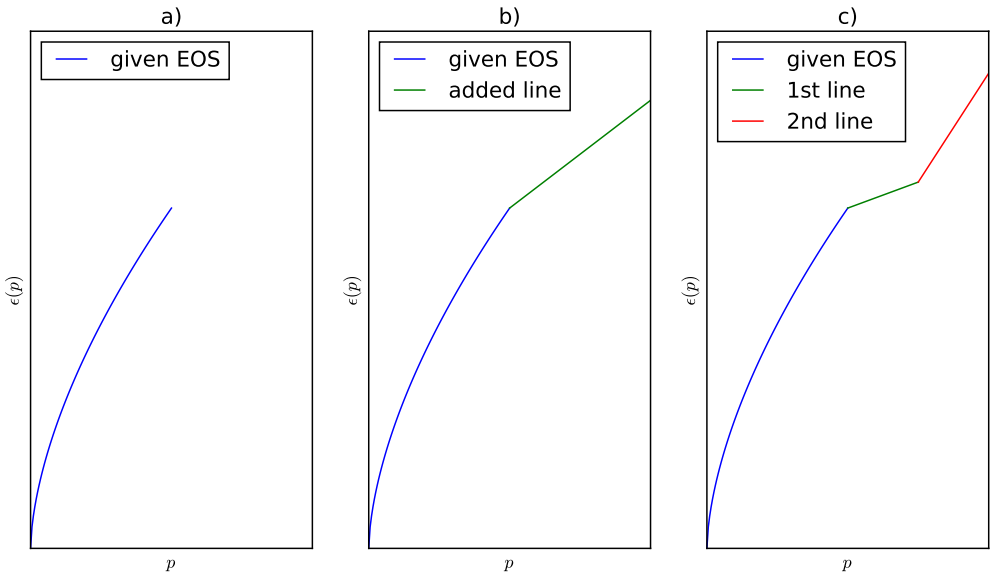

Figure 1: a) Given polytropic equation of state, b) added a constructed straight line

Messung der Fadenlänge	l (m)
1	
2	
3	
4	
5	
Mittelwert $\bar{l}$	
Standardabweichung σ_l	