

# Inverse equation of state construction from mass-radius-relations of compact stars

Bachelor thesis

Jan Röder

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Preparations</b>	<b>2</b>
2.1	The Tolman-Oppenheimer-Volkoff equation . . . . .	2
2.2	Numerical solution . . . . .	3
2.2.1	Method . . . . .	3
2.2.2	Implementation . . . . .	3

# 1 Introduction

To this day, the nature of the equation of state [EOS] for neutron stars is highly debated. Previously discussed EOS have emerged from various topics of physics, such as nuclear matter, quark matter and many others, every one of which could potentially be modified to resemble other physical aspects or different conditions.

For the past years, it has been common to derive these EOS from a theory and then use it to construct first one compact star, then a whole series, determining a relation between total mass and radius. This method obviously is constrained by the fact that one cannot put an arbitrary number of theories and therefore variables into one EOS, as that would result in a problem impossible to solve, even for computers. This is the main reason to write this thesis, as we here will, in a way, revert the previous method.

For compact stars up to a certain mass, we will assume that we know the EOS well enough. Above that mass, we will numerically reconstruct an EOS from a mass-radius relation, and further compare the results with results from EOS dictated and constrained by theories.

## 2 Preparations

### 2.1 The Tolman-Oppenheimer-Volkoff equation

Before we get to the main topics of this thesis, it is important to set up a basis that our work emerges from.

Firstly, this involves a derivation of the equation used for determining the structure of compact stars; here, we will use the Tolman-Oppenheimer-Volkoff [TOV] equation. We later choose our unit system to be  $c = G = 1$ , so that every unit is a power of a length (also, that choice makes numerical evaluation a lot easier).

The derivation is based off assuming the star matter as a perfect/ideal fluid. The system shall further not evolve in time, therefore staying spherically symmetric. In terms of the stress-energy tensor we are left with:

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

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

$$T_{\mu\nu} = \text{diag}(\epsilon e^{\nu(r)}, P e^{\lambda(r)}, P r^2, P r^2 \sin^2(\theta)) \quad (2)$$

We now assume for the star to be in hydrostatic equilibrium. This is expressed by the condition:

$$\nabla_\nu T^{\mu\nu} = 0 \quad (3)$$

which after calculation yields the expression

$$\text{content...} \quad (4)$$

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

$$\text{content...} \quad (5)$$

The Einstein equations give us more equations containing  $\nu(r)$  and its derivatives. After plugging some of the equations into one another, we obtain  $\nu(r)$  and therefore, in conclusion, the full TOV equation:

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

Together with a second equation for the mass:

$$\frac{dm}{dr} = 4\pi r^2 \epsilon \quad (7)$$

## 2.2 Numerical solution

### 2.2.1 Method

In order to generate an initial mass-radius relation to test our "inverse" algorithm with, we use a fourth order Runge-Kutta method to solve the TOV equation along with the mass differential equation numerically. By looking at both equations, our ODE system is in the form

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

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

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

$f(y(t), t)$  then contains the right hand side of equations 6 and 7. As stated before, to create an initial mass radius relation to check the inverted algorithm with, we use a fourth order Runge-Kutta [RK4] code. The RK4 method works as follows:

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

with  $k_i$ :

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

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.

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

## References

- [1] H. J. Eichler, H.-D. Kronfeldt, J. Sahm, *Das Neue Physikalische Grundpraktikum*, Springer-Verlag, Berlin-Heidelberg, 2001.
- [2] H. Kuchling, *Taschenbuch der Physik, 21. Auflage*, Fachbuchverlag Leipzig, 2014.