NTNU
Norwegian University of
Science and Technology

Neutron stars

Investigating candidates for the equation of state in the relativistic mean-field approximation

Lars Sivertsen

MSc in Physics
Submission date: May 2018
Supervisor: Jens Oluf Andersen, IFY

Norwegian University of Science and Technology
Department of Physics

Abstract

Various models for the equation of state (EoS) for a neutron star have been considered using the relativistic mean-field approximation. Firstly, we derive Einstein's field equation and apply it to a spherically symmetric mass-distribution, giving us a mass-radius relation for the stars in each model. Using the path-integral approach, the EoS for an ideal cold neutron gas is derived. Then we introduce the σ–ω model where the strong force is mimicked by the exchange of scalar and vector mesons. Furthermore, scalar self-interactions are included for the σ-field, as well as an isospin force carried by the ρ-meson. Leptons are also added to enforce global charge neutrality. Lastly, the existence of hyperons in neutron stars is discussed. It is found that, though energetically unavoidable, including hyperons decreases the theoretical limit for neutron star masses below the most massive neutron star measured. Possible resolutions to this issue involves repulsive hyperon-hyperon interactions or a phase transition to quark matter.

Samandrag

Det er gjort betrakningar av diverse modellar for tilstandslikninga til ei nøytronstjerne i den relativistiske middelfelt approksimasjonen. Først vert Einstein si feltlikning utleia, deretter vert den anvendt på ei sfærisk-symmetrisk massefordeling. Dette gjev oss eit masse-radius førehald for stjernene i kvar modell. Vidare introduserar me σ–ω-modellen, der den sterke kjernekrafta er sett på som ei utveksling av skalar- og vektor-mesonar. Deretter vert skalare sjølv-interaksjonar inkludert for σ-feltet, samstundes som at ρ-mesonet og leptonar vert lagt til høvesvis for å representere ei isospinkraft og for å sikre global ladningsnøytralitet. Til slutt følger diskusjon av eksistensen av hyperonar i nøytronstjerner. Det er observert at til tross for at hyperonar er energimessig umogleg å unngå i nøytronstjerner, så senker dei den øvre teoretiske massegrensa under den største massa målt eksperimentelt til no. Moglege løysingar på dette problemet involverar fråstøytande hyperon-hyperon interaksjonar eller ein faseovergang til kvarkmaterie.

Acknowledgements

I would like to thank my supervisor, professor Jens O. Andersen, for taking on the responsibility of guiding me through this project. Without the challenging and exciting tasks he has given me, as well as all the hours he has spent reading my work and presenting me with precise comments, this thesis would not have been possible.

Contents

Chapter 1

Introduction

Looking up at the night sky one cannot help but notice the vast number of shining dots, twinkling across the horizon. These lights are only a minuscule fraction of the real number of stellar objects governing our universe. Although they appear to us as tiny grains of sand, almost all of them originate from large balls of burning gas, which we call stars. Most of these stars, about 90 percent, are kept in hydro-static equilibrium by the balance of gravity and the exerted energy from hydrogen fusing to helium. These are the main-sequence stars, the class to which our own sun belongs.

When a main-sequence star burns its hydrogen, a layer of helium starts to build up inside the core. This process is driven by the star's own gravity. If the star has a mass of approximately 10 or more solar masses, the temperature gradient inside the core will be so large that it mixes the elements uniformly inside the core. For this reason, the hydrogen fusion process will continue throughout the core until there is no hydrogen left there.

As the fusion process continues, the helium core becomes more massive. It then shrinks in size due to the increased gravitational pressure. This increases the temperature, eventually causing helium to fuse to carbon in the center.

Once the core has exhausted the hydrogen, the hydrogen burning continues in a shell around the helium-and-carbon core. As the core becomes more massive, the gravitational forces on the core and its surrounding shell increase, leading to a greater rate of fusion. The outer layers then expand, causing the star to become less dense near the surface, as well as giving it an increased radius.

When the star runs out of helium, the core can no longer sustain the gravitational pressure, and so it collapses. It does so until it reaches temperatures where carbon can burn and produce mainly neon, sodium, magnesium and oxygen. The process of burning up nuclear fuel and then collapsing until temperatures reach the threshold for heavier elements, continues inside the core until it becomes pure nickel. Fusion of nickel or heavier elements does not generate any energy, and no star can be sustained by burning such elements – no matter how massive it is.

The fusion in the outer layers is still puffing up the star. As the radius increases, the temperature decreases, making the star glow redder than it used to when it was on the main sequence. Close to the very end of its life, the star has become so large and red that we call it a red supergiant.

Finally, when the star is completely out of nuclear fuel, the core collapses again. However, since the star cannot extract energy from nickel fusion, it keeps collapsing until the degeneracy pressure in the core becomes too large. The core then bounces, creating a shock-wave that moves through the outer layers. Some of this energy is then used to fuse heavier elements, which is the reason we can find elements such as uranium or plutonium here on Earth. The rest of the energy is used to throw most of the mass of the star away. The luminosity from such an event, known as a supernova, can be as bright as all of the stars in a whole galaxy combined.

After the supernova there are two possible end states for the star. If the star had a mass of more than 30 solar masses, the remaining core would have collapsed beyond return and a black hole would be formed. Less massive stars leave a dense core of completely degenerate matter. This is what we know as a neutron star.

The name arose from an idea proposed by Landau in 1931 about a star shaped as a giant nucleus [1]. As the name suggests, a neutron star was originally thought of as a compact object consisting mainly of neutrons. The idea was that the star was so dense that it squeezes the protons and electrons together to form neutron matter. However, later it has become evident that they also contain other particles such as protons and electrons due to the beta decay. There are also models including even more exotic particles know as hyperons [2, 3], as well as models that proposes a phase transitions to quark matter inside the core [2, 4].

Even though the complete neutron star composition is not described yet, we know one thing for sure: They are extreme objects. Neutron stars have on average a mass of about 1.5 solar masses, and a radius of the order of 10 kilometres. This means that the density inside the core is so large that a teaspoon neutron star matter would have a mass comparable to a reasonable sized mountain. To be fair, it does not really make sens to talk about a teaspoon of neutron star. Even if you due to some miracle were able to pull out a hand full, things would not end well. Once no longer inside the star, there is nothing holding the matter together, meaning that it would become a gas in an instant. It would rip you apart. And that is without considering the intense radiation you would experience due to the beta decay of the free neutrons.

Neutron stars are not only dense, they also have strong magnetic fields. A red super giant has a radius about a few hundred to a thousand times the sun's, which means that the surface area decreases by a factor of about $\sim 10^9$ when it collapses to a neutron star. Due to flux conservation the magnetic field increases by the same factor, resulting in some of them having field strengths as high as $10^{10} \sim 10^{15}$ gauss [5].[1] To compare, the field strength of a refrigerator magnet is about 50 gauss. By the same token, due to conservation of angular momentum, some neutron stars can rotate with periods of a few hundreds of a second [6, p. 277]. If a neutron star has a companion, it may be accelerated even further reaching millisecond periods.[2] Comparing this to the 86 000 seconds it takes for the Earth to complete a full rotation, we understand that living on a neutron star might make you dizzy.

Being such extreme objects, neutron stars are ideal when probing models for high-density matter. They have properties we are not even close to reproduce in the lab, making high precision measurements of neutron stars a high priority in physics today. The purpose of this master's thesis is to go through the steps of some of the well-known models for dense matter using the framework of the relativistic mean-field approximation, to see their strengths and find out where they fail.

In 1939 Tolman, Oppenheimer and Volkoff used general relativity to construct an equilibrium equation for a static, spherically symmetric mass distribution. This equation, later known as the Tolman-Oppenheimer-Volkoff (TOV) equation, gives a relation between the change in pressure as a function of the distribution's mass, radius and energy density. The TOV-equation revealed that there must be a maximum mass for such a mass distribution, no matter how the pressure and energy density are related. This means that we can use the limiting mass to falsify our model: We should not predict a maximum mass smaller than the most massive neutron star measured, which today is approximately two solar masses [7]. Even better, it turns out that the mass-radius relation is uniquely determined by the EoS; in addition, Lindblom showed in 1992 that given a set of widely spread measurements, the mass-radius relation can be obtained to good accuracy [8]. This means that in the future, with better measurements and given that neutron stars span a large enough range of masses, we can find the EoS numerically and use it as a test for the models we propose. This highly motivates the pursuit of a neutron star mass-radius relation when searching for a realistic model for dense matter.

In some way, it is beautiful how the neutron stars we observe today are the result of the same events that created the building blocks of our solar system. Looking up at the night sky we see the true creators of our world. As Carl Sagan famously said: "We are made of star-stuff".

[1] The neutron stars with the most intense magnetic fields are called magnetars.

[2] The fastest rotation neutron stars are called pulsars.

Chapter 2

Einstein's field equation

When Albert Einstein completed his work on general relativity in 1915, he had spent several years to develop the field equations using the principple of equivalence. Later the same year, in 1915, David Hilbert showed that Einstein's field equation also could be derived using Hamilton's principle on a suitable Lagrangian. In this and the following section we will follow in Hilbert's footsteps and find Einstein's equation from the Einstein-Hilbert (EH) action using two different methods. Firstly, we assume that the Christoffel symbols are on the form given by (C.1) and find Einstein's equation by extremizing the EH-action while varying it with respect to the metric tensor. Secondly, we relax one more constraint, treating the Christoffel symbols and the metric tensor as independent variables. This will result in an additional equation giving the form of the Christoffel symbols. The latter method is often referred to as the Palatini approach after Attilio Palatini.[1]

2.1 Variation with respect of the metric tensor

Einstein's field equation can be derived from the Einstein-Hilbert action [10]

$$S_{\text{EH}} = \int \mathrm{d}^4x \left[\frac{1}{16\pi G} g^{\mu\nu} R_{\mu\nu} + \mathscr{L}_{\text{M}} \right] \sqrt{-\det(g_{\mu\nu})}, \tag{2.1}$$

where $g^{\mu\nu}$ is the metric tensor, G is Newton's gravitational constant, $R_{\mu\nu}$ is the Ricci tensor defined by C.3 and $\mathscr{L}_{\text{M}}$ is the matter Lagrangian. By convention, one often writes $\kappa \equiv 8\pi G$, and defines the Ricci scalar as $R \equiv g^{\mu\nu} R_{\mu\nu}$. In this chapter I will also denote the determinant of a tensor $A_{\mu\nu}$ as simply $|A|$. The action then simplifies to

$$S_{\text{EH}} = \int \mathrm{d}^4x \left[\frac{1}{2\kappa} R + \mathscr{L}_M \right] \sqrt{-|g|}. \tag{2.2}$$

By Hamilton's principle, we find the equations of motion by setting the variation of the action δS_{EH} to zero. Hence,

$$\begin{aligned} \delta S_{\text{EH}} = 0 &= \int \mathrm{d}^4x \left[\frac{\sqrt{-|g|}}{2\kappa} \frac{\delta R}{\delta g^{\mu\nu}} \delta g^{\mu\nu} + \frac{R}{2\kappa} \frac{\delta\big(\sqrt{-|g|}\big)}{\delta g^{\mu\nu}} \delta g^{\mu\nu} + \frac{\delta\big(\mathscr{L}_M \sqrt{-|g|}\big)}{\delta g^{\mu\nu}} \delta g^{\mu\nu} \right] \\ &= \int \mathrm{d}^4x \left[\frac{1}{2\kappa} \frac{\delta R}{\delta g^{\mu\nu}} + \frac{R}{2\kappa} \frac{\delta\big(\sqrt{-|g|}\big)}{\sqrt{-|g|}\delta g^{\mu\nu}} + \frac{\delta\big(\mathscr{L}_M \sqrt{-|g|}\big)}{\sqrt{-|g|}\delta g^{\mu\nu}} \right] \sqrt{-|g|}\delta g^{\mu\nu}. \end{aligned} \tag{2.3}$$

[1] Palatini variation, often known as Palatini formalism, was actually invented by Einstein. The name occurred as some physicists started to mix up the approach with the related Palatini identity [9].

The choice of $\delta g^{\mu\nu}$ is arbitrary, so the variation does not depend on it. Then the expression inside the brackets becomes zero and we find that

$$\frac{1}{2\kappa}\left[\frac{\delta R}{\delta g^{\mu\nu}}+\frac{R}{\sqrt{-|g|}}\frac{\delta\big(\sqrt{-|g|}\big)}{\delta g^{\mu\nu}}\right]=-\frac{1}{\sqrt{-|g|}}\frac{\delta\big(\mathscr{L}_M\sqrt{-|g|}\big)}{\delta g^{\mu\nu}}. \tag{2.4}$$

We now examine these terms assuming that the Christopher symbols are on the form given by (C.1):

$$\Gamma^{\lambda}_{\mu\nu}=\frac{1}{2}g^{\lambda\alpha}\left(g_{\alpha\mu,\nu}+g_{\nu\alpha,\mu}-g_{\mu\nu,\alpha}\right). \tag{2.5}$$

Here we have used the comma notation to represent derivatives:

$$g_{\mu\nu,\rho}\equiv\partial_\rho g_{\mu\nu}. \tag{2.6}$$

Without loss of generality, we may choose our coordinate system so that there exists some point P for which spacetime is locally flat. This means that we can set the Christoffel symbols $\Gamma^{\lambda}_{\mu\nu}$ equal to zero at P. For this to be true for $\alpha=\lambda$, we must have that the expression inside the parenthesis is zero, and thus

$$g_{\mu\nu,\alpha}=0, \tag{2.7}$$

for all μ,ν and α at P. Moreover, the last two terms in the Ricci tensor (C.3)

$$R_{\mu\nu}\equiv\Gamma^{\rho}_{\nu\mu,\rho}-\Gamma^{\rho}_{\rho\mu,\nu}+\Gamma^{\rho}_{\rho\lambda}\Gamma^{\lambda}_{\mu\nu}-\Gamma^{\rho}_{\nu\lambda}\Gamma^{\lambda}_{\rho\mu}, \tag{2.8}$$

vanish so that we obtain

$$\delta R=g^{\mu\nu}\delta R_{\mu\nu}+R_{\mu\nu}\delta g^{\mu\nu}. \tag{2.9}$$

We assume that the metric tensor $g^{\mu\nu}$ is symmetric. Then, by definition, the Chistoffel symbols are symmetric in the lower indices due to (2.5). If we then define a variational four-vector

$$\delta r^{\kappa}=g^{\mu\nu}\delta\Gamma^{\kappa}_{\mu\nu}-g^{\mu\kappa}\delta\Gamma^{\lambda}_{\mu\lambda}, \tag{2.10}$$

we can use (2.7) and (2.8) to obtain

$$\begin{aligned}g^{\mu\nu}\delta R_{\mu\nu}=&g^{\mu\nu}\left(\delta\Gamma^{\rho}_{\mu\nu,\rho}-\delta\Gamma^{\rho}_{\mu\rho,\nu}\right)\\=&g^{\mu\nu}\left(\partial_\rho\delta\Gamma^{\rho}_{\mu\nu}-\partial_\nu\delta\Gamma^{\rho}_{\mu\rho}\right)\\=&\partial_\rho\left(g^{\mu\nu}\delta\Gamma^{\rho}_{\mu\nu}\right)-\partial_\nu\left(g^{\mu\nu}\delta\Gamma^{\rho}_{\mu\rho}\right)\\=&\partial_\rho\left(g^{\mu\nu}\delta\Gamma^{\rho}_{\mu\nu}\right)-\partial_\rho\left(g^{\mu\rho}\delta\Gamma^{\nu}_{\mu\nu}\right)\\=&\partial_\rho\left(g^{\mu\nu}\delta\Gamma^{\rho}_{\mu\nu}-g^{\mu\rho}\delta\Gamma^{\nu}_{\mu\nu}\right)\\=&\partial_\rho\delta r^{\rho}.\end{aligned} \tag{2.11}$$

Furthermore, (2.7) gives

$$\partial_\rho\sqrt{-|g|}=0, \tag{2.12}$$

and thus

$$\partial_\rho\delta r^{\rho}=\partial_\rho\left(\frac{\sqrt{-|g|}}{\sqrt{-|g|}}\delta r^{\rho}\right)=\frac{1}{\sqrt{-|g|}}\partial_\rho\big(\sqrt{-|g|}\delta r^{\rho}\big). \tag{2.13}$$

This derivative is a four divergence where the factor $(-|g|)^{-\frac{1}{2}}$ is needed to make it Lorenz invariant. Such a term has no physical meaning since we can always add a four divergence to the Lagrangian without changing the equations of motion. We can therefore neglect it and set it to zero. Thus we have determined that the first term in equation (2.4) yields

$$\frac{\delta R}{\delta g^{\mu\nu}}=R_{\mu\nu}. \tag{2.14}$$

To obtain the second term on the left-hand side of (2.4) we start by using the chain rule to find

$$\delta\sqrt{-|g|} = -\frac{1}{2\sqrt{-|g|}}\delta|g|. \tag{2.15}$$

For a matrix $A\left(t\right)$, the differential of the determinant is given by [11, p.169-171]

$$\delta|A\left(t\right)| = \mathrm{tr}\Big\{\mathrm{adj}\big[A(t)\big]\delta A(t)\Big\}, \tag{2.16}$$

which gives

$$\delta|g| = \mathrm{tr}\left[\mathrm{adj}\left(g_{\mu\nu}\right)\delta g_{\nu\rho}\right] = \mathrm{tr}\left[|g|g^{\mu\nu}\delta g_{\nu\rho}\right] = |g|g^{\mu\nu}\delta g_{\mu\nu}. \tag{2.17}$$

Using the identity [12]

$$|g|g^{\mu\nu}\delta g_{\mu\nu} = -|g|g_{\mu\nu}\delta g^{\mu\nu}, \tag{2.18}$$

combined with (2.15) results in

$$\frac{R}{\sqrt{-|g|}}\frac{\delta\sqrt{-|g|}}{\delta g^{\mu\nu}} = -\frac{1}{2}g_{\mu\nu}R. \tag{2.19}$$

Finally, the numerator on the right side of (2.4) is

$$\begin{aligned}\delta\big(\mathscr{L}_M\sqrt{-|g|}\big) &= \sqrt{-|g|}\delta\mathscr{L}_M + \mathscr{L}_M\delta\sqrt{-|g|}\\ &= \sqrt{-|g|}\delta\mathscr{L}_M + \mathscr{L}_M\Big(-\frac{1}{2}\sqrt{-|g|}g_{\mu\nu}\delta g^{\mu\nu}\Big),\end{aligned} \tag{2.20}$$

which gives

$$\frac{1}{\sqrt{-|g|}}\frac{\delta\big(\mathscr{L}_M\sqrt{-|g|}\big)}{\delta g^{\mu\nu}} = \frac{\delta\mathscr{L}_M}{\delta g^{\mu\nu}} - \frac{1}{2}g_{\mu\nu}\mathscr{L}_M. \tag{2.21}$$

Einstein's equation (2.4) then becomes

$$\frac{1}{2\kappa}R_{\mu\nu} - \frac{1}{4\kappa}g_{\mu\nu}R = -\frac{\mathscr{L}_M}{\delta g^{\mu\nu}} + \frac{1}{2}\mathscr{L}_M g_{\mu\nu} \tag{2.22}$$

To clean this up a bit, we multiply both sides with 2κ,

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \kappa\left(-2\frac{\delta\mathscr{L}_M}{\delta g^{\mu\nu}} + \mathscr{L}_M g_{\mu\nu}\right), \tag{2.23}$$

and recognize that from (C.5) the expression inside the brackets is the definition of the stress-energy tensor $T_{\mu\nu}$. The resulting equation of motion is then

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \kappa T_{\mu\nu}. \tag{2.24}$$

It is worth noting that the left side of this equation is a function of the system's energy and momentum, while the right side is determined by the curvature of space. A physical interpretation of this is that the energy of the system instructs spacetime how to curve, while the shape of spacetime determines how the energy flows.

2.2 Variation using the Palatini approach

In this section, we would like to find Einstein's equation while relaxing another constraint. As mentioned, we will now assume that the Christoffel symbols are independent on the metric, and show that this gives the same equations as before. However, this approach will result in two equations instead of one: Einstein's equation and one giving the form of the Christoffel symbols.

Even though the form of the Christoffel symbols is unknown, we will assume that they are symmetric in their lower indices, which is one of the basic assumptions of general relativity. If one had not assumed this, the remaining extra degree of freedom could have been resolved in a few different ways. One possibility is to add a coupling between spin and the gravitational field. This is known as the Einstein-Cartan theory of gravity, and proposes that fermionic and bosonic matter interferes differently with the gravitational field. This theory is so far neither supported, or falsified, but some suggests that it is necessary to account for dark matter [13]. Another possible solution is "Teleparallelism", where one lets the curvature vanish while the torsion is nonzero.[2] This provides a whole new set of equations, which are dynamically equivalent to general relativity [14]. However, it turns out that there still are some slight differences [15]. There are also plenty of other methods, but we will stick to the good old general relativity.

In the Palatini formalism, the EH-action is still the same, except that we now let the Ricci scalar R be dependent on the unknown Christoffel symbols Γ:

$$S_{\mathrm{EH}} = \int \mathrm{d}^4x \left[\frac{1}{2\kappa} R\left(\Gamma\right) + \mathscr{L}_{\mathrm{M}}\right] \sqrt{-|g|}. \tag{2.25}$$

Since R is only explicitly dependent on Γ, and the Christoffel symbols are assumed independent on the metric, we have that

$$\frac{\delta R_{\mu\nu}}{\delta g^{\mu\nu}} = 0. \tag{2.26}$$

Thus, varying the EH-action with respect to the metric tensor still gives Einstein's equation (2.24).

To find the form of the Christoffel symbols, we vary the action with respect to Γ and find that

$$\delta S_{\mathrm{EH}} = \int \mathrm{d}^4x \frac{1}{2\kappa} g^{\mu\nu} \sqrt{-|g|} \delta R_{\mu\nu}, \tag{2.27}$$

since $\mathscr{L}_{\mathrm{M}}$ is independent on Γ. From (2.8) we have that

$$\delta R_{\mu\nu} = \delta\Gamma^{\rho}_{\mu\nu,\rho} - \delta\Gamma^{\rho}_{\mu\rho,\nu} + \Gamma^{\rho}_{\sigma\rho}\delta\Gamma^{\sigma}_{\mu\nu} + \Gamma^{\sigma}_{\mu\nu}\delta\Gamma^{\rho}_{\sigma\rho} - \Gamma^{\rho}_{\sigma\nu}\delta\Gamma^{\sigma}_{\mu\rho} - \Gamma^{\sigma}_{\mu\rho}\delta\Gamma^{\rho}_{\sigma\nu}. \tag{2.28}$$

Note that

$$\begin{aligned}&\int \mathrm{d}^4x \left[g^{\mu\nu}\sqrt{-|g|}\partial_\rho \delta\Gamma^{\rho}_{\mu\nu} - g^{\mu\nu}\sqrt{-|g|}\partial_\nu\delta\Gamma^{\rho}_{\mu\rho}\right] \\ &= \int \mathrm{d}^4x \left[-\partial_\rho\left(\sqrt{-|g|}g^{\mu\nu}\right)\delta\Gamma^{\rho}_{\mu\nu} + \partial_\nu\left(\sqrt{-|g|}g^{\mu\nu}\right)\delta\Gamma^{\rho}_{\mu\rho}\right] + (\text{boundary terms}),\end{aligned} \tag{2.29}$$

where the last step is achieved through a partial integration. By Hamilton's principle, the action integral has by definition no variation at the boundaries, and hence the boundary terms are zero. The variation of the EH-action can then be written as

$$\begin{aligned}\delta S_{EH} = \int \mathrm{d}^4x \Big[&- \frac{1}{\sqrt{-|g|}}\partial_\rho\left(\sqrt{-|g|}g^{\mu\nu}\right)\delta\Gamma^{\rho}_{\mu\nu} + \frac{1}{\sqrt{-|g|}}\partial_\nu\left(\sqrt{-|g|}g^{\mu\nu}\right)\delta\Gamma^{\rho}_{\mu\rho} \\ &+ g^{\mu\nu}\Big(\Gamma^{\sigma}_{\mu\nu}\delta\Gamma^{\rho}_{\sigma\rho} + \Gamma^{\rho}_{\sigma\rho}\delta\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\sigma}_{\mu\rho}\delta\Gamma^{\rho}_{\sigma\nu} - \Gamma^{\rho}_{\sigma\nu}\delta\Gamma^{\sigma}_{\mu\rho}\Big)\Big]\sqrt{-|g|}.\end{aligned} \tag{2.30}$$

The indices are just dummy variables, so we can interchange them:

$$\begin{aligned}\delta S_{\mathrm{EH}} = \int \mathrm{d}^4x \Big\{ &- \frac{1}{\sqrt{-|g|}}\partial_\rho\Big(\sqrt{-|g|}g^{\mu\nu}\Big) + \left[\frac{1}{\sqrt{-|g|}}\partial_\gamma\Big(\sqrt{-|g|}g^{\mu\gamma}\Big) + g^{\alpha\beta}\Gamma^{\mu}_{\alpha\beta}\right]\delta^{\nu}_{\rho} \\ &+ g^{\mu\nu}\Gamma^{\sigma}_{\rho\sigma} - g^{\sigma\nu}\Gamma^{\mu}_{\sigma\rho} - g^{\mu\sigma}\Gamma^{\nu}_{\rho\sigma}\Big\}\sqrt{-|g|}\delta\Gamma^{\rho}_{\mu\nu}.\end{aligned} \tag{2.31}$$

[2] Torsion does here refer to the anti-symmetric part of the stress-energy tensor.

As before, the variation $\delta\Gamma$ is arbitrary, so the expression inside the brackets must be zero. Then we write

$$A_{\rho}^{\mu\nu} + B_{\rho}^{\mu\nu} = 0, \tag{2.32}$$

where we have defined

$$\begin{aligned} A_{\rho}^{\mu\nu} =& g^{\mu\nu}\Gamma^{\sigma}_{\rho\sigma} - g^{\sigma\nu}\Gamma^{\mu}_{\sigma\rho} - g^{\mu\sigma}\Gamma^{\nu}_{\rho\sigma} - \frac{1}{\sqrt{-|g|}} g^{\mu\nu}\partial_{\rho}\sqrt{-|g|} - g^{\mu\nu}_{,\rho}, & (2.33)\\ B_{\rho}^{\mu\nu} =& \left(\frac{1}{\sqrt{-|g|}} g^{\mu\gamma}\partial_{\gamma}\sqrt{-|g|} + g^{\mu\gamma}_{,\gamma} + g^{\alpha\beta}\Gamma^{\mu}_{\alpha\beta}\right)\delta^{\nu}_{\rho}. & (2.34)\end{aligned}$$

Using our assumption that the Christoffel symbols are symmetric in the lower indices, we find that $A_{\rho}^{\mu\nu}$ must be symmetric in the upper indices. Then $B_{\rho}^{\mu\nu}$ must also be symmetric in the upper indices since we can move it to the right-hand side of equation (2.32). The only possible non-zero elements of $B_{\rho}^{\mu\nu}$ are the ones where $\rho = \nu$. If we then define the tensor

$$C^{\mu} = \frac{1}{\sqrt{-|g|}} g^{\mu\gamma}\partial_{\gamma}\sqrt{-|g|} + g^{\mu\gamma}_{,\gamma} + g^{\alpha\beta}\Gamma^{\mu}_{\alpha\beta}, \tag{2.35}$$

we see that when $\mu \neq \nu$, we have

$$B_{\nu}^{\mu\nu} = C^{\mu} = B_{\nu}^{\nu\mu} = 0. \qquad \text{(No summation implied)} \tag{2.36}$$

Thus $C^{\mu} = 0$ for all μ, and hence $B_{\rho}^{\mu\nu}$ must vanish for all μ, ν and ρ.

We proceed by contracting $A_{\rho}^{\mu\nu}$ with μ and ν:

$$\begin{aligned} 0 &= -\frac{1}{\sqrt{-|g|}} g_{\mu\nu}g^{\mu\nu}\partial_{\rho}\sqrt{-|g|} - g_{\mu\nu}g^{\mu\nu}_{,\rho} + g_{\mu\nu}g^{\mu\nu}\Gamma^{\sigma}_{\rho\sigma} - g_{\mu\nu}g^{\sigma\nu}\Gamma^{\mu}_{\sigma\rho} - g_{\mu\nu}g^{\mu\sigma}\Gamma^{\nu}_{\rho\sigma}\\ &= -\frac{4}{\sqrt{-|g|}}\partial_{\rho}\sqrt{-|g|} - g_{\mu\nu}g^{\mu\nu}_{,\rho} + 4\Gamma^{\sigma}_{\rho\sigma} - \delta^{\sigma}_{\mu}\Gamma^{\mu}_{\sigma\rho} - \delta^{\sigma}_{\nu}\Gamma^{\nu}_{\rho\sigma}\\ &= -\frac{4}{\sqrt{-|g|}}\partial_{\rho}\sqrt{-|g|} - g_{\mu\nu}g^{\mu\nu}_{,\rho} + 2\Gamma^{\sigma}_{\rho\sigma}. \end{aligned} \tag{2.37}$$

Interchanging variation with differentiation in (2.15), we see that

$$g_{\mu\nu}g^{\mu\nu}_{,\rho} = -\frac{2}{\sqrt{-|g|}}\partial_{\rho}\sqrt{-|g|}, \tag{2.38}$$

and obtain

$$\Gamma^{\sigma}_{\rho\sigma} = \frac{1}{\sqrt{-|g|}}\partial_{\rho}\sqrt{-|g|}. \tag{2.39}$$

Substituting this result back in (2.32), and remembering that $B_{\rho}^{\mu\nu}$ is zero, we finally arrive at

$$\begin{aligned} 0 &= -g^{\mu\nu}\Gamma^{\sigma}_{\rho\sigma} - g^{\mu\nu}_{,\rho} + g^{\mu\nu}\Gamma^{\sigma}_{\rho\sigma} - g^{\sigma\nu}\Gamma^{\mu}_{\sigma\rho} - g^{\mu\sigma}\Gamma^{\nu}_{\rho\sigma}\\ &= -\left(g^{\mu\nu}_{,\rho} + g^{\sigma\nu}\Gamma^{\mu}_{\sigma\rho} + g^{\mu\sigma}\Gamma^{\nu}_{\rho\sigma}\right). \end{aligned} \tag{2.40}$$

Under the condition that the Christoffel symbols are symmetric in the lower indices, this equation has the unique solution [16, p.61-62]

$$\Gamma^{\sigma}_{\mu\nu} = \frac{1}{2}g^{\sigma\alpha}\left(g_{\alpha\mu,\nu} + g_{\nu\alpha,\mu} - g_{\mu\nu,\alpha}\right), \tag{2.41}$$

which we recognize as the form we assumed in the previous section.

2.3 Summary

In this chapter we have derived the equations of motion for general relativity using two approaches: One where we assumed the form of the Christoffel symbols, and one where we only assumed that they are symmetric in their lower indices. Both approaches yielded the same result, but the latter, also called the Palatini approach, relies on fewer assumptions.

Chapter 3

The Tolman-Oppenheimer-Volkov equation

After deriving Einstein's equation, we now want to apply it. As a starting point, we look for an equation describing the pressure inside a star assuming that the star may be modelled by a static, spherically symmetric, perfect mass distribution. By perfect, we here mean that there is no sheer stress or heat transfer within the volume of interest.

3.1 The general spherically symmetric metric

The most general static, spherically symmetric metric in spherical coordinates is diagonal and given by

$$\begin{aligned}\mathrm{d}s^2 &= g_{00}\mathrm{d}t^2 + g_{11}\mathrm{d}r^2 + g_{22}\mathrm{d}\theta^2 + g_{33}\mathrm{d}\phi^2 \\ &= A(r)\mathrm{d}t^2 - B(r)\mathrm{d}r^2 - r^2C(r)\left[\mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi^2\right],\end{aligned} \tag{3.1}$$

where A, B and C are functions of r only. Substituting $R = r\sqrt{C(r)}$, we see that

$$\mathrm{d}R = \frac{1}{2\sqrt{C(r)}}r\mathrm{d}r + \sqrt{C(r)}\mathrm{d}r = \mathrm{d}r\left(\frac{R}{2C(r)} + \sqrt{C(r)}\right), \tag{3.2}$$

which inserted into the line element (3.1) gives

$$\begin{aligned}\mathrm{d}s^2 =& A(r)\mathrm{d}t^2 - B(r)\left(\frac{R}{2C(r)} + \sqrt{C(r)}\right)^{-2}\mathrm{d}R^2 - C(r)\frac{R^2}{C(r)}\left(\mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi^2\right) \\ =& \alpha(R)\mathrm{d}t^2 - \beta(R)\mathrm{d}R^2 - R^2\left(\mathrm{d}\theta^2 + \sin^2\theta\mathrm{d}\phi\right),\end{aligned} \tag{3.3}$$

where we have defined the functions

$$\alpha(R) \equiv A\left(\frac{R}{\sqrt{C(r)}}\right), \tag{3.4}$$

$$\beta(R) \equiv B(r)\left(\frac{R}{2C(r)} + \sqrt{C(r)}\right)^{-2}. \tag{3.5}$$

In the following, we rename the variable R to r.

As a boundary condition we impose that the metric must be flat far away from the source. This means that

$$\lim_{r\to\infty}\alpha(r) = \lim_{r\to\infty}\beta(r) = 1. \tag{3.6}$$

Also, α and β cannot change signs since this violates the signature $(+,-,-,-)$ of the metric. Then we can without loss of generality write the functions α and β as exponential functions:

$$\alpha(r) = \mathrm{e}^{2a(r)}, \tag{3.7}$$

$$\beta(r) = \mathrm{e}^{2b(r)}, \tag{3.8}$$

where $a(r)$ and $b(r)$ are functions of r only. It is also worth to note that since $g^{\mu\nu}$ is the inverse of $g_{\mu\nu}$, and since the metric is diagonal, we have that

$$g^{\mu\nu} = \frac{1}{g_{\mu\nu}}. \tag{3.9}$$

3.2 The stress-energy tensor and the Ricci scalar

The star is assumed static, so at any point the four-velocity is given by

$$u^\mu \equiv \frac{\mathrm{d}x^\mu}{\mathrm{d}\tau} = \left(\frac{\mathrm{d}x^0}{\mathrm{d}\tau}, \frac{\mathrm{d}x^1}{\mathrm{d}\tau}, \frac{\mathrm{d}x^2}{\mathrm{d}\tau}, \frac{\mathrm{d}x^3}{\mathrm{d}\tau}\right) = \left(\frac{1}{\sqrt{g_{00}}}, 0, 0, 0\right) = (\mathrm{e}^{-a}, 0, 0, 0), \tag{3.10}$$

since the three-velocity is zero. For a perfect mass-distribution the stress-energy tensor is [17, p. 70]

$$T^{\mu\nu} = -P(r)g^{\mu\nu} + [P(r) + \epsilon(r)]u^\mu u^\nu, \tag{3.11}$$

where $\epsilon(r)$ is the energy density and $P(r)$ is the pressure. By assumption there is no sheer stress or heat-conduction and hence the stress-energy tensor is diagonal. Using this, and the fact that

$$T^\mu_\nu = g_{\rho\nu}T^{\mu\rho} = -P(r)\delta^\mu_\nu + [P(r) + \epsilon(r)]g_{\nu\rho}u^\mu u^\rho, \tag{3.12}$$

one obtains

$$T^0_0 = -P(r) + [P(r) + \epsilon(r)]\mathrm{e}^{2a}\mathrm{e}^{-a}\mathrm{e}^{-a} = \epsilon(r), \tag{3.13}$$

$$T^i_i = -P(r). \tag{3.14}$$

Before we go back to Einstein's equation, we need the expressions for the Christoffel symbols. There are 64 Christoffel symbols in total. Since the symbols are symmetric in the lower indices, only 40 of them are independent. Because we assumed that the star is static, taking derivatives of the metric with respect to time gives zero. Looking at the expression for the Christoffel symbols (2.5), we also notice that only the terms that contain diagonal elements of the metric tensor are non-zero. In fact, after a closer inspection, there are only 9 nonzero independent Christoffel symbols:

$$\begin{aligned}
&\Gamma^0_{01} = a', \\
&\Gamma^1_{00} = a'\mathrm{e}^{2(a-b)}, \quad \Gamma^1_{11} = b', \quad \Gamma^1_{22} = -r\mathrm{e}^{-2b}, \quad \Gamma^1_{33} = -r\mathrm{e}^{-b}\sin^2\theta, \\
&\Gamma^2_{12} = \frac{1}{r}, \qquad\quad \Gamma^2_{33} = -\sin\theta\cos\theta, \\
&\Gamma^3_{13} = \frac{1}{r}, \qquad\quad \Gamma^3_{23} = \frac{\cos\theta}{\sin\theta}.
\end{aligned} \tag{3.15}$$

Here we have denoted derivatives with respect to r by $'$ and skipped writing the r-dependence such that $\partial_r y(r) \equiv y'$. Going back to Einstein's equation

$$T_{\mu\nu} = \frac{1}{\kappa}\left(R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R\right), \tag{3.16}$$

we see that since $T_{\mu\nu}$ is diagonal, so is $R_{\mu\nu}$. From (2.8) the diagonal elements of the Ricci tensor is straightforward to compute:

$$\begin{aligned}
R_{00} &= \mathrm{e}^{2(a-b)} \left[a'' + (a')^2 - a'b' + \frac{2}{r} a' \right], && (3.17)\\
R_{11} &= a'b' - a'' - (a')^2 + \frac{2}{r} b', && (3.18)\\
R_{22} &= \mathrm{e}^{-2b} \left[r\left(b' - a'\right) - 1 \right] + 1, && (3.19)\\
R_{33} &= \left\{ \mathrm{e}^{-2b} \left[r\left(b' - a'\right) - 1 \right] + 1 \right\} \sin^2\theta = \sin^2\theta R_{22}. && (3.20)
\end{aligned}$$

Then the Ricci scalar becomes

$$\begin{aligned}
R =& g^{\mu\nu} R_{\mu\nu} = g^{00} R_{00} + g^{11} R_{11} + g^{22} R_{22} + g^{33} R_{33} \\
=& - \frac{R_{00}}{\mathrm{e}^{2a(r)}} + \frac{R_{11}}{\mathrm{e}^{2b(r)}} + \frac{2R_{22}}{r^2} \\
=& - 2\mathrm{e}^{-2b} \left[a'' + (a')^2 - a'b' + \frac{2}{r}(a' - b') + \frac{1}{r^2}\left(1 - \mathrm{e}^{2b}\right) \right]. \qquad (3.21)
\end{aligned}$$

Now we start evaluating the energy-stress tensor. Firstly, we look at Einstein's equation (3.16) for T_{00}:

$$T_{00} = g_{00} T^0_0 = \mathrm{e}^{2a} \epsilon(r) = \mathrm{e}^{2(a-b)} \frac{1}{\kappa} \left[\frac{2b'}{r} - \frac{1}{r^2}\left(1 - \mathrm{e}^{2b}\right) \right]. \qquad (3.22)$$

This can be rewritten as

$$\epsilon(r) \kappa r^2 = \mathrm{e}^{-2b}\left(2b'r - 1\right) + 1 = 2b'r\mathrm{e}^{-2b} - \mathrm{e}^{-2b} + 1. \qquad (3.23)$$

Note that

$$\frac{\mathrm{d}}{\mathrm{d}r}\left(r\left[\mathrm{e}^{-2b} - 1\right]\right) = -2b'r\mathrm{e}^{-2b} + \mathrm{e}^{-2b} - 1, \qquad (3.24)$$

which inserted into (3.23) gives

$$-\frac{\mathrm{d}}{\mathrm{d}r}\left(r\left[\mathrm{e}^{-2b} - 1\right]\right) = \epsilon(r)\kappa r^2. \qquad (3.25)$$

Finally, integrating both sides from zero to r gives the unknown function e^{2y}:

$$\begin{aligned}
-r\left(-1 + \mathrm{e}^{-2b}\right) &= 2G \int_0^r 4\pi\epsilon(r) r^2 \mathrm{d}r \\
r\left(-1 + \mathrm{e}^{-2b}\right) &= -2GM(r) \\
\mathrm{e}^{-2b} &= 1 - \frac{2GM(r)}{r} \\
\beta = \mathrm{e}^{2b} &= \left[1 - \frac{2GM(r)}{r}\right]^{-1} \qquad (3.26)
\end{aligned}$$

where we have used that the mass of a sphere with energy density $\epsilon(r)$ and radius R is[1]

$$M(R) = \int_0^R 4\pi\epsilon(r) r^2 \mathrm{d}r. \qquad (3.27)$$

[1] One might be more familiar with the relation $M(R) = \int_0^R 4\pi\rho(r)r^2\mathrm{d}r$ where ρ is the mass density. However, in units where $c = 1$, we have $\epsilon = \rho c^2 = \rho$.

3.3 Finding the TOV equation

Going back to Einstein's equation and computing the 11 component gives

$$T_{11} = g_{11}T^1_1 = -\mathrm{e}^{2b}[-P(r)] = \frac{1}{\kappa r^2}\left(2ra' - \mathrm{e}^{2b} + 1\right). \tag{3.28}$$

By rearranging the terms and solve for a' we use (3.26) to find

$$\begin{aligned} a' &= \frac{1}{2r}\left(P\kappa r^2 \mathrm{e}^{2b} + \mathrm{e}^{2b} - 1\right) \\ &= \frac{1}{2r}\left(P\kappa r^2 + 2GM(r)\right)\left[1 - \frac{2GM(r)}{r}\right]^{-1} \\ &= \frac{4\pi P(r)Gr^3 + M(r)G}{r\left[r - 2GM(r)\right]}. \end{aligned} \tag{3.29}$$

By definition, the stress-energy tensor has no divergence [18, p. 218]. In other words

$$\nabla_\mu T^{\mu\nu} = \nabla_\mu T_{\mu\nu} = \nabla_\mu T^\mu_\nu = 0, \tag{3.30}$$

where ∇_μ denotes the covariant derivative such that for a tensor $A_{\mu\nu}$ we have

$$\nabla_\sigma A^{\mu\nu} = \partial_\sigma A^{\mu\nu} + \Gamma^\mu_{\sigma\rho}A^{\rho\mu} + \Gamma^\nu_{\rho\sigma}A^{\mu\rho}, \tag{3.31}$$

$$\nabla_\sigma A_{\mu\nu} = \partial_\sigma A_{\mu\nu} - \Gamma^\rho_{\mu\sigma}A_{\rho\nu} - \Gamma^\rho_{\sigma\nu}A_{\mu\rho}, \tag{3.32}$$

$$\nabla_\sigma A^\mu_\nu = \partial_\sigma A^\mu_\nu + \Gamma^\mu_{\sigma\rho}A^\rho_\nu - \Gamma^\rho_{\nu\sigma}A^\mu_\rho. \tag{3.33}$$

Since the fluid is static and spherically symmetric, we have that

$$\partial_t P = \partial_\theta P = \partial_\phi P = \partial_t \epsilon = 0, \tag{3.34}$$

which implies that

$$\nabla_\mu T^\mu_1 = 0. \tag{3.35}$$

Using that T^μ_ν is diagonal, this gives

$$\begin{aligned} 0 =&\partial_\mu T^\mu_1 + \Gamma^\mu_{\mu\rho}T^\rho_1 - \Gamma^\rho_{1\mu}T^\mu_\rho \\ =&\partial_r T^1_1 + \left(\Gamma^0_{01} + \Gamma^1_{11} + \Gamma^2_{21} + \Gamma^3_{31}\right)T^1_1 - \Gamma^0_{10}T^0_0 - \Gamma^1_{11}T^1_1 - \Gamma^2_{21}T^2_2 - \Gamma^3_{31}T^3_3 \\ =& -\partial P(r) - a'\left[P(r) + \epsilon(r)\right] \end{aligned} \tag{3.36}$$

Inserting the expression for a' (3.29), we obtain

$$\begin{aligned} \frac{\mathrm{d}P(r)}{\mathrm{d}r} &= -G\left[P(r) + \epsilon(r)\right]\frac{\left[4\pi P(r)r^3 + M(r)\right]}{r\left[r - 2GM(r)\right]} \\ &= -\frac{G\epsilon(r)M(r)}{r^2}\left[1 + \frac{P(r)}{\epsilon(r)}\right]\left[1 + \frac{4\pi P(r)r^3}{M(r)}\right]\left[1 - \frac{2GM(r)}{r}\right]^{-1}. \end{aligned} \tag{3.37}$$

This equation is often referred to as the Tolman-Oppenheimer-Volkov (TOV) equation after R. C. Tolman, J. R. Oppenheimer and G. M Volkoff, due to the papers their original papers from 1939 [19][20].

3.4 The physics of the TOV equation

We will now briefly discuss the physics of the TOV equation. For this purpose, we will in this section, and this section only, go back to units where $c \neq 1$:

$$\frac{\mathrm{d}P(r)}{\mathrm{d}r} = -\frac{G\epsilon(r)M(r)}{c^2r^2}\left[1+\frac{P(r)}{\epsilon(r)}\right]\left[1+\frac{4\pi P(r)r^3}{c^2M(r)}\right]\left[1-\frac{2GM(r)}{c^2r}\right]^{-1}. \tag{3.38}$$

In the non-relativistic limit, the first two brackets goes as $1+v^2/c^2$ [21] and are therefore corrections to Newtonian gravity from special relativity. Noting that the last bracket is exactly the second diagonal element of the metric tensor, g^{11}, it is clear that this is a correction from general relativity. Flat space-time is hence described by the limit

$$\frac{2M(r)G}{rc^2} \ll 1. \tag{3.39}$$

In the limit when (3.39) is satisfied and

$$\frac{P(r)}{\epsilon(r)} \ll 1, \tag{3.40}$$

$$\frac{4\pi P(r)r^3}{c^2M(r)} \ll 1, \tag{3.41}$$

equation (3.38) simplifies to

$$\frac{\mathrm{d}P(r)}{\mathrm{d}r} = -\frac{G\epsilon(r)M(r)}{c^2r^2}, \tag{3.42}$$

which we recognize as Newton's equation for hydrostatic equilibrium. Note also that the expression (3.38) has a singularity when

$$r = \frac{2GM(r)}{c^2}. \tag{3.43}$$

If r becomes smaller than this quantity, the metric component g_{11} changes sign, and the space-time interval becomes spacelike. The limit (3.43) is often referred to as the Schwarzschild radius, and is a boundary from within information cannot escape. This means that a star at least must have a radius so that

$$R > \frac{2GM(R)}{c^2}, \tag{3.44}$$

otherwise it would not shine; it would be a black hole.

3.5 Solving the TOV equation with constant density

We have three state variables describing the star: The pressure $P(r)$ given by the TOV equation, the mass of the star $M(r)$ given by (3.27) and the energy density $\epsilon(r)$. To be able to solve the system, we now need a third equation giving us an expression for $\epsilon(r)$. Later, we will derive some possible EoS using a few different assumptions regarding the insides of the star, but for now we will assume that the star just has a constant density. This means in physical terms that the star consists of an incompressible fluid, which obviously is a big simplification for a ball of gas. Since the elasticity module of a rigid body is infinite, this also means that the speed of sound inside the star is infinite, which is inconsistent with special relativity. However, it is instructive to see where this leads, and find out how the result differs from Newtonian physics.

With constant density, the mass of the star is

$$M(r) = \frac{4}{3}\pi\epsilon r^3. \tag{3.45}$$

Equation (3.37) then becomes

$$\begin{aligned}\frac{\mathrm{d}P(r)}{\mathrm{d}r} &= -G\left[P(r)+\epsilon\right]\frac{\left[4\pi P(r)r^3+M(r)\right]}{r\left[r-2M(r)G\right]}\\ \frac{\mathrm{d}P(r)}{[3P(r)+\epsilon][P(r)+\epsilon]} &= -\frac{4}{3}\pi G\frac{r\mathrm{d}r}{1-\frac{8}{3}\pi G\epsilon r^2}.\end{aligned} \tag{3.46}$$

Denoting the central pressure at $r=0$ for P_c and integrating both sides from $r=0$ to r we obtain

$$\int_{P_c}^{P}\frac{\mathrm{d}P(r)}{[3P(r)+\epsilon][P(r)+\epsilon]} = -\frac{4}{3}\pi G\int_0^r\frac{r\mathrm{d}r}{1-\frac{8}{3}\pi\epsilon Gr^2}. \tag{3.47}$$

Performing the integral yields

$$\begin{aligned}\frac{1}{2\epsilon}\ln\left\{\frac{[3P(r)+\epsilon][P_c+\epsilon]}{[3P_c+\epsilon][P(r)+\epsilon]}\right\} &= \frac{1}{4\epsilon}\ln\left[1-\frac{8}{3}\pi G\epsilon r^2\right]\\ \frac{[3P(r)+\epsilon][P_c+\epsilon]}{[3P_c+\epsilon][P(r)+\epsilon]} &= \sqrt{1-\frac{8}{3}\pi G\epsilon r^2}\end{aligned} \tag{3.48}$$

By definition, the star has a pressure equal to zero at the surface. Hence, denoting the surface radius R so that $P(R)=0$ we obtain

$$\frac{P_c+\epsilon}{3P_c+\epsilon} = \sqrt{1-\frac{8}{3}\pi G\epsilon R^2}. \tag{3.49}$$

Inserted into (3.48), this gives

$$\begin{aligned}\frac{3P(r)+\epsilon}{P(r)+\epsilon}\sqrt{1-\frac{8}{3}\pi G\epsilon R^2} &= \sqrt{1-\frac{8}{3}\pi G\epsilon r^2}\\ P(r)\left(3\sqrt{1-\frac{8}{3}\pi G\epsilon R^2}-\sqrt{1-\frac{8}{3}\pi G\epsilon r^2}\right) &= \epsilon\left(\sqrt{1-\frac{8}{3}\pi G\epsilon r^2}-\sqrt{1-\frac{8}{3}\pi G\epsilon R^2}\right).\end{aligned} \tag{3.50}$$

Using that the total mass of the sphere is given by

$$M = \frac{4}{3}\pi\epsilon R^3, \tag{3.51}$$

we finally arrive at the pressure:

$$\begin{aligned}P(r) &= \epsilon\frac{\sqrt{1-\frac{8}{3}\pi G\epsilon r^2}-\sqrt{1-\frac{8}{3}\pi G\epsilon R^2}}{3\sqrt{1-\frac{8}{3}\pi G\epsilon R^2}-\sqrt{1-\frac{8}{3}\pi G\epsilon r^2}}\\ &= \epsilon\frac{\sqrt{1-\frac{2MGr^2}{R^3}}-\sqrt{1-\frac{2MG}{R}}}{3\sqrt{1-\frac{2MG}{R}}-\sqrt{1-\frac{2MGr^2}{R^3}}}.\end{aligned} \tag{3.52}$$

From (3.43) we have that the star's Schwarzschild radius is given by the expression

$$r_{\mathrm{schw}} = 2GM(R) = \frac{8\pi\epsilon R^3G}{3}. \tag{3.53}$$

A star cannot have a Schwarzschild radius, and hence for a star with constant density, we always have

$$R > \frac{8\pi G\epsilon R^3}{3}. \tag{3.54}$$

Solving for R we find

$$R < \sqrt{\frac{3}{8\pi G\epsilon}}, \tag{3.55}$$

which is a purely relativistic result. This is also clear from (3.52) since the pressure diverges for $r = R$ when $R = 2GM(R)$. Note that it is also possible to formulate this in terms of the star's mass. Cubing (3.55) and multiplying both sides with $4\pi\epsilon/3$ gives

$$M(R) = \frac{4\pi\epsilon R^3}{3} < \frac{4\pi\epsilon}{3}\left(\frac{3}{8G\pi\epsilon}\right)^{3/2} = \sqrt{\frac{3}{32\pi G^3\epsilon}}. \tag{3.56}$$

To see that this result is connected to relativity, we now look at the Newtonian case for the pressure:

$$\frac{\mathrm{d}P(r)}{\mathrm{d}r} = -\frac{\epsilon M(r)G}{r^2} = -\frac{4\pi\epsilon^2 Gr}{3}. \tag{3.57}$$

Integrating both sides from the central pressure P_c to the pressure P somewhere inside the star gives

$$\begin{aligned}\int_{P_c}^{P}\mathrm{d}P' &= -\frac{4\pi G\epsilon^2}{3}\int_0^r r'\mathrm{d}r' \\ P - P_c &= \frac{2\pi G\epsilon^2 r^2}{3}. \end{aligned}\tag{3.58}$$

Doing the same, except letting the limit go from the central pressure to the pressure at the star's surface, which by our boundary condition is zero, yields

$$\begin{aligned}\int_{P_c}^{0}\mathrm{d}P &= \frac{4\pi G\epsilon^2}{3}\int_0^R r'\mathrm{d}r' \\ P_c &= \frac{2\pi G\epsilon^2 R^2}{3}. \end{aligned}\tag{3.59}$$

Substituted into (3.58) results in

$$P(r) = \frac{2\pi G\epsilon^2}{3}\left(R^2 - r^2\right). \tag{3.60}$$

This expression has no singularities that sets any limits to the star's size or mass, as oppose the expression given by general relativity.

From (3.55) we have a upper limit for the star's radius and mass. But, we can in addition find an even narrower constraint from the pressure obtained in (3.52). To find this limit, we go back to equation (3.49) to see that

$$\begin{aligned}1 - \frac{2MG}{R} &= \left(\frac{P_c + \epsilon}{3P_c + \epsilon}\right)^2 \\ -\frac{2MG}{R} &= \left(\frac{\frac{P_c}{\epsilon} + 1}{3\frac{P_c}{\epsilon} + 1}\right)^2 - 1 \\ \frac{M}{R} &= \frac{1}{2G}\left[1 - \left(\frac{\frac{P_c}{\epsilon} + 1}{3\frac{P_c}{\epsilon} + 1}\right)^2\right]. \end{aligned}\tag{3.61}$$

Since both ϵ and P_c are positive, the maximal ratio between the star's mass and its radius is given when $P_c = 0$:

$$\frac{M}{R} = \frac{1}{2G}\left(1 - \frac{1}{3^2}\right) = \frac{1}{2G}\left(1 - \frac{1}{9}\right) = \frac{4}{9G} \tag{3.62}$$

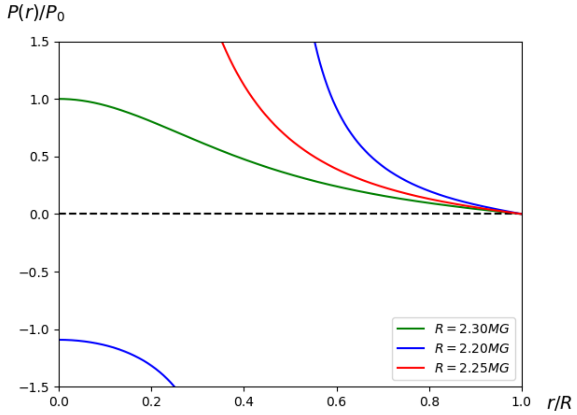


Figure 3.1: Solutions to the TOV equation for a spherically symmetric non-rotating mass distribution with constant density plotted for a few values of the star's radius R. P_0 is some normalization constant chosen so that $P(0) = 1$ for the case $R = 2.3MG$.

Figure 3.1 shows the plot of the pressure for some possible values of R. We observe that as soon as the star's radius goes beyond the limit $R = (9/4)MG = 2.25MG$ the pressure becomes discontinuous, which is unphysical for a star in equilibrium.

We can also rewrite (3.61) in terms of R or M only. Firstly we find the upper limit for R by inserting the star's mass in (3.61):

$$\frac{4\pi\epsilon R^3}{3R} = \frac{4\pi\epsilon R^2}{3} < \frac{4}{9G}. \tag{3.63}$$

Then

$$R < \sqrt{\frac{1}{3G\pi\epsilon}}. \tag{3.64}$$

Cubing this equation and multiplying both sides with $4\pi\epsilon/3$ gives the upper limit for the mass:

$$M < \frac{4\pi\epsilon}{3}\left(\frac{1}{3G\pi\epsilon}\right)^{\frac{3}{2}} = \sqrt{\frac{16}{243\pi G^3\epsilon}}. \tag{3.65}$$

It is fairly intriguing that such a simplified picture as a constant density star actually gives a finite limiting mass. Let us now assume that we have measured a neutron with mass 1.4 solar masses and radius 15 km. This would give us an average density of order $\sim 10^{17}\,\mathrm{kg/m^3}$. This will then result in a maximum mass of ~ 10 solar masses, which astonishingly is of the same order of magnitude as the accepted result today, which is about 2-3 solar masses.

In the Newtonian limit, equation (3.52) must become (3.60) for it to be valid. In this limit, space-time is flat, and thus we have from (3.39) that

$$M \ll R. \tag{3.66}$$

Expanding (3.52) to first order in M/R using that

$$\sqrt{1-x} \approx 1 - \frac{1}{2}x, \quad \text{for} \quad x \ll 1, \tag{3.67}$$

gives

$$P(r) \approx \epsilon \frac{\left(1 - \frac{MGr^2}{R^3}\right) - \left(1 - \frac{MG}{R}\right)}{3\left(1 - \frac{MG}{R}\right) - \left(1 - \frac{MGr^2}{R^3}\right)} = \epsilon \frac{-\frac{MGr^2}{R^3} + \frac{MG}{R}}{2 - 3\frac{MG}{R} + \frac{MGr^2}{R^3}} = \epsilon \frac{-\frac{r^2}{R^2} + 1}{\frac{2R}{MG} - 3\frac{MG}{R} + \frac{r^2}{R^2}}. \tag{3.68}$$

The term

$$\frac{2R}{MG}, \tag{3.69}$$

is much larger than the other two terms in the denominator, so we write

$$P(r) \approx \epsilon \frac{1-\frac{r^2}{R^2}}{\frac{2R}{MG}} = \frac{M\epsilon G}{2R^3}\left(R^2-r^2\right) = \frac{2\pi\epsilon^2 G}{3}\left(R^2-r^2\right), \tag{3.70}$$

which is the same as the result for Newton's equations, as one would expect. Figure 3.2 shows how Newtons equation is a good approximation to the solution of the TOV equation as soon as the ratio M/R becomes small enough. It is also worth noting that the TOV equation always predicts a higher pressure than Newtonian theory. This is because curved space-time predicts higher gravitational forces on a mass, and hence the hydrostatical pressure must be bigger to withstand the forces when the star is in equilibrium.

3.6 Summary

In this chapter we derived the structure equations for a spherically symmetric, non-rotating mass distribution. It was found that in contrast to Newton's equilibrium equation, this equation yields a maximum mass for a given equation of state. It was also found that the TOV equation had a singularity when the star has a radius of $R = 2GM$, known as the Swarzchild radius.

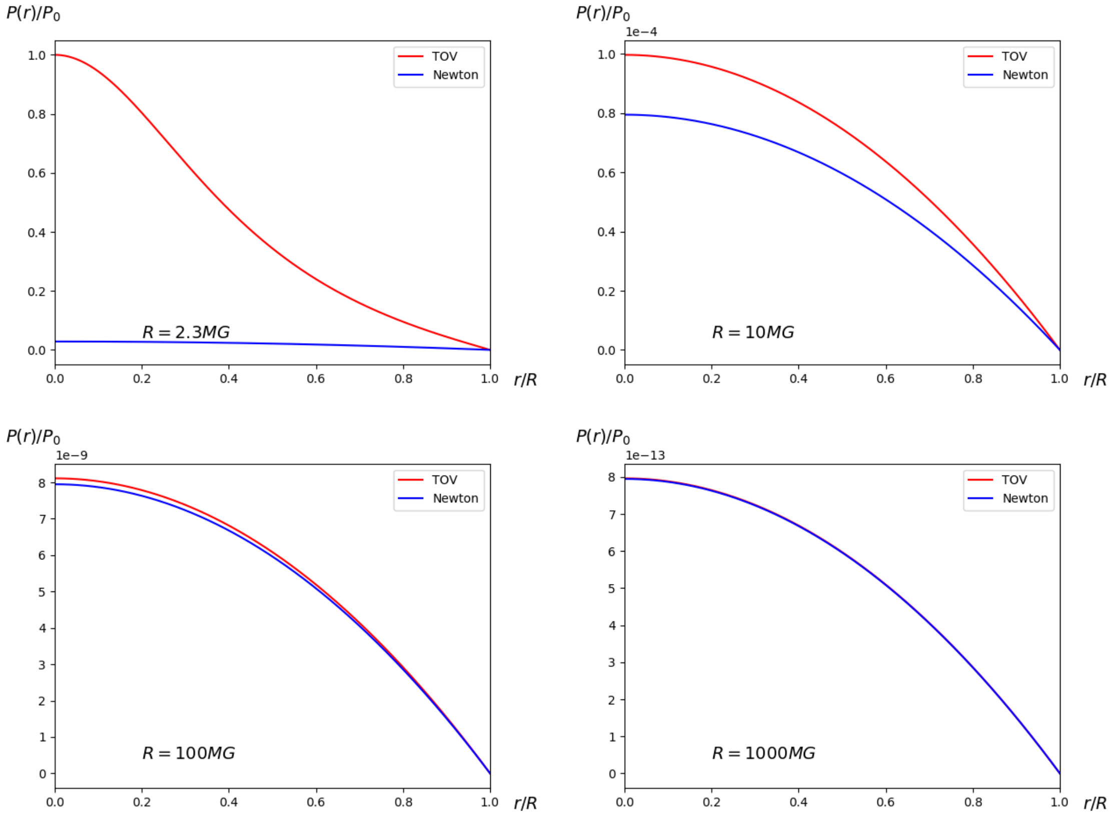


Figure 3.2: Solutions to the structure equations for a spherically symmetric non-rotating mass distribution with constant density using Einstein gravity (TOV) and Newtonian gravity (Newton) plotted for a few values of the star's radius R. P_0 is some normalization constant chosen so that $P(0) = 1$ in the case when $R = 2.3MG$

Chapter 4

Thermodynamics of fermions

In quantum mechanics we are often interested in finding the probability that a given initial state i transitions to some final state f. In this chapter, we will develop Feynman's path integral formalism to calculate these probabilities. Throughout this thesis, possible equations of state for neutron stars consisting mainly of Fermionic matter will be discussed. We will therefore for most consider Fermions when we set sails for this chapter's main goal; finding the mother of all quantities in thermodynamics, namely the partition function.

4.1 The path-integral formalism for a non-relativistic particle in one dimension

In quantum mechanics, the probability of a system starting out in state $|i\rangle$ at time t_i, to end up in state $|f\rangle$ at time t_f, is given by

$$\omega_{fi} = \langle f| \, \mathrm{e}^{-\mathrm{i}(t_f - t_i)\hat{H}(\hat{p},\hat{x})} \, |i\rangle \, . \tag{4.1}$$

This amplitude is often referred to as the propagator, as it propagates the particle from one state to another with probability given by its magnitude. For a non-relativistic free particle traveling in one dimension the Hamiltonian is given by

$$\hat{H}(\hat{p},\hat{x}) = \frac{\hat{p}^2}{2m} + \hat{V}(\hat{x}). \tag{4.2}$$

Here, $\hat{V}$ denotes the potential energy, while $\hat{p}$ and $\hat{x}$ are the momentum and position operators respectively, with eigenvalues and eigenstates defined so that

$$\begin{aligned} \hat{p}\,|p\rangle &= p\,|p\rangle \, , \\ \hat{x}\,|x\rangle &= x\,|x\rangle \, . \end{aligned} \tag{4.3}$$

The state $|p\rangle$ is assumed to form a complete orthogonal set over momentum space, while $|x\rangle$ is assumed to form a complete orthogonal set over position space. In mathematical terms, this means that they by definition satisfy

$$\int |x\rangle \langle x| = \int |p\rangle \langle p| = I, \qquad \text{(Completeness relation)} \tag{4.4}$$

$$\langle x_i|x_j\rangle = \delta(x_i - x_j), \qquad \langle p_i|p_j\rangle = \delta(p_i - p_j), \qquad \text{(Orthogonality relation)} \tag{4.5}$$

where I is the unity operator. Let us now say that we have measured a non-relativistic particle to be in a state $|x_i\rangle$ at time t_i. Then the probability of that particle being in state $|x_f\rangle$ at time t_f is

$$\omega_{fi} = \langle x_f| \, \mathrm{e}^{-\mathrm{i}(t_f - t_i)\hat{H}(\hat{p},\hat{x})} \, |x_i\rangle \, . \tag{4.6}$$

If we then split the time interval in M pieces by inserting $M-1$ complete sets of $|x\rangle$, and define

$$\epsilon = \frac{t_f - t_i}{M}, \tag{4.7}$$

we obtain

$$\omega_{fi} = \int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j \right) \langle x_f | \, \mathrm{e}^{-i\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_{M-1} \rangle \langle x_{M-1} | \, \mathrm{e}^{-i\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_{M-2} \rangle \cdots \langle x_1 | \, \mathrm{e}^{-i\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_i \rangle \, . \tag{4.8}$$

If we now let $M \to \infty$, we see that this integral goes over all possible paths the particle can take between the initial and final state. This method is often referred to as the Feynman path integral approach or Feynman formalism, after Richard P. Feynman.

To investigate further, we insert additional M complete sets of $|p\rangle$ to find

$$\begin{aligned} \omega_{fi} = \int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j \right) \left(\prod_{k=1}^{M} \mathrm{d}p_k \right) \langle x_M | p_M \rangle \langle p_M | \, \mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_{M-1} \rangle \\ \times \langle x_{M-1} | p_{M-1} \rangle \langle p_{M-1} | \, \mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_{M-2} \rangle \cdots \langle x_1 | p_1 \rangle \langle p_1 | \, \mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_0 \rangle \, , \end{aligned} \tag{4.9}$$

where we have defined $x_f \equiv x_M$ and $x_i \equiv x_0$. We see that we may write this expression more compactly by introducing

$$A_j \equiv \langle x_j | p_j \rangle \langle p_j | \, \mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_{j-1} \rangle \, , \tag{4.10}$$

so that

$$\omega_{fi} = \int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j \right) \left(\prod_{k=1}^{M} \mathrm{d}p_k A_k \right). \tag{4.11}$$

To calculate this expression, we now use that ϵ is small and the relation [22]

$$\mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} = \hat{\mathrm{N}}[\mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})}] + \mathcal{O}(\epsilon^2), \tag{4.12}$$

where $\hat{\mathrm{N}}$ is the normal order operator. Then

$$\langle p_k | \, \mathrm{e}^{-\mathrm{i}\epsilon \hat{H}(\hat{p},\hat{x})} \, | x_j \rangle = \langle p_k | \, \mathrm{e}^{-\mathrm{i}\epsilon H(p_k, x_j)} \, | x_j \rangle = \langle p_k | x_j \rangle \, \mathrm{e}^{-\mathrm{i}\epsilon H(p_k, x_j)}, \tag{4.13}$$

in the limit $\epsilon \to 0$ since the normal ordering allows the momentum and position operators to act on the momentum and position eigenstates, respectively. By definition, Euclidean space and momentum space are connected by a Fourier transform

$$\langle x_j | p_k \rangle = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{\mathrm{i} p_k x_j}. \tag{4.14}$$

Using the above results, we find

$$\begin{aligned} A_j &= \langle x_j | p_j \rangle \langle p_j | \, \mathrm{e}^{-\mathrm{i}\epsilon H(\hat{p},\hat{x})} \, | x_{j-1} \rangle \\ &= \langle x_j | p_j \rangle \langle p_j | x_{j-1} \rangle \, \mathrm{e}^{-\mathrm{i}\epsilon H(p_j, x_{j-1})} \\ &= \frac{1}{2\pi} \mathrm{e}^{-\mathrm{i}\left[p_j (x_{j-1} - x_j) + H(p_j, x_{j-1}) \right]}. \end{aligned} \tag{4.15}$$

Inserting the system's Hamiltonian (4.2) this becomes

$$A_j = \frac{1}{2\pi} \mathrm{e}^{\mathrm{i}\epsilon \left[\frac{p_j}{\epsilon} (x_j - x_{j-1}) - \frac{p_j^2}{2m} - V(x_{j-1}) \right]}. \tag{4.16}$$

Integrating over the momentum p_j gives us a standard Gaussian integral with solution

$$\int \mathrm{d}p_j A_j = \sqrt{\frac{m}{2\pi\mathrm{i}\epsilon}}\mathrm{e}^{\mathrm{i}\epsilon\left[\frac{m}{2\epsilon^2}(x_j-x_{j-1})^2-V(x_{j-1})\right]}. \tag{4.17}$$

Letting $M \to \infty$ we then find the transition probability

$$\begin{aligned} \omega_{fi} &= \int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j\right)\left(\frac{m}{2\pi\mathrm{i}\epsilon}\right)^{\frac{M}{2}} \mathrm{e}^{\mathrm{i}\epsilon\sum_{k=1}^{M}\left[\frac{m}{2\epsilon^2}(x_k-x_{k-1})^2-V(x_{k-1})\right]} \\ &= \int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j\right)\left(\frac{m}{2\pi\mathrm{i}\epsilon}\right)^{\frac{M}{2}} \mathrm{e}^{i\int_{t_i}^{t_f}\mathrm{d}t\left[\frac{m}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2-V(x)\right]}, \end{aligned} \tag{4.18}$$

where we have used the definition of the derivative

$$\lim_{\epsilon\to 0}\frac{x_j - x_{j-1}}{\epsilon} = \frac{\mathrm{d}x_j}{\mathrm{d}t}, \tag{4.19}$$

as well as the fact that the exponent is a Riemann sum. Now we define the total measure $\mathcal{D}x(t)$ so that

$$\int \left(\prod_{j=1}^{M-1} \mathrm{d}x_j\right)\left(\frac{m}{2\pi\mathrm{i}\epsilon}\right)^{\frac{M}{2}} \equiv \int_{x_i,t_i}^{x_f,t_f} \mathcal{D}x(t). \tag{4.20}$$

Observing that the exponent of (4.18) is in fact the Lagrangian for a non-relativistic free particle, we can define the action

$$S[x(t)] \equiv \int_{t_i}^{t_f} \mathrm{d}t\, L[x(t)] = \int_{t_i}^{t_f} \mathrm{d}t\left[\frac{m}{2}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2 - V(x)\right]. \tag{4.21}$$

Then the propagator can compactly be written as

$$\omega_{fi} = \int_{x_i,t_i}^{x_f,t_f} \mathcal{D}x(t)\,\mathrm{e}^{\mathrm{i}S[x(t)]}. \tag{4.22}$$

4.2 The path-integral formalism for a relativistic particle

We now want to extend the path integral-formalism to a relativistic Fermion in three spatial dimensions. To do so, we have to abandon the assumption that the system can be completely described by the wave function of the particle. We rather impose that the solutions are given by a composition of fields and move over to the realms of quantum field theory. For more about why quantum mechanics fails, and why quantum field theory is the solution, see for instance [23].

In particle physics, we assume that spacetime is flat. We can do this as gravity is incredible week in relation to the other fundamental forces on small scales. We therefore use the Minkowski spacetime metric $(1, -1, -1, -1)$.

How to generalize the results obtained in the previous section to quantum field theory is thoroughly considered in [24, p. 275-292]. To summarize, our procedure for finding the probability of a Fermion transitioning from an initial state $|\psi_i\rangle$ at time t_i, to some finial state $|\psi_f\rangle$ at time t_f, is closely related to the approach used before: The transition amplitude is still given by

$$\omega_{fi} = \langle\psi_f|\,\mathrm{e}^{-\mathrm{i}(t_f-t_i)\hat{H}}\,|\psi_i\rangle\,. \tag{4.23}$$

However, we need to do some slight changes. For one, we substitute the Lagrangian L with the Lagrangian density $\mathscr{L}$. Secondly, we replace the position operator $\hat{\boldsymbol{x}}(t)$ with a field operator $\hat{\psi}(\boldsymbol{x}, t)$, and the momentum operator $\hat{\boldsymbol{p}}(t)$ with the conjugate momenta operator $\hat{\pi}(\boldsymbol{x}, t)$ defined as

$$\hat{\pi}(\boldsymbol{x}, t) = \frac{\partial\hat{\mathscr{L}}(\hat{\psi}, \hat{\pi})}{\partial(\partial_0\hat{\psi})}. \tag{4.24}$$

The field $\hat{\psi}$ and conjugate momenta $\hat{\pi}$ have eigenstates given by

$$\hat{\psi}(\boldsymbol{x},0)\left|\psi\right\rangle = \psi(\boldsymbol{x})\left|\psi\right\rangle, \tag{4.25}$$
$$\hat{\pi}(\boldsymbol{x},0)\left|\pi\right\rangle = \pi(\boldsymbol{x})\left|\pi\right\rangle, \tag{4.26}$$

and are assumed to form complete orthogonal sets so that

$$\int \mathrm{d}\psi \left|\psi\right\rangle\left\langle\psi\right| = \int \mathrm{d}\pi \left|\pi\right\rangle\left\langle\pi\right| = I, \qquad \text{(Completeness)} \tag{4.27}$$
$$\left\langle\psi_a|\psi_b\right\rangle = \prod_{\boldsymbol{x}} \delta\left[\psi_a(\boldsymbol{x}) - \psi_b(\boldsymbol{x})\right], \quad \left\langle\pi_a|\pi_b\right\rangle = \prod_{\boldsymbol{x}} \delta\left[\pi_a(\boldsymbol{x}) - \pi_b(\boldsymbol{x})\right]. \qquad \text{(Orthogonality)} \tag{4.28}$$

Fermions must also follow the anti-commutation relations

$$\left\{\hat{\psi}_\alpha(\boldsymbol{x},t), \hat{\psi}^\dagger_\beta(\boldsymbol{y},t)\right\} = \delta_{\alpha\beta}\delta(\boldsymbol{x}-\boldsymbol{y}), \tag{4.29}$$
$$\left\{\hat{\psi}_\alpha(\boldsymbol{x},t), \hat{\psi}_\beta(\boldsymbol{y},t)\right\} = \left\{\hat{\psi}^\dagger_\alpha(\boldsymbol{x},t), \hat{\psi}^\dagger_\beta(\boldsymbol{y},t)\right\} = 0, \tag{4.30}$$

where ψ_α denotes the α component of the four dimensional spinor ψ. It can be shown [25, p. 28] that this relation implies that ψ is anti-periodic, which means that if the system returns to its initial state after some time t_{per}, then

$$\psi(\boldsymbol{x},0) = -\psi(\boldsymbol{x},t_{\mathrm{per}}). \tag{4.31}$$

Let us now assume that we have a system in a state ψ_i at $t_i = 0$. The probability that the system is in state ψ_f at time t_f is then

$$\omega_{fi} = \left\langle\psi_f\right| \mathrm{e}^{-i\hat{H}(\hat{\psi},\hat{\pi})t_f} \left|\psi_i\right\rangle. \tag{4.32}$$

As before, we split the Hamiltonian in M products by introducing

$$\epsilon = \frac{t_f - t_i}{M} = \frac{t_f}{M}, \tag{4.33}$$

and inserting complete sets of ψ and π:

$$\begin{aligned}\omega_{fi} &= \int \left(\prod_{j=1}^{M-1} \mathrm{d}\psi_j\right)\left(\prod_{k=1}^{M} \mathrm{d}\pi_k\right) \left\langle\psi_f|\pi_M\right\rangle \left\langle\pi_M\right| \mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\psi},\hat{\pi})} \left|\psi_{M-1}\right\rangle \cdots \left\langle\psi_1|\pi_1\right\rangle \left\langle\pi_1\right| \mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\psi},\hat{\pi})} \left|\psi_i\right\rangle \\ &= \int \left(\prod_{j=1}^{M} \mathrm{d}\psi_j \mathrm{d}\pi_j\right) \left\langle\psi_f|\pi_M\right\rangle \left\langle\pi_M\right| \mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\psi},\hat{\pi})} \left|\psi_{M-1}\right\rangle \cdots \\ &\qquad \cdots \left\langle\psi_1|\pi_1\right\rangle \left\langle\pi_1\right| \mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\psi},\hat{\pi})} \left|\psi_i\right\rangle \prod_{\boldsymbol{x}} \delta[\psi_f(\boldsymbol{x}) - \psi_M(\boldsymbol{x})].\end{aligned} \tag{4.34}$$

Letting M go to infinity, we have from (4.12)

$$\left\langle\pi_j\right| \mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\psi},\hat{\pi})} \left|\psi_j\right\rangle = \left\langle\pi_j|\psi_j\right\rangle \mathrm{e}^{-\mathrm{i}\epsilon H(\psi_j,\pi_j)}, \tag{4.35}$$

which combined with the overlap between the field ψ and the conjugate momenta π,

$$\left\langle\psi_j|\pi_k\right\rangle = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{\mathrm{i}\int \mathrm{d}^3x\, \pi_k \psi_j}, \tag{4.36}$$

gives

$$
\begin{aligned}
\langle\psi_j|\pi_j\rangle\,\langle\pi_j|\,\mathrm{e}^{-\mathrm{i}\epsilon\hat{H}(\hat{\pi},\hat{\psi})}\,|\psi_{j-1}\rangle &= \frac{1}{\sqrt{2\pi}}\mathrm{e}^{\mathrm{i}\int \mathrm{d}^3x\,\pi_j\psi_j}\,\langle\pi_j|\psi_{j-1}\rangle\,\mathrm{e}^{-\mathrm{i}\epsilon H(\pi_j,\psi_{j-1})} \\
&= \frac{1}{2\pi}\mathrm{e}^{\mathrm{i}\int \mathrm{d}^3x\,\pi_j(\psi_j-\psi_{j-1})-\mathrm{i}\epsilon H(\pi_j,\psi_{j-1})} \\
&= \frac{1}{2\pi}\mathrm{e}^{\mathrm{i}\epsilon\left[\frac{1}{\epsilon}\int \mathrm{d}^3x\,\pi_{j-1}(\psi_j-\psi_{j-1})-H(\pi_j,\psi_{j-1})\right]} \\
&= \frac{1}{2\pi}\mathrm{e}^{\mathrm{i}\epsilon\int \mathrm{d}^3x\,\left[\pi_j\frac{1}{\epsilon}(\psi_j-\psi_{j-1})-\mathscr{H}(\pi_j,\psi_{j-1})\right]}
\end{aligned}
\tag{4.37}
$$

Hence,

$$
\begin{aligned}
\omega_{fi} &= \int\left(\prod_{k=1}^{M}\frac{\mathrm{d}\psi_k\mathrm{d}\pi_k}{2\pi}\right)\mathrm{e}^{\mathrm{i}\epsilon\sum_{j=1}^{M}\int\mathrm{d}^3x\,\left[\pi_j\frac{\psi_{j+1}-\psi_j}{\epsilon}-\mathscr{H}(\psi_j,\pi_j)\right]}\delta(\psi_f-\psi_M) \\
&= N\int_{\psi(\boldsymbol{x},0)}^{\psi(\boldsymbol{x},t_f)}\mathcal{D}\psi\mathcal{D}\pi\mathrm{e}^{\mathrm{i}\int_0^{t_f}\,\mathrm{d}t\int\mathrm{d}^3x\,\left[\pi\frac{\mathrm{d}\psi}{\mathrm{d}t}-\mathscr{H}(\psi,\pi)\right]}\delta(\psi_f-\psi_M) \\
&= N\int_{\psi(\boldsymbol{x},0)}^{\psi(\boldsymbol{x},t_f)}\mathcal{D}\psi\mathcal{D}\pi\mathrm{e}^{\mathrm{i}\int_0^{t_f}\,\int\mathrm{d}^3x\,\mathscr{L}(\psi,\pi)}\delta(\psi_f-\psi_M) \\
&= N\int_{\psi(\boldsymbol{x},0)}^{\psi(\boldsymbol{x},t_f)}\mathcal{D}\psi\mathcal{D}\pi\mathrm{e}^{\mathrm{i}S(\psi,\pi)}\delta(\psi_f-\psi_M),
\end{aligned}
\tag{4.38}
$$

where we have defined

$$
\mathcal{D}\psi=\prod_{j=1}^{M}\mathrm{d}\psi_j,\quad \mathcal{D}\pi=\prod_{j=1}^{M}\mathrm{d}\pi_j,\quad N=\prod_{j=1}^{M}\frac{1}{2\pi},\quad \prod_{\boldsymbol{x}}\delta[\psi_f(\boldsymbol{x})-\psi_M(\boldsymbol{x})]=\delta(\psi_f-\psi_M),
\tag{4.39}
$$

and used that the systems Lagrangian density and Hamiltonian density are connected through a Legendre transform

$$
\mathscr{L}(\psi,\pi)=\pi(\boldsymbol{x},t)\frac{\mathrm{d}\psi}{\mathrm{d}t}-\mathscr{H}(\psi,\pi).
\tag{4.40}
$$

When $M\to\infty$, the delta function in equation (4.39) is of no significance. Therefore, we just leave it out of the calculations from here on.

4.3 Finding the partition function for a medium

Before we can move on and find the thermodynamic properties of Fermionic matter, we need some statistical physics. Quantities such as pressure, free energy, density and chemical potential can all be found using the system's partition function. For a system with constant number of particles, one uses the canonical partition function, which is a sum over all possible energy configurations the system can have [26, p. 55]

$$
Z=\sum_i \mathrm{e}^{-\beta E_i}.
\tag{4.41}
$$

In the above expression, we have defined β as the inverse temperature

$$
\beta=\frac{1}{T}.
\tag{4.42}
$$

For a system with Hamiltonian H, the partition function can be written in a basis of number eigenstates $|i\rangle$ as a trace

$$
Z=\sum_i \mathrm{e}^{-\beta E_i}=\sum_i\langle i|\,\mathrm{e}^{-\beta\hat{H}}\,|i\rangle=\mathrm{Tr}\Big[\mathrm{e}^{-\beta\hat{H}}\Big].
\tag{4.43}
$$

The number eigenstates are assumed to form a complete set, so that

$$\sum_{i=0}^{\infty} |i\rangle \langle i| = I, \tag{4.44}$$

and hence we can write the partition function in terms of the spinor eigenstates instead:

$$Z = \sum_i \langle i| \,\mathrm{e}^{-\beta \hat{H}} \,|i\rangle = \sum_i \int \mathrm{d}\psi \, \langle i|\psi\rangle \, \langle\psi| \,\mathrm{e}^{-\beta\hat{H}} \,|i\rangle = \sum_i \int \mathrm{d}\psi \, \langle\psi| \,\mathrm{e}^{-\beta\hat{H}} \,|i\rangle \, \langle i|\psi\rangle = \int \mathrm{d}\psi \, \langle\psi| \,\mathrm{e}^{-\beta\hat{H}} \,|\psi\rangle \, . \tag{4.45}$$

Note that from (4.38) we have

$$\langle\psi| \,\mathrm{e}^{-\mathrm{i}\beta\hat{H}} \,|\psi\rangle = N \int_{\psi(\boldsymbol{x},0)}^{\psi(\boldsymbol{x},\beta)} \mathcal{D}\pi\mathcal{D}\psi \mathrm{e}^{\mathrm{i}S(\psi,\pi)} \equiv N \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\pi\mathcal{D}\psi \mathrm{e}^{\mathrm{i}\int_0^\beta \mathrm{d}t \int \mathrm{d}^3x \mathscr{L}(\psi,\pi)}, \tag{4.46}$$

where we in the last step used the anti-periodc form (4.31) of Fermionic fields. This means that if we perform a $\pi/2$ counter clockwise turn in the complex plane by the substitution t = -iτ, or in other words, transforming the calculations from Minkowski space to Euclidean space, we find

$$\begin{aligned} Z &= \int \mathrm{d}\psi \, \langle\psi| \,\mathrm{e}^{-\beta\hat{H}} \,|\psi\rangle \\ &= N \int \mathrm{d}\psi \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\pi\mathcal{D}\psi \mathrm{e}^{\mathrm{i}(-\mathrm{i})\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \, [-\mathscr{L}_{\mathrm{E}}(\psi,\pi)]} \\ &= N \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\pi\mathcal{D}\psi \mathrm{e}^{-\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \, \mathscr{L}_{\mathrm{E}}(\psi,\pi)} \\ &= N \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\pi\mathcal{D}\psi \mathrm{e}^{-S_{\mathrm{E}}(\psi,\pi)}, \end{aligned} \tag{4.47}$$

where we have defined $\mathcal{L}_{\mathrm{E}}$ and S_E as the Lagrangian density and action in Euclidean space, respectively. The extra minus sign in the exponential comes from the fact that we transformed the Minkowski metric $(1, -1 -1 -1)$ to the negative of the Euclidean metric $(1, 1, 1, 1)$. We conclude that the partition function is connected to the transition amplitudes ω_{fi} by a Wick rotation. Also, the thermodynamics of a system is not changed if we multiply the partition function by a constant. Hence, we can just forget it and write

$$Z = \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\psi\mathcal{D}\pi \mathrm{e}^{\mathrm{i}S(\psi,\pi)}. \tag{4.48}$$

Let us now consider a system with varying number of particles. We then use the grand canonical partition function

$$Z = \mathrm{Tr}\left[\mathrm{e}^{-\beta\left(\hat{H}-\mu\hat{N}\right)}\right], \tag{4.49}$$

where μ is the chemical potential and $\hat{N}$ is the number operator. In field theory, particles may decay, annihilate, etc. and so the particle number is not always a conserved quantity. We then more generally insert the charge operator $\hat{Q}_i$ with corresponding chemical potential μ_i for each conserved charge Q_i the system has, so that

$$Z = \mathrm{Tr}\left[\mathrm{e}^{-\beta(\hat{H}-\mu_i\hat{Q}_i)}\right]. \tag{4.50}$$

A conserved charge is a quantity that is conserved due to a continuous symmetry in the system's Lagrangian[1]. This could in some cases be the number of particles, but it could also be the electric charge, isospin, colour charge, and so on. The chemical potential μ_i is defined so that it corresponds to the energy needed to add one more charge Q_i to the system. This explains the minus sign convention

[1] See Noether's theorem in Appendix E.

in front of μ_i: If μ_i is positive, we need energy to add a charge Q_i to the system. From (4.50) we find that the easiest way to define the Hamiltonian density is

$$\hat{\mathscr{H}} = \hat{\mathscr{H}_0} - \mu_i \hat{\rho}_i, \tag{4.51}$$

where $\mathscr{H}_0$ is the zero-density Hamiltonian of the system and $\hat{\rho}_i$ is the i-th charge density operator. Written in terms of the action S_E, the result is still on the same form as before,

$$Z = \int_{\psi(\boldsymbol{x},0)}^{-\psi(\boldsymbol{x},0)} \mathcal{D}\pi \mathcal{D}\psi \mathrm{e}^{-S_E(\psi,\pi)}, \tag{4.52}$$

except that the action now contains an extra term $\mu_i \hat{q}_i$.

4.4 Calculating the partition function

We are now finally ready to calculate the partition function for the Fermionic field. Without interactions, this field is described by the Dirac Lagrangian

$$\mathscr{L} = \mathrm{i}\bar{\psi}\gamma^\mu \partial_\mu \psi - m\bar{\psi}\psi = \bar{\psi}\left(\mathrm{i}\not{\partial} - m\right)\psi, \tag{4.53}$$

where we have introduced the gamma matrices γ^μ given by (C.9), and used the bar notation for Fermions $\bar{\psi} = \gamma^0\psi^\dagger$, as well as the Feynman slash notation $\partial^\mu \gamma_\mu = \not{\partial}$. This Lagrangian is invariant under a global phase-transformation $\psi \to \psi' = \psi \mathrm{e}^{-\mathrm{i}\alpha}$, which we see by insertion:

$$\mathscr{L} \to \mathscr{L}' = \bar{\psi}\mathrm{e}^{i\alpha}\left(\mathrm{i}\not{\partial} - m\right)\psi \mathrm{e}^{-\mathrm{i}\alpha} = \bar{\psi}\left(\mathrm{i}\not{\partial} - m\right)\psi = \mathscr{L}. \tag{4.54}$$

Then, by Noether's theorem, we find from (E.7) the conserved current

$$\begin{aligned}
j^\mu &= \frac{\partial \mathscr{L}}{\partial\left(\partial_\mu \psi\right)}\delta\psi + \frac{\partial \mathscr{L}}{\partial\left(\partial_\mu \bar{\psi}\right)}\delta\bar{\psi} \\
&= \frac{\partial \mathscr{L}}{\partial\left(\partial_\mu \psi\right)}\left(-\mathrm{i}\psi\right) + \frac{\partial \mathscr{L}}{\partial\left(\partial_\mu \bar{\psi}\right)}\left(\mathrm{i}\bar{\psi}\right) \\
&= \frac{\partial}{\partial(\partial_\mu \psi)}\left(\mathrm{i}\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi\right)\left(-\mathrm{i}\psi\right) + \frac{\partial}{\partial(\partial_\mu \bar{\psi})}\left(\mathrm{i}\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi\right)\left(\mathrm{i}\bar{\psi}\right) \\
&= \mathrm{i}\bar{\psi}\gamma^\mu\left(-\mathrm{i}\psi\right) \\
&= \bar{\psi}\gamma^\mu\psi.
\end{aligned} \tag{4.55}$$

This corresponds to conservation of particle number. The system then has a corresponding conserved charge Q given by (E.8):

$$Q = \int \mathrm{d}^3x\, j^0 = \int \mathrm{d}^3x\, \bar{\psi}\gamma^0\psi = \int \mathrm{d}^3x\, \psi^\dagger\psi, \tag{4.56}$$

which in this case means probability conservation. Introducing the conserved charge, we re-define the Hamiltonian density as

$$\mathscr{H} = \mathscr{H}_0 - \mu\psi^\dagger\psi. \tag{4.57}$$

Because of (4.40), we find that this is the same as writing

$$\mathscr{L} = \mathscr{L}_0 + \mu\psi^\dagger\psi, \tag{4.58}$$

where

$$\mathscr{L}_0 = \bar{\psi}\left(\mathrm{i}\not{\partial} - m\right)\psi. \tag{4.59}$$

It should be mentioned that including a chemical potential in the Lagrangian breaks Lorentz invariance. We can see this from a physical prospective: Since the chemical potential tells us how much work we need to do to insert another particle, having zero chemical potential means that the particle is in a vacuum. In a vacuum, all directions and speeds are equivalent, since we are free to choose whatever rest frame we want. When the chemical potential is non-zero, the particle moves in a medium, which means its position and velocity is measured relatively to this medium. Now different speeds and positions are not equivalent anymore. Unless the medium is uniform, we would also have different properties at different places in the medium.

Having introduced a chemical potential term, the Lagrangian density in Euclidean space becomes

$$\begin{aligned}\mathscr{L}_{\mathrm{E}} &= -\bar{\psi}_{\mathrm{E}}\bigg(\mathrm{i}\gamma^0\frac{\partial}{-\mathrm{i}\partial\tau} + \mathrm{i}\boldsymbol{\gamma}\cdot\nabla - m + \mu\bigg)\psi_{\mathrm{E}} \\ &= \bar{\psi}_{\mathrm{E}}\bigg(\gamma^0\frac{\partial}{\partial\tau} + \boldsymbol{\gamma}_{\mathrm{E}}\cdot\nabla + m - \mu\bigg)\psi \\ &\equiv \bar{\psi}_{\mathrm{E}}(\not{\partial}_{\mathrm{E}} + m - \mu)\psi_{\mathrm{E}}, \end{aligned} \tag{4.60}$$

where γ^μ_{E} denotes the Dirac matrices defined by (C.10), and we have used that $(\gamma^0)^2 = (\gamma^0_{\mathrm{E}})^2 = 1$. We have also written the Dirac spinors in Euclidean space as ψ_{E}, and defined

$$\not{\partial}_{\mathrm{E}} = \partial_\mu\gamma^\mu_{\mathrm{E}}, \tag{4.61}$$

in the last step. Throughout this chapter we will skip the E subscript, as all calculations will take place in Euclidean space. To proceed, we expand the field ψ in frequency momentum space as a Fourier series

$$\psi(\boldsymbol{x}, t) = \frac{1}{\sqrt{\beta V}}\sum_{n=-\infty}^{\infty}\sum_{\boldsymbol{p}}\psi_{n,\boldsymbol{p}}\mathrm{e}^{\mathrm{i}(\omega_n t + \boldsymbol{p}\cdot\boldsymbol{x})}, \tag{4.62}$$

with coefficients

$$\psi_{n,\boldsymbol{p}} = \frac{1}{\sqrt{\beta V}}\int_0^\beta \mathrm{d}t\int \mathrm{d}^3x\,\psi(\boldsymbol{x}, t)\mathrm{e}^{-\mathrm{i}(\omega_n t + \boldsymbol{p}\cdot\boldsymbol{x})}, \tag{4.63}$$

where ω_n are the Matsubara frequencies defined by (D.1) and V is the three-volume of the system. The action then becomes

$$\begin{aligned} S =& \int_0^\beta \mathrm{d}\tau\int\mathrm{d}^3x\left[\bar{\psi}\left(\gamma^0\frac{\partial}{\partial\tau} + \boldsymbol{\gamma}\cdot\nabla + m - \mu\gamma^0\right)\psi\right] \\ =& \frac{1}{\beta V}\int_0^\beta\mathrm{d}\tau\int\mathrm{d}^3x \\ &\times\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty}\sum_{\boldsymbol{p}}\sum_{\boldsymbol{p}'}\bar{\psi}_{n,\boldsymbol{p}}\mathrm{e}^{-\mathrm{i}(\omega_n\tau+\boldsymbol{p}\cdot\boldsymbol{x})}\left(\gamma^0\frac{\partial}{\partial\tau} + \boldsymbol{\gamma}\cdot\nabla + m - \mu\gamma^0\right)\psi_{m,\boldsymbol{p}'}\mathrm{e}^{\mathrm{i}\left(\omega_m\tau + \boldsymbol{p}'\cdot\boldsymbol{x}\right)} \\ =& \frac{1}{\beta V}\int_0^\beta\mathrm{d}\tau\int\mathrm{d}^3x\sum_{n,m,\boldsymbol{p},\boldsymbol{p}'}\bar{\psi}_{n,\boldsymbol{p}}\left(\mathrm{i}\gamma^0\omega_m + \mathrm{i}\boldsymbol{\gamma}\cdot\boldsymbol{p}' + m - \mu\gamma^0\right)\psi_{m,\boldsymbol{p}'}\mathrm{e}^{\mathrm{i}\left[(\omega_m-\omega_n)\tau + (\boldsymbol{p}'-\boldsymbol{p})\cdot\boldsymbol{x}\right]} \\ =& \frac{1}{\beta V}\sum_{n,m,\boldsymbol{p},\boldsymbol{p}'}\bar{\psi}_{n,\boldsymbol{p}}\left(\mathrm{i}\gamma^0\omega_m + \mathrm{i}\boldsymbol{\gamma}\cdot\boldsymbol{p}' + m - \mu\gamma^0\right)\psi_{m,\boldsymbol{p}'}\beta V\delta(\omega_n - \omega_n)\delta(\boldsymbol{p} - \boldsymbol{p}') \\ =& \sum_{n,\boldsymbol{p}}\bar{\psi}_{n,\boldsymbol{p}}\left[\mathrm{i}\gamma^0\omega_n + \mathrm{i}\boldsymbol{\gamma}\cdot\boldsymbol{p} + m - \mu\gamma^0\right]\psi_{n,\boldsymbol{p}}. \end{aligned} \tag{4.64}$$

Now we can calculate the partition function:

$$Z = \int\mathcal{D}\pi\mathcal{D}\psi\mathrm{e}^{-\sum_{n,\boldsymbol{p}}\bar{\psi}_{n,\boldsymbol{p}}\left[\gamma^0(\mathrm{i}\omega_n - \mu) + \mathrm{i}\boldsymbol{\gamma}\cdot\boldsymbol{p} + m\right]\psi_{n,\boldsymbol{p}}}. \tag{4.65}$$

The conjugate momenta of this theory is given by

$$\pi(\boldsymbol{x}, t) = \frac{\partial\mathscr{L}}{\partial(\partial_0\psi)} = \frac{\partial}{\partial(\partial_0\psi)}\left[\bar{\psi}\left(\mathrm{i}\not{\partial} + m\right)\psi + \mu\psi^\dagger\psi\right] = \mathrm{i}\psi^\dagger, \tag{4.66}$$

and we obtain

$$
\begin{aligned}
Z &= \int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \mathrm{e}^{-\sum_{n,\boldsymbol{p}} \bar{\psi}_{n,\boldsymbol{p}}\left[\gamma^0(\mathrm{i}\omega_n-\mu)+\mathrm{i}\boldsymbol{\gamma}\cdot\boldsymbol{p}+m\right]\psi_{n,\boldsymbol{p}}} \\
&= \int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \mathrm{e}^{\sum_{n,\boldsymbol{p}} \mathrm{i}\psi^\dagger_{n,\boldsymbol{p}}\left[-(\omega_n+\mathrm{i}\mu)-\gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{p}+\mathrm{i}\gamma^0 m\right]\psi_{n,\boldsymbol{p}}}. \qquad (4.67)
\end{aligned}
$$

The expression sandwiched between the spinors in the exponential is a matrix, and hence we have an Gaussian integral over the Grassmann variables ψ and $\mathrm{i}\psi^\dagger$. Then we can use that for a matrix D, [25, p. 27]

$$
\int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \mathrm{e}^{\mathrm{i}\psi^\dagger D\psi} = \int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \mathrm{e}^{\sum_{n,\boldsymbol{p}} \mathrm{i}\psi^\dagger_{n,\boldsymbol{p}} D\psi_{n,\boldsymbol{p}}} = \det D. \qquad (4.68)
$$

If we call the eigenvalues of the matrix d_i, we find that

$$
\ln\det D = \ln\left(\prod_i d_i\right) = \ln \mathrm{e}^{\sum_i \ln d_i} = \ln \mathrm{e}^{\mathrm{Tr}\ln D} = \mathrm{Tr}\ln D. \qquad (4.69)
$$

We can use this, as well as the relation[2]

$$
(\boldsymbol{\sigma}\cdot\boldsymbol{p})^2 = \boldsymbol{p}^2, \qquad (4.70)
$$

to obtain

$$
\begin{aligned}
\ln Z &= \ln\det\left[-(\omega_n+\mathrm{i}\mu)-\gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{p}+\mathrm{i}\gamma^0 m\right] \\
&= \ln\det\left[\begin{pmatrix} -(\omega_n+\mathrm{i}\mu) & 0 \\ 0 & -(\omega_n+\mathrm{i}\mu)\end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & -1\end{pmatrix}\begin{pmatrix} 0 & -\mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{p} \\ \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{p} & 0\end{pmatrix} + \begin{pmatrix} \mathrm{i}m & 0 \\ 0 & -\mathrm{i}m\end{pmatrix}\right] \\
&= \ln\det\begin{pmatrix} -\left[(\omega_n+\mathrm{i}\mu)-\mathrm{i}m\right] & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{p} \\ \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{p} & -\left[(\omega_n+\mathrm{i}\mu)+\mathrm{i}m\right]\end{pmatrix} \\
&= \ln\det\Big[(\omega_n+\mathrm{i}\mu)^2+m^2+(\boldsymbol{\sigma}\cdot\boldsymbol{p})^2\Big] \\
&= \ln\det\Big[(\omega_n+\mathrm{i}\mu)^2+m^2+\boldsymbol{p}^2\Big] \\
&= \ln\Big[(\omega_n+\mathrm{i}\mu)^2+E_{\boldsymbol{p}}^2\Big]^2 \\
&= \mathrm{Tr}\ln\Big[(\omega_n+\mathrm{i}\mu)^2+E_{\boldsymbol{p}}^2\Big]^2 \\
&= \sum_{n,\boldsymbol{p}}\ln\left[(\omega_n+\mathrm{i}\mu)^2+E_{\boldsymbol{p}}^2\right]^2 \qquad (4.71)
\end{aligned}
$$

where we have defined $E_{\boldsymbol{p}}^2 = \boldsymbol{p}^2 + m^2$. It is important to stress that we by convention assume that numbers are multiplied with a suiting identity matrix so that our expressions make sense. For example, we have

$$
\omega_n + \boldsymbol{\sigma}\cdot\boldsymbol{p} \equiv \omega_n I_2 + \boldsymbol{\sigma}\cdot\boldsymbol{p}, \qquad (4.72)
$$

where I_n denotes the $n\times n$ identity matrix.

4.5 Finding the grand potential

Now that we have the partition function for the Fermionic field, we can calculate some of the thermodynamic quantities. In this section, we consider the grand potential defined by

$$
\Omega_{\mathrm{G}} \equiv U - TS - \mu_i Q_i, \qquad (4.73)
$$

[2] This follows from the commutation relations of the Pauli matrices (C.7).

where U is the systems internal energy, T is the temperature and S is the entropy. We note that the first two terms represent the internal energy minus the energy we can get from the environment. The last part gives the energy needed to add new charges Q to the system. In the following, we will look at the grand potential per volume

$$\Omega = \frac{\Omega_{\mathrm{G}}}{V}. \tag{4.74}$$

To calculate this quantity, we first note that the grand canonical partition function is in addition to (4.41), also defined by the relation

$$Z = \mathrm{e}^{-\beta V \Omega} = \mathrm{e}^{\beta V P}, \tag{4.75}$$

where P is the pressure. The grand potential density, and then also the pressure, is given by

$$\Omega = -P = -\frac{1}{\beta V}\ln Z = -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \ln\left[(\omega_n + \mathrm{i}\mu)^2 + E_{\boldsymbol{p}}^2\right]^2. \tag{4.76}$$

Since n vary over all positive and negative integers, we may write [25, p. 29]

$$\begin{aligned}
\Omega &= -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \ln\Big\{\Big[\big(\omega_n + \mathrm{i}\mu\big)^2 + E_{\boldsymbol{p}}^2\Big]\Big[\big(-\omega_n + \mathrm{i}\mu\big)^2 + E_{\boldsymbol{p}}^2\Big]\Big\} \\
&= -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \ln\Big[\big(\omega_n^2 - \mu^2 + E_{\boldsymbol{p}}^2 + 2\omega_n \mathrm{i}\mu\big)\big(\omega_n^2 - \mu^2 + E_{\boldsymbol{p}}^2 - 2\omega_n \mathrm{i}\mu\big)\Big] \\
&= -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \ln\Big\{\Big[\omega_n - \mathrm{i}\big(E_{\boldsymbol{p}} - \mu\big)\Big]\Big[\omega_n + \mathrm{i}\big(E_{\boldsymbol{p}} + \mu\big)\Big]\Big[\omega_n + \mathrm{i}\big(E_{\boldsymbol{p}} - \mu\big)\Big]\Big[\omega_n - \mathrm{i}\big(E_{\boldsymbol{p}} + \mu\big)\Big]\Big\} \\
&= -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \ln\Big\{\Big[\omega_n^2 + \big(E_{\boldsymbol{p}} - \mu\big)^2\Big]\Big[\omega_n^2 + \big(E_{\boldsymbol{p}} + \mu\big)^2\Big]\Big\} \\
&= -\frac{1}{\beta V}\sum_{n,\boldsymbol{p}} \Big\{\ln\Big[\omega_n^2 + (E_{\boldsymbol{p}} - \mu)^2\Big] + \ln\Big[\omega_n^2 + (E_{\boldsymbol{p}} + \mu)^2\Big]\Big\}.
\end{aligned} \tag{4.77}$$

Using the identity [27]

$$\frac{1}{2\beta}\sum_n \ln\left[\omega_n^2 + \epsilon^2\right] = \frac{1}{2}\epsilon + \frac{1}{\beta}\ln\left[1 + \mathrm{e}^{-\beta\epsilon}\right], \tag{4.78}$$

we find that

$$\begin{aligned}
\Omega &= -\frac{2}{V}\sum_{\boldsymbol{p}} \left\{\frac{1}{2}\Big[E_{\boldsymbol{p}} - \mu\Big] + \frac{1}{\beta}\ln\Big[1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} - \mu)}\Big] + \frac{1}{2}\Big[E_{\boldsymbol{p}} + \mu\Big] + \frac{1}{\beta}\ln\Big[1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} + \mu)}\Big]\right\} \\
&= -\frac{2}{V}\sum_{\boldsymbol{p}} \left\{E_{\boldsymbol{p}} + T\Big[\ln\Big(1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} + \mu)}\Big) + \ln\Big(1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} - \mu)}\Big)\Big]\right\}.
\end{aligned} \tag{4.79}$$

In the continuum limit, we may replace the sum over momenta with an integral

$$\sum_{\boldsymbol{p}} \to V\int \frac{\mathrm{d}^3 p}{(2\pi)^3}, \tag{4.80}$$

and we arrive at the grand potential

$$\Omega = -2\int \frac{\mathrm{d}^3 p}{(2\pi)^3}\left\{E_{\boldsymbol{p}} + T\Big[\ln\Big(1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} - \mu)}\Big) + \ln\Big(1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}} + \mu)}\Big)\Big]\right\}. \tag{4.81}$$

We recognize the fist term as four times the contribution from the zero-point energy. The factor four comes from the fact that a spin $\frac{1}{2}$-particle has two spin states, combined with the contribution from particles and ant-particles. This term is infinite, and we will discuss how to handle this later in chapter 7. The last two terms are contributions from anti-particles and particles respectively.

4.6 The pressure and energy density at zero temperature

The general expression obtained for the grand potential (4.81) can be solved by writing the integrand as an infinite series and integrate term by term. However, at zero temperature assuming $\mu > 0$, the equation simplifies as

$$\ln\Big[1 + \mathrm{e}^{-\beta(E_{\boldsymbol{p}}+\mu)}\Big] = 0, \tag{4.82}$$

at $T = 1/\beta = 0$. If we now forget about the zero-point energy[3], the pressure becomes

$$P = 2\int \frac{\mathrm{d}^3p}{(2\pi)^3} T \ln\Big[1 + \mathrm{e}^{-\beta(E-\mu)}\Big]. \tag{4.83}$$

Here we skipped writing the subscript $\boldsymbol{p}$ for simplicity. The integral is spherically symmetric, so integrating over the angles gives a factor 4π:

$$P = \frac{8\pi}{(2\pi)^3}\int_0^\infty \mathrm{d}p\, p^2 T \ln\Big[1 + \mathrm{e}^{-\beta(E-\mu)}\Big] = \frac{1}{\pi^2}\int_0^\infty \mathrm{d}p\, p^2 T \ln\Big[1 + \mathrm{e}^{-\beta(E-\mu)}\Big]. \tag{4.84}$$

Integrating by parts, we find

$$\frac{1}{3\pi^2}\int_0^\infty \mathrm{d}p\, \frac{p^4}{\sqrt{p^2+m^2}}\,\frac{T\beta\mathrm{e}^{-\beta(E-\mu)}}{1+\mathrm{e}^{-\beta(E-\mu)}} = \frac{1}{3\pi^2}\int_0^\infty \mathrm{d}p\, \frac{p^4}{E}\,\frac{1}{\mathrm{e}^{-\beta(\mu-E)}+1}. \tag{4.85}$$

One of the possible representations of the Heaviside step function is

$$\theta(x-y) = \lim_{r\to\infty}\frac{1}{\mathrm{e}^{-r(x-y)}+1} = \begin{cases} 1, & \text{for}\, x > y \\ 0, & \text{for}\, x < y \end{cases}. \tag{4.86}$$

Hence, at zero temperature we may write

$$P = \frac{1}{3\pi^2}\int_0^\infty \mathrm{d}p\, \frac{p^4}{E}\theta(\mu - E) = \frac{1}{3\pi^2}\int_0^\infty \mathrm{d}p\, \frac{p^4}{E}\theta(\mu - \sqrt{p^2+m^2}) = \frac{1}{3\pi^2}\int_0^{\sqrt{\mu^2-m^2}} \mathrm{d}p\, \frac{p^4}{E}. \tag{4.87}$$

Changing integration variable from p to E yields

$$P = \frac{1}{3\pi^2}\int_m^\mu \mathrm{d}E\, (E^2-m^2)^{3/2}. \tag{4.88}$$

This integral can be computed through a series of substitutions and partial integrations with the result

$$\begin{aligned} P &= \frac{1}{3\pi^2}\,\frac{1}{8}\Big[E\sqrt{E^2-m^2}\Big(2E^2-5m^2\Big) + 3m^4\ln\Big(\sqrt{E^2-m^2}+E\Big)\Big]_m^\mu \\ &= \frac{1}{24\pi^2}\left[\mu\sqrt{\mu^2-m^2}\Big(2\mu^2-5m^2\Big) + 3m^4\ln\left(\frac{\sqrt{\mu^2-m^2}+\mu}{m}\right)\right]. \end{aligned} \tag{4.89}$$

Now we want to find the charge density $\rho = Q/V$. Using equation (4.50), we note that

$$\frac{\partial Z}{\partial\mu} = \frac{\partial}{\partial\mu}\sum_i \mathrm{e}^{-\beta(E_i-\mu Q)} = Q\beta\sum_i \mathrm{e}^{-\beta(E_i-\mu Q)} = Q\beta Z, \tag{4.90}$$

which we can rewrite as

$$\rho = \frac{Q}{V} = \frac{1}{\beta VZ}\frac{\partial Z}{\partial\mu} = \frac{1}{\beta V}\frac{\partial}{\partial\mu}\ln Z = \frac{1}{\beta V}\frac{\partial}{\partial\mu}(\beta VP) = \frac{\partial P}{\partial\mu}, \tag{4.91}$$

[3] We will handle this term in chapter 7.

where we in the next to last step used equation (4.76). Now we can go back to (4.84) and find the charge density

$$\rho = \frac{\partial}{\partial\mu}\left\{\frac{1}{\pi^2}\int_0^\infty \mathrm{d}p\, p^2 T \ln\Big[1+\mathrm{e}^{-\beta(E-\mu)}\Big]\right\} = \frac{1}{\pi^2}\int_0^\infty \mathrm{d}p\, p^2\, \frac{1}{\mathrm{e}^{-\beta(E-\mu)}+1}. \tag{4.92}$$

At $T=0$ we can again use (4.86) to obtain the result

$$\rho = \frac{1}{\pi^2}\int_0^{\sqrt{\mu^2-m^2}} \mathrm{d}p\, p^2 = \frac{1}{3\pi^2}(\mu^2-m^2)^{3/2}. \tag{4.93}$$

Lastly, we find the energy density ϵ by using (4.73):

$$\epsilon = \frac{U}{V} = \mu\rho + \Omega = \mu\rho - P. \tag{4.94}$$

Inserting expressions for charge density and pressure we find

$$\begin{aligned}\epsilon &= \frac{\mu}{3\pi^2}(\mu^2-m^2)^{3/2} - \frac{1}{24\pi^2}\left[\mu\sqrt{\mu^2-m^2}(2\mu^2-5m^2) + 3m^4\ln\left(\frac{\sqrt{\mu^2-m^2}+\mu}{m}\right)\right]\\ &= \frac{1}{24\pi^2}\left[\mu\sqrt{\mu^2-m^2}(6\mu^2-3m^2) + 3m^4\ln\left(\frac{\sqrt{\mu^2-m^2}+\mu}{m}\right)\right].\end{aligned} \tag{4.95}$$

It is more convenient to write these expressions in terms of the Fermi momentum $p_{\mathrm{F}} = \sqrt{E_{\mathrm{F}} - m^2} = \sqrt{\mu^2-m^2}$, which is the momentum-eigenstate with highest momentum eigenvalue occupied by particles at zero temperature. Here we use that the Fermi energy E_{F} is equal to the chemical potential μ at zero temperature. Then we obtain the following relations:

$$P = \frac{1}{24\pi^2}\left[\sqrt{p_{\mathrm{F}}^2+m^2}(2p_{\mathrm{F}}^3 - 3m^2 p_{\mathrm{F}}) + 3m^4\ln\left(\frac{p_{\mathrm{F}}+\sqrt{p_{\mathrm{F}}^2+m^2}}{m}\right)\right], \tag{4.96}$$

$$\epsilon = \frac{1}{24\pi^2}\left[\sqrt{p_{\mathrm{F}}^2+m^2}(6p_{\mathrm{F}}^3 + 3m^2 p_{\mathrm{F}}) - 3m^4\ln\left(\frac{p_{\mathrm{F}}+\sqrt{p_{\mathrm{F}}^2+m^2}}{m}\right)\right], \tag{4.97}$$

$$\rho = \frac{p_{\mathrm{F}}^3}{3\pi^2}. \tag{4.98}$$

Using the series expansions

$$\sqrt{p_{\mathrm{F}}^2+m^2} = m + \frac{p_{\mathrm{F}}^2}{2m} - \frac{p_{\mathrm{F}}^4}{8m^3} + \mathcal{O}\left(\frac{p_{\mathrm{F}}^6}{m^5}\right), \tag{4.99}$$

and

$$\ln\left(\frac{p_{\mathrm{F}}+\sqrt{p_{\mathrm{F}}^2+m^2}}{m}\right) = \frac{p_{\mathrm{F}}}{m} - \frac{p_{\mathrm{F}}^3}{6m^3} + \frac{3p_{\mathrm{F}}^5}{40m^5} + \mathcal{O}\left(\frac{p_{\mathrm{F}}^7}{m^7}\right), \tag{4.100}$$

we find that in the non-relativistic limit $p_{\mathrm{F}} \ll m$ the pressure is to leading order in p_{F} given by

$$\begin{aligned}P_{\mathrm{NR}} &\approx \frac{1}{24\pi^2}\left[\left(m+\frac{p_{\mathrm{F}}^2}{2m}-\frac{p_{\mathrm{F}}^4}{8m^3}\right)\left(2p_{\mathrm{F}}^3 - 3m^2 p_{\mathrm{F}}\right) + 3m^4\left(\frac{p_{\mathrm{F}}}{m} - \frac{p_{\mathrm{F}}^3}{6m^3} + \frac{3p_{\mathrm{F}}^5}{40m^5}\right)\right]\\ &= \frac{1}{24\pi^2}\left[2p_{\mathrm{F}}^3 m - 3p_{\mathrm{F}} m^3 + \frac{p_{\mathrm{F}}^5}{m} - \frac{3p_{\mathrm{F}}^4}{2} - \frac{p_{\mathrm{F}}^7}{4m} + 3m^3 p_{\mathrm{F}} - \frac{mp_{\mathrm{F}}^3}{2} + \frac{9p_{\mathrm{F}}^5}{40m}\right]\\ &\approx \frac{p_{\mathrm{F}}^5}{15\pi^2 m}.\end{aligned} \tag{4.101}$$

From (4.98) we find the Fermi momentum

$$p_{\mathrm{F}} = \sqrt[3]{3\pi^2\rho}, \tag{4.102}$$

which inserted into (4.101) results in

$$P_{\mathrm{NR}} = \frac{(3\pi^2\rho)^{5/3}}{15\pi^2 m}. \tag{4.103}$$

In the same manner, the energy density in the $p_{\mathrm{F}} \ll m$ limit is to the fifth order in p_{F}

$$\begin{aligned}
\epsilon_{\mathrm{NR}} &\approx \frac{1}{24\pi^2}\left[\left(m + \frac{p_{\mathrm{F}}^2}{2m} - \frac{p_{\mathrm{F}}^4}{8m^3}\right)\left(6p_{\mathrm{F}}^3 + 3m^2 p_{\mathrm{F}}\right) - 3m^4\left(\frac{p_{\mathrm{F}}}{m} - \frac{p_{\mathrm{F}}^3}{6m^3} + \frac{3p_{\mathrm{F}}^5}{40m^5}\right)\right] \\
&= \frac{1}{24\pi^2}\left[6mp_{\mathrm{F}}^3 + 3m^3 p_{\mathrm{F}} + \frac{3p_{\mathrm{F}}^5}{m} + \frac{3mp_{\mathrm{F}}^3}{2} - \frac{3p_{\mathrm{F}}^7}{4m^3} - \frac{3p_{\mathrm{F}}^5}{8m} - 3m^3 p_{\mathrm{F}} + \frac{mp_{\mathrm{F}}^3}{2} - \frac{9p_{\mathrm{F}}^5}{40m}\right] \\
&\approx \frac{1}{24\pi^2}\left[8mp_{\mathrm{F}}^3 + \frac{12p_{\mathrm{F}}^5}{5m}\right] \\
&= \rho m + \frac{(3\pi^2\rho)^{5/3}}{10m\pi^2}.
\end{aligned} \tag{4.104}$$

where the first term represents the energy density from the rest mass of the conserved charge Q [26, p. 153]. To leading order, we combine (4.101) and (4.104) to form the relation

$$\epsilon_{\mathrm{NR}} = \rho m = \frac{m}{3\pi^2}(15\pi^2 m)^{3/5} P_{\mathrm{NR}}^{3/5}. \tag{4.105}$$

In the same manner we find that in the ultra-relativistic limit $p_{\mathrm{F}} \gg m$

$$P_{\mathrm{UR}} \approx \frac{1}{24\pi^2}\left[2p_{\mathrm{F}}^4 - 3m^2 p_{\mathrm{F}}^2 + 3m^4 \ln\left(\frac{2p_{\mathrm{F}}}{m}\right)\right] \approx \frac{p_{\mathrm{F}}^4}{12\pi^2} = \frac{(3\pi^2\rho)^{4/3}}{12\pi^2}, \tag{4.106}$$

and

$$\epsilon_{\mathrm{UR}} \approx \frac{1}{24\pi^2}\left[6p_{\mathrm{F}}^4 + 3m^2 m p_{\mathrm{F}}^2 - 3m^4 \ln\left(\frac{2p_{\mathrm{F}}}{m}\right)\right] \approx \frac{p_{\mathrm{F}}^4}{4\pi^2} = \frac{(3\pi^2\rho)^{4/3}}{4\pi^2}, \tag{4.107}$$

which gives

$$\epsilon_{\mathrm{UR}} = 3P_{\mathrm{UR}}. \tag{4.108}$$

We have now found a simple model for the EoS of a cold spherically symmetric neutron star. Unfortunately, we cannot measure the pressure inside a neutron star and so we are not able to check whether or not this model is good. However, by using the EoS to solve the equilibrium equations (3.37) and (3.42) we can predict what radiuses and masses stars with such an EoS would have a stable equilibrium. These are quantities we can measure, at least to some extent, and we are in particular interested in finding a maximum mass, which is what we are going to do in the next section.

4.7 Numerical solution to the structure equations

The solutions to the structure equations presented here are produced by the Python script given in Appendix G.1, where a simple fourth-order Runge-Kutta routine with constant step size is used. In the calculations, we parameterise the mass and radius as functions of the central pressure P_c. Some of the Figures presented use central pressures over a range of 8 orders of magnitude, which is the reason why we solve for exponentially increasing central pressures. This is to make sure that there are roughly equally many data points everywhere in the plot. All calculations in this section are done in units where $\hbar \neq c \neq 1$.

This section uses three main code parts: One for the non-relativistic case, one for the ultra-relativistic case, and one for arbitrary relativity. There is also some code that calculates $\mathrm{d}P/\mathrm{d}\epsilon$, which will be of importance in the stability analysis later.

For the non-relativistic case, the code runs a loop over a range of central pressures where it uses a constant time step Runge-Kutta routine at each iteration to solve (3.38) and (3.42). The Runge-Kutta solver starts with the initial condition $r = 0$ and $M(r = 0) = 0$ and increases the radius until the pressure becomes negative. The point where $P(r)$ changes sign gives the star's radius and mass with the given central pressure. Because of some stability issues, for a few central pressures, $P(r)$ never reaches zero. This is a purely numerical feature, and is solved by escaping the loop if the mass is equal at two iterations.

We will later find that in the ultra-relativistic limit the pressure never becomes negative (or zero), and thus the radius becomes infinite. This is *not* a stability issue; it happens for all central pressures and choice of step length. Therefore, this program only solves up to some given r in the Runge-Kutta solver, and plots the behavior for a given central pressure.

The program for arbitrary relativity is similar to the one for the non-relativistic case. However, since there is no explicit connection between the pressure P and the energy density ϵ, we must use a root solver to express the Fermi momentum p_{F} for a given pressure. The Fermi momentum is then inserted into the energy density so that $\mathrm{d}P/\mathrm{d}r$ can be calculated in each step in the Runge-Kutta solver.

When doing computations on a computer, it is always favorable to work with numbers that are roughly of the same size. If not, numerically stability can become a problem. Therefore we introduce a scaling factor ϵ_0 with units energy density so that

$$\epsilon = \epsilon_0 \bar{\epsilon}, \tag{4.109}$$

$$P = \epsilon_0 \overline{P}, \tag{4.110}$$

where $\bar{\epsilon}$ and $\overline{P}$ are dimensionless numbers. Note that since ϵ_0 is a scaling factor, it is a free parameter we can choose. In the same manner, it is also convenient to write the mass

$$M = M_\odot \overline{M}, \tag{4.111}$$

where $M_\odot \approx 1.988 \cdot 10^{30}\,\text{kg}$ denotes the solar mass. In units where $c \neq \hbar \neq 1$ the structure equations becomes

$$\frac{\mathrm{d}\overline{M}}{\mathrm{d}r} = \frac{4\pi\epsilon_0\bar{\epsilon}r^2}{M_\odot c^2}, \tag{4.112}$$

$$\frac{\mathrm{d}\overline{P}_{\text{New}}}{\mathrm{d}r} = -\frac{GM_\odot\overline{M}\bar{\epsilon}}{c^2r^2}, \tag{4.113}$$

$$\frac{\mathrm{d}\overline{P}_{\text{TOV}}}{\mathrm{d}r} = -\frac{GM_\odot\overline{M}\bar{\epsilon}}{c^2r^2}\left[1 + \frac{\overline{P}_{\text{TOV}}}{\bar{\epsilon}}\right]\left[1 + \frac{4\pi\epsilon_0\overline{P}_{\text{TOV}}r^3}{M_\odot\overline{M}c^2}\right]\left[1 - \frac{2GM_\odot\overline{M}}{rc^2}\right]^{-1}, \tag{4.114}$$

where $\overline{P}_{\text{New}}$ and $\overline{P}_{\text{TOV}}$ are the dimensionless pressure solution of the Newtonian– and the TOV–structure equation respectively.

4.7.1 The non-relativistic case

In the non-relativistic case, we previously found the relation between the energy density and the pressure

$$\epsilon_{\text{NR}} = \frac{mc^2}{3\pi^2\hbar^3}(15m\pi^2\hbar^3)^{3/5}P_{\text{NR}}^{3/5} \equiv K_{\text{NR}}P_{\text{NR}}^{3/5}, \tag{4.115}$$

where we have defined the constant

$$K_{\text{NR}} = \frac{mc^2}{3\pi^2\hbar^3}(15m\pi^2\hbar^3)^{3/5}. \tag{4.116}$$

Thus we may write

$$\bar{\epsilon}_{\text{NR}} = K_{\text{NR}}\epsilon_0^{-2/5}\bar{P}_{\text{NR}}^{3/5}. \tag{4.117}$$

For our simple model, we assume that the star consists only of neutrons, which means that the mass m is the neutron mass $m \approx 1.675 \cdot 10^{-27}$kg. Inserted values gives the constant K_{NR},

$$K_{\text{NR}} = 8.225 \cdot 10^{15}\,\text{kg}^{2/5}\text{km}^{-2/5}\,\text{s}^{-4/5}. \tag{4.118}$$

If we now introduce the constants

$$\begin{aligned} R_0 &= \frac{GM_\odot}{c^2}, \qquad \bar{K}_{\text{NR}} = K_{\text{NR}}\epsilon_0^{-2/5}, \\ \alpha &= R_0\bar{K}_{\text{NR}}, \qquad \beta = \frac{4\pi\epsilon_0\bar{K}_{\text{NR}}}{M_\odot c^2}, \end{aligned} \tag{4.119}$$

and substitute the non-relativistic EoS (4.115) into the structure equations (4.112)-(4.114), we find

$$\frac{\mathrm{d}\bar{M}}{\mathrm{d}r} = \beta r^2\bar{P}^{3/5} \tag{4.120}$$

$$\frac{\mathrm{d}\bar{P}_{\text{New}}}{\mathrm{d}r} = -\frac{R_0\bar{M}\bar{\epsilon}}{r^2}, \tag{4.121}$$

$$\frac{\mathrm{d}\bar{P}_{\text{TOV}}}{\mathrm{d}r} = -\frac{\alpha\bar{M}\bar{P}^{3/5}}{r^2}\left[1 + \frac{\bar{P}^{2/5}}{\bar{K}_{\text{NR}}}\right]\left[1 + \frac{\beta\bar{P}r^3}{\bar{K}_{\text{NR}}\bar{M}}\right]\left[1 - \frac{2R_0\bar{M}}{r}\right]^{-1}. \tag{4.122}$$

On a side note we recognize from equation (3.43) that R_0 is half the sun's Schwarzschild radius. Now we just have to determine the constants. Firstly, we find that

$$R_0 = 1.477\,\text{km}. \tag{4.123}$$

Secondly, we rearange the expression for α so that

$$\epsilon_0 = \left(\frac{\alpha}{R_0K_{\text{NR}}}\right)^{-5/2}. \tag{4.124}$$

For numerical stability we want the numerical values of the constants α, β and $\bar{K}_{\text{NR}}$ to be of about the same magnitude in the unit system chosen. We see that if we choose

$$\alpha = 1\,\text{km}, \tag{4.125}$$

then

$$\epsilon_0 = 1.6266 \cdot 10^{40}\,\text{kg/km}\,\text{s}^2, \tag{4.126}$$

which leads to

$$\bar{K}_{\text{NR}} = 0.6770\,\text{kg/km}\,\text{s}^2, \tag{4.127}$$

and

$$\beta = 0.7744\,\text{km}^{-3}. \tag{4.128}$$

Using these parameters, we obtain the mass-radius relation shown in Figure 4.1. Just as we saw for the constant density EoS in the previous chapter, Newtonian gravity still has no upper mass limit. For Einstein gravity however, there is a peak at 0.96 solar masses with radius 7.91 km. The central pressures considered range from $10^{-6} < \bar{P}_{\text{c}} < 10^4$ with increasing central pressures along the curve starting out at infinite radius. We also note that as the central pressure decreases, the two graphs gets closer and closer, telling us that relativistic effects vanish for small central pressures. This is as expected, since lower central pressures gives smaller Fermi energies, and thus are less relativistic.

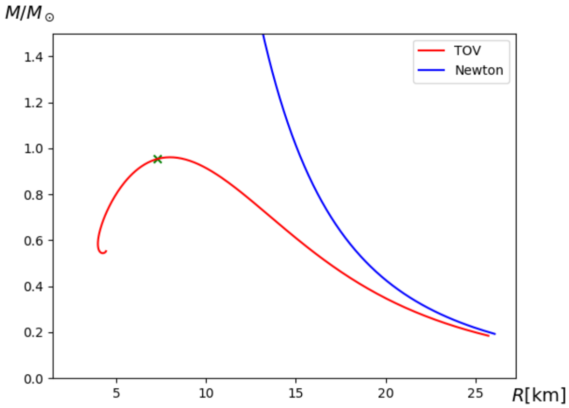


Figure 4.1: Mass-radius relation for a non-relativistic, cold ideal Fermi-gas using the structure equations for Newtonian gravity (Newton) and Einstein gravity (TOV). Moving along the red curve from infinite radius, the green cross marks the point where the speed of sound surpasses the speed of light. This is discussed in detail in section 4.7.4

4.7.2 The ultra-relativistic case

In the ultra-relativistic case we have the relation

$$\bar{\epsilon} = 3\overline{P}. \tag{4.129}$$

The structure equations then becomes

$$\frac{\mathrm{d}\overline{M}}{\mathrm{d}r} = \frac{12\pi\epsilon_0\overline{P}r^2}{M_\odot c^2} = \beta\overline{P}r^2, \tag{4.130}$$

$$\frac{\mathrm{d}\overline{P}}{\mathrm{d}r} = -\frac{3R_0\overline{M}\overline{P}}{r^2}, \qquad \text{(Newtonian)} \tag{4.131}$$

$$\frac{\mathrm{d}\overline{P}}{\mathrm{d}r} = -\frac{4R_0\overline{M}\overline{P}}{r^2}\left[1+\frac{\beta\overline{P}r^3}{3\overline{M}}\right]\left[1-\frac{2R_0\overline{M}}{r}\right]^{-1}, \qquad \text{(TOV)} \tag{4.132}$$

where we have defined

$$\beta = \frac{12\pi\epsilon_0}{M_\odot c^2}. \tag{4.133}$$

If we now choose the value of this constant to be

$$\beta = 1\,\mathrm{km}^{-3}, \tag{4.134}$$

we find the scaling factor

$$\epsilon_0 = \frac{M_\odot c^2\beta}{12\pi} = 4.740\cdot 10^{48}\,\mathrm{kg/km\,s^2}. \tag{4.135}$$

Trying to run the program used in the non-relativistic case, one soon finds that the maximum number of iterations $nMax$ in the Runge Kutta routine is reached even for very small values of P_c. In fact, the star's radius becomes infinite for all central pressures as the pressure only approaches zero asymptotically. The problem here is that we have assumed that the neutrons are ultra relativistic throughout the star, which is a self contradicting statement. Since the pressure by definition is zero at the surface, equation (4.96) yields that the Fermi energy should be zero. This means that when integrating from the center of the star and out, the neutrons at some point becomes non-relativistic. Thus it is clear that no such star could exist.

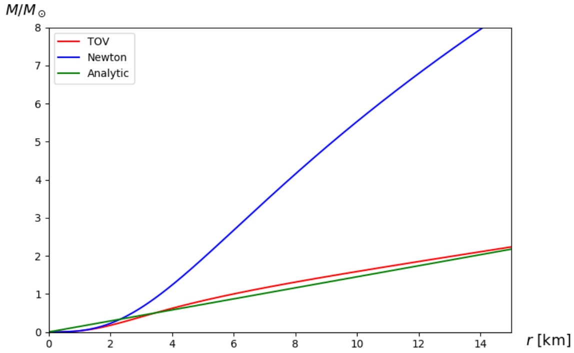


Figure 4.2: Accumulated mass M as a function of distance from the center r for a star consisting of a relativistic ideal gas of neutrons with central pressure $\overline{P}_c = 0.1$. The plots show the numerical and the analytic solution to the TOV equation as well as the solution to the Newtonian structure equation.

Figure 4.2 and 4.3 shows the behaviour of the mass and the pressure as functions of the distance from the star's center respectively. We note that the pressure decreases faster with relativistic corrections and that the mass increases much slower using the TOV-equation than the Newtonian equation, as expected, since we already have seen that relativity increases the strength of gravity.

It should be mentioned that in the ultra-relativistic case it is also possible to find an analytic solution to the TOV equation. We see this by making the anzats

$$\overline{P}(r) = Kr^n, \tag{4.136}$$

where K is some constant and n is an integer. If we now insert this expression into (4.130) we find

$$M(r) = \frac{1}{3+n}\beta K r^{n+3}. \tag{4.137}$$

While integrating (4.130) to obtain (4.137), we have assumed that $n \neq -3$. This is reasonable as we see from (4.132) that the pressure never falls of faster than r^{-2}. Inserting (4.137) into (4.132) then yields

$$\begin{aligned}\frac{\mathrm{d}\overline{P}}{\mathrm{d}r} = nKr^{n-1} &= -\frac{4R_0\beta K r^{n+3} K r^n}{r^2(n+3)}\left[1 + \frac{\beta K r^n r^3 (n+3)}{3\beta K r^{n+3}}\right]\left[1 - \frac{2R_0\beta K r^{n+2}}{n+3}\right]^{-1} \\ &= -\frac{4R_0\beta K^2 r^{2n+3}}{r^2}\left[\frac{6+n}{3(3+n)}\right]\left[1 - \frac{2R_0\beta K r^{n+2}}{n+3}\right]^{-1}. \end{aligned} \tag{4.138}$$

This should hold for all r, so we have to choose n so that the r-dependence vanishes. We see that for $n = -2$ we obtain

$$\frac{-2K}{r^3} = -\frac{16R_0\beta K^2}{3r^3}\left[1 - 2R_0\beta K\right]^{-1}, \tag{4.139}$$

which solved for K gives

$$K = \frac{3}{14\beta R_0}. \tag{4.140}$$

The analytic solution then becomes

$$\overline{P} = \frac{3}{14\beta R_0 r^2} = \frac{c^4}{56\epsilon_0 \pi G r^2}. \tag{4.141}$$

This solution does not only predict infinite mass and radius as seen in the numerical solutions, but also requires a divergent central pressure. For that reason I find this solution to be even worse than

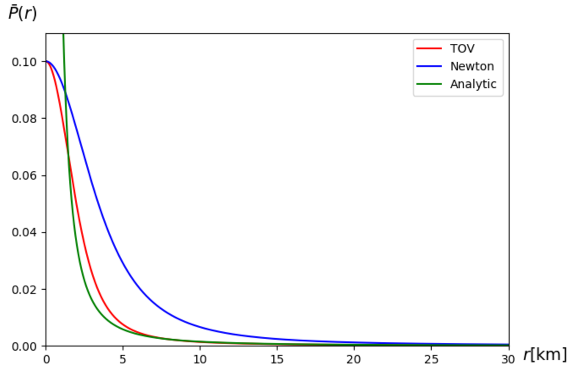


Figure 4.3: Dimensionless pressure $\overline{P}$ as a function of distance from the center r for a ultra-relativistic ideal gas of neutrons. The central pressure is $\overline{P} = 0.1$ for the solution to the TOV equation and the Newtonian structure equation, while it is infinite for the analytic solution due to reasons discussed in the text.

the numerical ones. Even so, for large r, Figure 4.2 and 4.3 shows that the analytic and numerical solution for the mass and pressure follow each other to some degree. This has to do with the fact that the TOV-equation predicts $\overline{P} \propto 1/r^2$ for large r. Anyway, one might argue that it does not really make any sense to identify which of these solutions that is the best, as they all build on the same faulty assumption that the star is throughout ultra-relativistic.

4.7.3 Solutions for arbitrary relativity

In the general case, we must solve the problem using the original equations (4.96) and (4.97) for the pressure and energy density. To do so, we start our integration at the center with $P = P_\mathrm{c}$ and use an builtin root finder to retrieve the Fermi momentum from the equation

$$P - P(p_\mathrm{F}) = 0. \tag{4.142}$$

Calculating the Fermi momentum at each step gives us the energy density ϵ which in turn is inserted in the Runge-Kutta routine.

As before, we want to make the variables dimensionless. We therefore introduce the new constants

$$E_0 = mc^2, \qquad \gamma = \frac{E_0^4}{\hbar^3 c^3 \epsilon_0}, \tag{4.143}$$

and define

$$\beta = \frac{4\pi\epsilon_0}{M_\odot c^2}, \tag{4.144}$$

so that the structure equations yields

$$\frac{\mathrm{d}\overline{M}}{\mathrm{d}r} = \beta \overline{P} r^2, \tag{4.145}$$

$$\frac{\mathrm{d}\overline{P}}{\mathrm{d}r} = -\frac{R_0 \overline{\epsilon_0} \overline{M}}{r^2}\left[1 + \frac{\overline{P}}{\bar{\epsilon}}\right]\left[1 + \frac{\beta \overline{P} r^3}{\overline{M}}\right]\left[1 - \frac{2R_0\overline{M}}{r}\right]^{-1}. \tag{4.146}$$

Then we define the dimensionless Fermi momentum

$$p_\mathrm{F} = \frac{E_0 \overline{p_\mathrm{F}}}{c}, \tag{4.147}$$

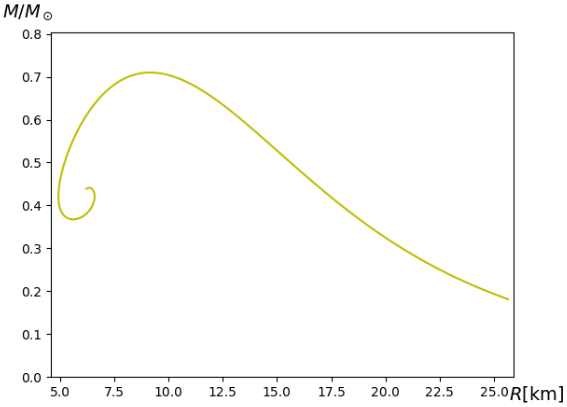


Figure 4.4: Mass-radius relation for a star consisting of an ideal gas of neutrons for arbitrary relativity. Dimensionless central pressures are chosen between 10^{-6} and 10^{4} with increasing central pressures along the curve starting out at infinite radius.

to find the dimensionless pressure and energy density

$$\overline{P} = \frac{\gamma}{24\pi^2}\left[\sqrt{\bar{p}_{\mathrm{F}}^2+1}\left(2\bar{p}_{\mathrm{F}}^3 - 3\bar{p}_{\mathrm{F}}\right) + 3\ln\left(\bar{p}_{\mathrm{F}} + \sqrt{\bar{p}_{\mathrm{F}}^2+1}\right)\right], \tag{4.148}$$

$$\bar{\epsilon} = \frac{\gamma}{24\pi^2}\left[\sqrt{\bar{p}_{\mathrm{F}}^2+1}\left(6\bar{p}_{\mathrm{F}}^3 + 3\bar{p}_{\mathrm{F}}\right) - 3\ln\left(\bar{p}_{\mathrm{F}} + \sqrt{\bar{p}_{\mathrm{F}}^2+1}\right)\right]. \tag{4.149}$$

If we for instance now choose

$$\gamma = 1\,\mathrm{kg/km\,s^2}, \tag{4.150}$$

we find the scaling factor

$$\epsilon_0 = 1.625\cdot 10^{40}\,\mathrm{kg\,/km\,s^2}. \tag{4.151}$$

and can compute the last constant

$$\beta = 1.1426\,\mathrm{km}^{-3}. \tag{4.152}$$

Using the program in the Appendix G.1, we obtain the curve given by Figure 4.4. Our computation predicts a maximum mass of 0.71 solar masses with radius 9.16km. To compare, Oppenheimer and Volkoff's original result from 1939 obtained without the use of a computer was $M = 0.71M_\odot$ and $R = 9.5$km [19].

It is also interesting to plot the non-relativistic and arbitrary relativistic solutions together to see some connections. From Figure 4.5 we find that the solutions to the non-relativistic and arbitrary relativistic TOV-equation are within an relative error of 1% at around $R = 23\,\mathrm{km}$. This corresponds to a central pressure of about $\overline{P}_{\mathrm{c}} \approx 10^{-5}$. The solution to Newton's equation needs an even smaller central pressure before we can recon that it is a good approximation.

Looking closer, we see that the masses, up to the maximum mass (we will later later see that beyond this limit the solutions are unstable), in general are larger the less relativity we account for. This is as mentioned in the previous chapter due to the fact that relativity works to amplify gravity, and hence prevents the stars from growing big.

4.7.4 Stability analysis

The mass-radius relations obtained shows the equilibrium configurations for pure neutron stars. However, we do not yet know if a star on the $M(R)$ curve is stable or not. For one, a stable star must satisfy

$$\frac{\mathrm{d}P}{\mathrm{d}\epsilon} \geq 0, \tag{4.153}$$

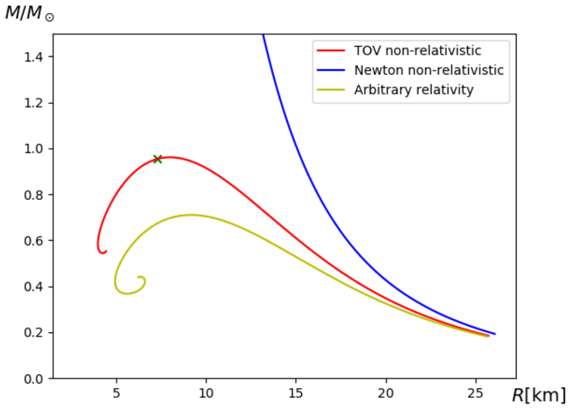


Figure 4.5: All mass-radius relations obtained earlier collected in a sample plot. For more details, see Figure 4.1 and 4.4.

everywhere. This becomes clear if we imagine a region where this was not true. If we increase the pressure, the energy density in this region, and hence also the gravitational pressure, decreases. There is nothing that stops an everlasting expansion since the pressure increases more and more as the star becomes less dense. In the same manner, if we increase the energy density, the pressure decreases while the gravitational forces becomes stronger, leading to the collapse of that region. The condition (4.153) is often called the ”microscopic stability” condition [28, p. 258]. Causality also demands that the speed of sound never exceeds the speed of light. The speed of sound v_s in a medium is given by the relativistic Euler-equation

$$v_\mathrm{s}^2 = c^2 \frac{\mathrm{d}P}{\mathrm{d}\epsilon}, \tag{4.154}$$

and so our stability criterion becomes

$$0 \leq \frac{\mathrm{d}P}{\mathrm{d}\epsilon} \leq 1. \tag{4.155}$$

In the non-relativistic case we use (4.115) to obtain

$$\frac{\mathrm{d}\overline{P}}{\mathrm{d}\epsilon} = \frac{5}{3}\overline{K}_{\mathrm{NR}}^{-5/3}\bar{\epsilon}^{2/3} = \frac{5}{3}\overline{K}_{\mathrm{NR}}^{-3/5}\overline{P}^{2/5}. \tag{4.156}$$

We see that since P is strictly positive, we only need to handle the causality restriction. The pressure is always largest at the center, so we may rewrite (4.155) as

$$P \leq P_\mathrm{c} \leq \left(\frac{3}{5}\right)^{5/2} \overline{K}_{\mathrm{NR}}^{3/2} \approx 0.0113. \tag{4.157}$$

The mass and radius of a star with this central pressure is indicated with a green cross in Figure 4.1and 4.5. There cannot be any stable stars further along the direction of increasing central pressure on the curve (the stars in the ”spiral” are unstable).

For arbitrary relativity, we must find $\mathrm{d}P/\mathrm{d}\epsilon$ numerically. The program used for this is shown in Appendix G.1. We find that when the exact expressions for P and ϵ are used, there is no problem with causality for any Fermi energy. As calculated before, we see from Figure 4.6 that the non-relativistic limit has a steep relation between the energy density and the pressure. We also see the ultra-relativistic limit where the derivative approaches $\frac{1}{3}$.

Let us now consider a star on the curve in Figure 4.4 where

$$\frac{\mathrm{d}\overline{M}}{\mathrm{d}P_\mathrm{c}} < 0. \tag{4.158}$$

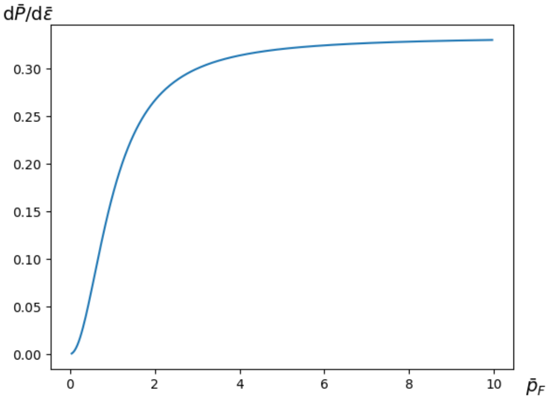


Figure 4.6: The derivative of the pressure with respect to the energy density as function of the dimensionless Fermi momentum for a star with central pressure $\overline{P}_\mathrm{c} = 0.1$.

If we compress such a star by a small amount, the central pressure $\overline{P}_\mathrm{c}$ increases while the mass remains constant. To get back to stable equilibrium the mass must decrease, which in turn means that the gravitational pressure is higher than the degeneracy pressure. Thus, the star collapses. If in some other scenario the star expands, the pressure decreases, while the mass is constant. Equilibrium is then achieved by increasing the mass, meaning that the gravitational forces are smaller than the degeneracy pressure. The star explodes. We conclude that a stable star cannot satisfy (4.158).

On the other hand, in the opposite case where

$$\frac{\mathrm{d}\overline{M}}{\mathrm{d}\overline{P}_\mathrm{c}} > 0, \tag{4.159}$$

a small compression still increases the pressure, but the mass must increase to return to equilibrium. The degeneracy pressure is larger than the gravitational pull, and the star will return to equilibrium. The opposite happens when the star expands. Thus a stable star satisfies (4.159).

In the non-relativistic case, this means that only stars to the right of the global maximum in Figure 4.1 can exist. For arbitrary relativity, we still have some regions in the spiral satisfying (4.159) that could be stable. However, if we perturb the surface of the star and look at the eigenmodes of the oscillations, we could find a so called Sturm-Liouville eigenvalue problem where stability is equivalent with the system only having real eigenvalues [29]. From this, it is possible to derive a few rules of thumb, assuming that there does not occur a phase-transition inside the star: [30]

- A star is stable if all of its eigenmodes are stable.
- Exactly one eigenmode changes stability where the curve $M(R)$ has an extremum.
- If the $M(R)$ curve has a (counter)clockwise direction at an extremum, the eigenmode that changes stability becomes (un)stable.

From this we find that only the stars on the $M(R)$ curve that are to the right of the global maximum in Figure 4.4 are stable.

4.8 Summary

In this chapter we have developed equations for the pressure and energy density for an ideal Fermi gas. We then solved these equations in the zero-temperature limit, before combining them with the structure equations to predict the mass-radius relation. As a result, we obtained a maximum mass of

$0.71M_{\odot}$, which is consistent with the first calculations done by Oppenheimer and Volkoff in 1939. We keep in mind that even though the EoS found is fairly simple, this result still is of the same order of magnitude as the most massive neutron star measured today which is the pulsar PSR J0348+0432 with its approximately two solar masses [7].

Chapter 5

The σ–ω model

So far, we have considered a neutron star consisting of an ideal, cold Fermi gas of neutrons. As a natural next step, we would like to include interactions between the particles. One should keep in mind that a realistic model of a neutron star would not only include neutrons. For one, we expect the presence of protons and electrons in the gas. This becomes clear if we consider a free neutron. Free neutrons are unstable with a half life of about ten minutes and will eventually go through beta decay to produce a proton and an electron. However, even if a neutron star mostly consisted of free neutrons, it could still be stable. This is an artifact of the Pauli principle: The electrons produced in beta decay become degenerate, and after enough electrons are created, the energy needed to emit an electron through beta decay is so high that it is not energetically favourable. Hence, the number of protons and electrons must be small compared to the number of neutrons.

Neutrons and protons are baryons, which means that they are particles built up of three quarks. These particles interact through the strong force which is mediated by gluons, described by the theory of quantum chromodynamics (QCD). We will not discuss QCD in this master thesis, but instead we consider a somewhat simpler model, known as the σ-ω model. In this framework, first used by Teller [31], Duerr [32] and Walecka [33], we assume that the strong force is mediated by two mesons, with spin zero and one, respectively.[1] This is in field theory represented by a scalar field σ and a vector field ω_μ. Note that this means that the theory should break down at distances where the mesons no longer behave as point particles, which is at around the femtometre scale, where the quarks manifest themselves [25, p. 221]. Also, we assume that the neutron and the proton both are two different states of one particle which we will call the nucleon. These two particle states are connected through a rotation in isospin space. Further we approximate by assuming that the star consists of static, uniform matter in its ground state, and replace the meson fields by their ground state expectation values [6, p. 168]. This approximation is known as the relativistic mean-field approximation (RMF).

In this chapter we use the σ-ω model to obtain an equation of state for the nucleons in a neutron star. This equation will then be used to find an upper limit for the maximum mass, and we will compare these results to the ones obtained earlier.

5.1 The free Lagrangian

The first step in any field theory description is to construct a suitable Lagrangian for the system. Before we add the interactions, we want to find the total free Lagrangian for the particles considered. Being a massive spin-less field, σ is described without interactions by the Klein-Gordon Lagrangian

$$\mathscr{L}_\sigma = \frac{1}{2}(\partial_\mu\sigma)(\partial^\mu\sigma) - \frac{1}{2}m_\sigma^2\sigma^2. \tag{5.1}$$

[1] Mesons are particles consisting of one quark and one anti-quark

Using the Euler-Lagrange equations (E.6), we obtain the equation of motion

$$\frac{\partial \mathscr{L}}{\partial \sigma} - \partial_\mu \frac{\partial \mathscr{L}}{\partial(\partial_\mu \sigma)} = -m_\sigma^2 \sigma - \partial_\mu \partial^\mu \sigma = -m_\sigma^2 \sigma - \Box \sigma = -(m_\sigma^2 + \Box\,)\sigma = 0, \tag{5.2}$$

where we have defined the d'Alembertian

$$\Box \equiv \partial_\mu \partial^\mu, \tag{5.3}$$

and used the relation

$$\frac{\partial}{\partial_\mu \sigma}(\partial_\mu \sigma)(\partial^\mu \sigma) = \partial^\mu \sigma + \partial_\mu \sigma \frac{\partial}{\partial_\mu \sigma}(\eta^{\mu\nu}\partial_\nu \sigma) = \partial^\mu \sigma + \eta^{\mu\nu}\partial_\mu \sigma \frac{\partial}{\partial_\mu \sigma}(\partial_\nu \sigma) = \partial^\mu \sigma + \partial^\nu \sigma \delta^\mu_\nu = 2\partial^\mu \sigma. \tag{5.4}$$

For the massive spin-one field ω_μ, we must have three degrees of freedom, one for each spin state. This means that there should be three independent plane-wave solutions to the equation of motion. Additionally, each of the spin states should alone satisfy the Klein-Gordon equation. We see that if we define

$$\omega_{\mu\nu} \equiv \partial_\mu \omega_\nu - \partial_\nu \omega_\mu, \tag{5.5}$$

and then choose the Proca Lagrangian

$$\begin{aligned}
\mathscr{L}_\omega &= -\frac{1}{4}\omega_{\mu\nu}\omega^{\mu\nu} + \frac{1}{2}m_\omega^2 \omega_\mu \omega^\mu \\
&\equiv -\frac{1}{4}(\partial_\mu \omega_\nu \partial^\mu \omega^\nu - \partial_\mu \omega_\nu \partial^\nu \omega^\mu - \partial_\nu \omega_\mu \partial^\mu \omega^\nu + \partial_\nu \omega_\mu \partial^\nu \omega^\mu) + \frac{1}{2}m_\omega^2 \omega_\mu \omega^\mu \\
&= -\frac{1}{2}(\partial_\mu \omega_\nu \partial^\mu \omega^\nu - \partial_\mu \omega_\nu \partial^\nu \omega^\mu) + \frac{1}{2}m_\omega^2 \omega_\mu \omega^\mu,
\end{aligned} \tag{5.6}$$

the equation of motion becomes

$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial \omega_\mu} - \partial_\mu \frac{\partial \mathscr{L}}{\partial(\partial_\mu \omega_\nu)} &= m_\omega^2 \omega^\mu + \frac{1}{2}\partial_\mu(2\partial^\mu \omega^\nu - 2\partial^\nu \omega^\mu) \\
&= m_\omega^2 \omega^\mu + \partial_\mu \partial^\mu \omega^\nu - \partial_\mu \partial^\nu \omega^\mu \\
&= (m_\omega^2 + \Box\,)\,\omega^\nu - \partial^\nu \partial_\mu \omega^\mu \\
&= 0.
\end{aligned} \tag{5.7}$$

Contracting this equation with ∂_ν we obtain

$$(m_\omega^2 + \Box\,)\,\partial_\nu \omega^\nu - \partial_\nu \partial^\nu \partial_\mu \omega^\mu = m_\omega^2 \partial_\mu \omega^\mu + \Box\,\partial_\nu \omega^\nu - \Box\,\partial_\mu \omega^\mu = m_\omega^2 \partial_\mu \omega = 0, \tag{5.8}$$

which implies that the field is divergenceless

$$\partial_\mu \omega^\mu = \partial^\mu \omega_\mu = 0, \tag{5.9}$$

since it is assumed to be massive. Then it follows that each component of the ω-field satisfies the Klein-Gordon equation as we required. Further, equation (5.9) imposes one constraint on the four components of the field, and so there are only three independent plane-wave solutions for ω_μ. The free Lagrangian proposed in (5.6) is satisfactory, and we proceed.

For the neutron and the proton, which are spin $\frac{1}{2}$-particles, we already have the free Lagrangian from (4.53):

$$\mathscr{L}_{\mathrm{np}} = \mathscr{L}_{\mathrm{n}} + \mathscr{L}_{\mathrm{p}} = \bar{\psi}_{\mathrm{n}}(\mathrm{i}\partial\!\!\!/ - m_{\mathrm{n}})\psi_{\mathrm{n}} + \bar{\psi}_{\mathrm{p}}(\mathrm{i}\partial\!\!\!/ - m_{\mathrm{p}})\psi_{\mathrm{p}}. \tag{5.10}$$

Assuming that the neutron and the proton masses are the same, and defining an eight-component spinor consisting of both the neutron and proton eigenstates

$$\psi = \begin{pmatrix} \psi_{\mathrm{n}} \\ \psi_{\mathrm{p}} \end{pmatrix}, \tag{5.11}$$

we may write

$$\mathscr{L}_{\text{np}} = \bar{\psi}(\mathrm{i}\partial\!\!\!/ - m)\psi. \tag{5.12}$$

Note that this is in some manner an abuse of notation. When we write for instance γ^μ, what we really mean is

$$I_2 \otimes \gamma^\mu = \begin{pmatrix} \gamma^\mu & 0 \\ 0 & \gamma^\mu \end{pmatrix}, \tag{5.13}$$

where I_n denotes the $n \times n$ identity matrix. Adding it all together, we arrive at the free Lagrangian

$$\mathscr{L}_{\text{free}} = \frac{1}{2}(\partial_\mu \sigma)(\partial^\mu \sigma) - \frac{1}{2}m_\sigma^2 \sigma^2 - \frac{1}{4}\omega_{\mu\nu}\omega^{\mu\nu} + \frac{1}{2}m_\omega^2 \omega_\mu \omega^\mu + \bar{\psi}(\mathrm{i}\partial\!\!\!/ - m)\psi. \tag{5.14}$$

5.2 The full Lagrangian

Now that we have the free Lagrangian, we add the interaction terms. In this model we neglect the mesonic interactions. Then the only remaining interactions are between each of the meson fields and the Dirac field ψ. Since the Lagrangian should be a Lorentz scalar, the scalar σ-field must be coupled to another scalar. The only possible choice is the baryon scalar density $\bar{\psi}\psi$ so that

$$\mathscr{L}_{\sigma_{\text{int}}} = \pm g_\sigma \sigma \bar{\psi}\psi, \tag{5.15}$$

where g_σ is some coupling constant. In the same manner, the ω_μ vector field must be contracted in some way, and the only Lorentz invariant choice is to couple it to the baryon four-current $\bar{\psi}\gamma^\mu\psi$ so that

$$\mathscr{L}_{\omega_{\text{int}}} = \pm g_\omega \omega_\mu \bar{\psi}\gamma^\mu\psi. \tag{5.16}$$

We will later find the signs of the interaction terms[2] by demanding that the expectation values of the meson fields are positive numbers. We can now complete the Lagrangian with the result

$$\mathscr{L} = \frac{1}{2}(\partial_\mu \sigma)(\partial^\mu \sigma) - \frac{1}{2}m_\sigma^2 \sigma^2 - \frac{1}{4}\omega_{\mu\nu}\omega^{\mu\nu} + \frac{1}{2}m_\omega^2 \omega_\mu \omega^\mu + \bar{\psi}(\mathrm{i}\partial\!\!\!/ - m)\psi \pm g_\sigma \sigma \bar{\psi}\psi \pm g_\omega \omega_\mu \bar{\psi}\gamma^\mu\psi. \tag{5.17}$$

5.3 The energy spectrum

Now that we have constructed a suitable Lagrangian, we are interested in the equations of motion. Again, using the Euler-Lagrange equations, we obtain for the scalar fields:

$$\sigma\text{-field:} \qquad (\Box + m_\sigma^2)\sigma = \pm g_\sigma \bar{\psi}\psi, \tag{5.18}$$

$$\omega\text{-field:} \qquad (\Box + m_\omega^2)\omega_\mu - \partial_\mu \partial^\nu \omega_\nu = \mp g_\omega \bar{\psi}\gamma_\mu\psi. \tag{5.19}$$

Remembering that $j^\mu = \bar{\psi}\gamma^\mu\psi$ is a conserved current due to Noether's theorem, which means that $\partial_\mu j^\mu = 0$, we can in the same way as in (5.8) contract the last equation with ∂^μ to find that with interactions, the condition

$$\partial^\mu \omega_\mu = 0, \tag{5.20}$$

still holds. Lastly, the equation of motion for the nucleon field is given by

$$(\mathrm{i}\partial\!\!\!/ - m)\psi \pm g_\sigma \sigma \psi \pm g_\omega \omega_\mu \gamma^\mu \psi = \left[\gamma^\mu(\mathrm{i}\partial_\mu \pm g_\omega \omega_\mu) - (m \mp g_\sigma \sigma)\right]\psi = 0. \tag{5.21}$$

Now we can use the RMF approximation to solve the equations (5.18), (5.19) and (5.21). We start out by splitting each field ϕ in a classical part $\langle\phi\rangle$ and a quantum part $\tilde{\phi}$ so that

$$\omega_\mu = \tilde{\omega}_\mu + \langle\omega_\mu\rangle, \qquad \sigma = \tilde{\sigma} + \langle\sigma\rangle, \qquad \psi = \tilde{\psi} + \langle\psi\rangle. \tag{5.22}$$

[2] Of course, we could just choose the plus or the minus sign, if we do not assume that the coupling constants are positive numbers.

In the case of the nucleon field, the vacuum expectation value is zero. We can see this by a symmetry argument. If we for instance consider the mass term for the fermion field, the transformation $\psi \rightarrow \langle\psi\rangle + \psi$ would result in

$$m^2\bar{\psi}\psi \rightarrow m^2\langle\bar{\psi}\rangle\langle\psi\rangle + m^2\langle\bar{\psi}\rangle\psi + m^2\bar{\psi}\langle\psi\rangle + m^2\bar{\psi}\psi. \tag{5.23}$$

The two terms in the middle are clearly not Lorentz invariant, thus $\langle\bar{\psi}\rangle$ and $\langle\psi\rangle$ must be zero. For this reason we just denote $\tilde{\psi} \equiv \psi$.

In the RMF approximation we assume that the fluctuations in the meson fields vanish. Also assuming rotational symmetry at each point, the ground state expectation values $\langle\sigma\rangle$ and $\langle\omega\rangle$ must be independent of the space-time coordinate x_μ. By the same argument, the spatial components of the mean ω-field must be zero and so the equations of motion becomes

$$\begin{aligned}
&\sigma\text{-field} && (\Box + m_\sigma^2)\langle\sigma\rangle = m_\sigma^2\langle\sigma\rangle = \pm g_\sigma\bar{\psi}\psi = \pm g_\sigma\langle\bar{\psi}\psi\rangle, && (5.24)\\
&\omega\text{-field} && (\Box + m_\omega^2)\langle\omega_0\rangle = m_\omega^2\langle\omega_0\rangle = \mp g_\omega\bar{\psi}\gamma_0\psi = \mp g_\omega\langle\bar{\psi}\gamma_0\psi\rangle, && (5.25)\\
&\psi\text{-field} && \Big[\gamma^\mu\big(\mathrm{i}\partial_\mu \pm g_\omega\langle\omega_0\rangle\big) - \big(m \mp g_\sigma\langle\sigma\rangle\big)\Big]\psi = 0. && (5.26)
\end{aligned}$$

Here we replaced the $\bar{\psi}\psi$ and $\bar{\psi}\gamma_0\psi$ with their ground state expectation values, since they both are equal to expressions only constituent of ground state expectation values. From (5.24) and (5.25) it is clear that the sign of the interaction term (5.15) must be positive, and the sign of (5.16) must be negative to obtain a positive expectation value for the meson fields. Looking at (5.26), we see that expression inside the brackets is independent of x_μ. Then we may Fourier transform the whole expression using

$$\psi(k_\mu) = \int \mathrm{d}^4x\, \psi(x_\mu)\mathrm{e}^{-\mathrm{i}k\cdot x}, \tag{5.27}$$

to find

$$\begin{aligned}
0 &= \int \mathrm{d}^4x \Big[\gamma^\mu\big(\mathrm{i}\partial_\mu - g_\omega\langle\omega_\mu\rangle\big) - \big(m - g_\sigma\langle\sigma\rangle\big)\Big]\psi(x_\mu)\mathrm{e}^{-\mathrm{i}k\cdot x}\\
&= \int \mathrm{d}^4x \Big[\gamma^\mu\big(k_\mu - g_\omega\langle\omega_\mu\rangle\big) - \big(m - g_\sigma\langle\sigma\rangle\big)\Big]\psi(x_\mu)\mathrm{e}^{-\mathrm{i}k\cdot x}\\
&= \Big[\gamma^\mu\big(k_\mu - g_\omega\langle\omega_\mu\rangle\big) - \big(m - g_\sigma\langle\sigma\rangle\big)\Big]\int \mathrm{d}^4x\, \psi(x_\mu)\mathrm{e}^{-\mathrm{i}k\cdot x}\\
&= \Big[\gamma^\mu\big(k_\mu - g_\omega\langle\omega_\mu\rangle\big) - \big(m - g_\sigma\langle\sigma\rangle\big)\Big]\psi(k_\mu).
\end{aligned} \tag{5.28}$$

If we now define the new field momenta

$$K_\mu \equiv k_\mu - g_\omega\langle\omega_\mu\rangle = k_\mu - g_\omega\langle\omega_0\rangle, \tag{5.29}$$

and the effective mass

$$m^\star \equiv m - g_\sigma\langle\sigma\rangle, \tag{5.30}$$

we obtain

$$\big(\not{K} - m^\star\big)\psi = 0. \tag{5.31}$$

We recognize this as the Dirac equation in the momentum representation of a fermion field with mass $m^\star$. Hence we may treat the nucleon field as a free Fermion field with shifted mass and momenta.

Now we can find the energy spectrum of the σ-ω model. Multiplying both sides of (5.31) with

$(\not{K} + m^\star)$ we find

$$\begin{aligned}(\not{K} + m^\star)(\not{K} - m^\star)\psi &= \big[\not{K}\not{K} - (m^\star)^2\big]\psi \\ &= \big[\gamma_\mu K^\mu \gamma_\nu K^\nu - (m^\star)^2\big]\psi \\ &= \big[K_\mu K_\nu \gamma^\mu \gamma^\nu - (m^\star)^2\big]\psi \\ &= \big[K_\mu K_\nu (2\eta^{\mu\nu} - \gamma^\nu\gamma^\mu) - (m^\star)^2\big]\psi \\ &= \big\{2K_\mu K^\mu - \big[\not{K}\not{K} - (m^\star)^2\big] - 2(m^\star)^2\big\}\psi \\ &= 2\big[K_\mu K^\mu - (m^\star)^2\big]\psi \\ &= 0, \end{aligned} \tag{5.32}$$

where we have used the anti-commutation relation (C.8) for the γ^μ-matrices. Remembering that $K_\mu K^\mu$ is just a scalar, we find

$$K_\mu K^\mu - (m^\star)^2 = K_0^2 - \boldsymbol{K}^2 - (m^\star)^2 = 0, \tag{5.33}$$

and the energy eigenvalue for the nucleons becomes

$$E \equiv K_0 = \sqrt{\boldsymbol{K}^2 + (m^\star)^2} = \sqrt{\boldsymbol{k}^2 + (m - g_\sigma\langle\sigma\rangle)^2}. \tag{5.34}$$

In the limit when the coupling constants g_σ and g_ω go to zero, we obtain the free field dispersion relation as we should. We also note that if $g_\sigma\langle\sigma\rangle \to m$, the effective mass of the field becomes zero.

5.4 The partition function

Before we can find the EoS of the σ–ω model, we need the partition function. Remembering that the transition amplitude and the partition function are connected by a Wick rotation, we transform the Lagrangian to Euclidean space:[3]

$$\begin{aligned}\mathscr{L} =& \frac{1}{2}(\partial_\mu\sigma)(\partial^\mu\sigma) + \frac{1}{2}m_\sigma^2\sigma^2 - \frac{1}{4}\omega_{\mu\nu}\omega^{\mu\nu} - \frac{1}{2}m_\omega^2\omega_\mu\omega^\mu + \bar\psi\big(\not\partial + m - g_\sigma\sigma + g_\omega\omega_0\gamma^0 - \mathrm{i}g_\omega\omega_i\gamma^i\big)\psi \\ =& \frac{1}{2}(\partial_\mu\sigma)^2 + \frac{1}{2}m_\sigma^2\sigma^2 - \frac{1}{4}\Big[(\partial_\mu\omega_\nu)^2 - 2(\partial_\mu\omega_\nu)(\partial_\nu\omega_\mu) + (\partial_\nu\omega_\mu)^2\Big] \\ &- \frac{1}{2}m_\omega^2\omega_\mu^2 + \bar\psi\big(\not\partial + m - g_\sigma\sigma + g_\omega\omega_0\gamma_0 - \mathrm{i}g_\omega\omega_i\gamma_i\big)\psi. \end{aligned} \tag{5.35}$$

In the last step we have used that in Euclidean space the covariant and contravariant version of a tensor are the same.[4] Including a chemical potential for the nucleons

$$\mathscr{L} \to \mathscr{L} - \mu\psi^\dagger\psi, \tag{5.36}$$

the action becomes

$$\begin{aligned} S = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \bigg\{ &\frac{1}{2}(\partial_\mu\sigma)^2 + \frac{1}{2}m_\sigma^2\sigma^2 - \frac{1}{4}\Big[(\partial_\mu\omega_\nu)^2 - 2(\partial_\mu\omega_\nu)(\partial_\nu\omega_\mu) + (\partial_\nu\omega_\mu)^2\Big] \\ &- \frac{1}{2}m_\omega^2\omega_\mu^2 + \bar\psi\big(\not\partial + m - g_\sigma\sigma + g_\omega\omega_0\gamma_0 - \mu\gamma_0 - \mathrm{i}g_\omega\omega_i\gamma_i\big)\psi\bigg\}. \end{aligned} \tag{5.37}$$

[3] Unless otherwise specified, we assume Euclidean space and corresponding γ-matrices throughout the chapter without bothering to denote the subscript E.

[4] Note that we are a bit naughty when we write $(\partial_\mu\sigma)(\partial_\mu\sigma) = (\partial_\mu\sigma)^2$. We still assume summation over all spacetime indices μ.

Expanding this expression around the ground state expectation values we find

$$\begin{aligned}S =&S_0 + \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \left[\Big(\frac{\delta S}{\delta\sigma}\Big)_0 \tilde{\sigma} + \Big(\frac{\delta S}{\delta\omega_\mu}\Big)_0 \tilde{\omega} + \Big(\frac{\delta S}{\delta\psi}\Big)_0 \psi + \Big(\frac{\delta S}{\delta\bar{\psi}}\Big)_0 \bar{\psi} \right] \\ &+ \frac{1}{2} \int_0^\beta \mathrm{d}\tau \int_0^\beta \mathrm{d}\tau' \int \mathrm{d}^3x \int \mathrm{d}^3x' \Bigg[\tilde{\sigma}\Big(\frac{\delta^2 S}{\delta\sigma\delta\sigma'}\Big)_0 \tilde{\sigma}' + \tilde{\omega}_\mu\Big(\frac{\delta^2 S}{\delta\omega_\mu\delta\omega'_\mu}\Big)_0 \tilde{\omega}'_\mu \\ &+ \tilde{\sigma}\Big(\frac{\delta^2 S}{\delta\sigma\delta\psi'}\Big)_0 \psi' + \tilde{\sigma}'\Big(\frac{\delta^2 S}{\delta\psi\delta\sigma'}\Big)_0 \psi + \tilde{\omega}_\mu\Big(\frac{\delta^2 S}{\delta\omega_\mu\delta\psi'}\Big)_0 \psi' + \tilde{\omega}'_\mu\Big(\frac{\delta^2 S}{\delta\psi\delta\omega'_\mu}\Big)_0 \psi \\ &+ \bar{\psi}'\Big(\frac{\delta^2 S}{\delta\sigma\delta\bar{\psi}'}\Big)_0 \tilde{\sigma} + \bar{\psi}\Big(\frac{\delta^2 S}{\delta\bar{\psi}\delta\sigma'}\Big)_0 \psi + \bar{\psi}'\Big(\frac{\delta^2 S}{\delta\omega_\mu\delta\bar{\psi}'}\Big)_0 \tilde{\omega}_\mu + \bar{\psi}\Big(\frac{\delta^2 S}{\delta\bar{\psi}\delta\omega'_\mu}\Big)_0 \tilde{\omega}'_\mu \\ &+ \tilde{\sigma}\Big(\frac{\delta^2 S}{\delta\sigma\delta\omega'_\mu}\Big)_0 \tilde{\omega}'_\mu + \tilde{\omega}_\mu\Big(\frac{\delta^2 S}{\delta\omega_\mu\delta\sigma'}\Big)_0 \tilde{\sigma}' + \bar{\psi}\Big(\frac{\delta^2 S}{\delta\bar{\psi}\delta\psi'}\Big)_0 \psi' + \bar{\psi}'\Big(\frac{\delta^2 S}{\delta\bar{\psi}'\delta\psi}\Big)_0 \psi \Bigg] + \ldots ,\end{aligned} \tag{5.38}$$

where we evaluate the ground state expectation values of the terms with a subscript zero. Inserting the expressions for the functional derivatives and S_0 up to second order in the expansion, we find

$$\begin{aligned}S = &\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \Bigg[\frac{1}{2} m_\sigma^2 \langle\sigma\rangle^2 - \frac{1}{2} m_\omega^2 \langle\omega_0\rangle^2 + \big(m_\sigma^2\langle\sigma\rangle - g_\sigma\langle\bar{\psi}\psi\rangle\big)\tilde{\sigma} - \big(m_\omega^2\langle\omega_0\rangle - g_\omega\langle\psi^\dagger\psi\rangle\big)\tilde{\omega}_\mu \\ &+ \tilde{\sigma}\big(\Box + m_\sigma^2\big)\tilde{\sigma} - \tilde{\omega}_\mu\big(\Box + m_\omega^2\big)\tilde{\omega}_\mu + \big(m_\sigma^2\langle\sigma\rangle - g_\sigma\langle\bar{\psi}\psi\rangle - m_\omega^2\langle\omega_0\rangle + g_\omega\langle\psi^\dagger\psi\rangle\big)\tilde{\sigma}\tilde{\omega}_\mu \\ &+ \bar{\psi}\big(\partial\!\!\!/ + m^\star - \mu^\star\gamma_0\big)\psi \Bigg],\end{aligned} \tag{5.39}$$

where we have introduced the effective chemical potential

$$\mu^\star = \mu - g_\omega\langle\omega_0\rangle. \tag{5.40}$$

Using the equations of motion (5.24) and (5.25), the coefficients in front of the terms linear in $\tilde{\sigma}$ and $\tilde{\omega}$, as well as the one in front of the cross term $\tilde{\sigma}\tilde{\omega}_\mu$ vanish. Then we may write

$$S = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \left[\frac{1}{2} m_\sigma^2 \langle\sigma\rangle^2 - \frac{1}{2} m_\omega^2 \langle\omega_0\rangle^2 + \tilde{\sigma}\big(\Box + m_\sigma^2\big)\tilde{\sigma} - \tilde{\omega}_\mu\big(\Box + m_\omega^2\big)\tilde{\omega}_\mu + \bar{\psi}\big(\partial\!\!\!/ + m^\star - \mu^\star\gamma_0\big)\psi \right]. \tag{5.41}$$

Since we have assumed that the matter is static, we have that the integral over the mean fields $\langle\sigma\rangle$ and $\langle\omega_0\rangle$ only yields a factor of βV. To solve the remaining part of the integral, we expand the nucleon fields in frequency momentum space using (4.62) and (4.63), while neglecting the fluctuations in the meson fields:

$$\begin{aligned}S =&\beta V\Big[\frac{1}{2} m_\sigma^2 \langle\sigma\rangle^2 - \frac{1}{2} m_\omega^2 \langle\omega_0\rangle^2\Big] \\ &+ \frac{1}{\beta V} \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \sum_{n,\boldsymbol{k}} \sum_{m,\boldsymbol{k}'} \psi^\dagger_{n,\boldsymbol{k}} \mathrm{e}^{-\mathrm{i}(\omega_n\tau + \boldsymbol{k}\cdot\boldsymbol{x})} \Big(\frac{\partial}{\partial\tau} + \gamma_0\boldsymbol{\gamma}\cdot\nabla + \gamma_0 m^\star - \mu^\star\Big) \psi_{m,\boldsymbol{k}'} \mathrm{e}^{\mathrm{i}(\omega_m\tau + \boldsymbol{k}'\cdot\boldsymbol{x})} \\ =&\beta V\Big[\frac{1}{2} m_\sigma^2 \langle\sigma\rangle^2 - \frac{1}{2} m_\omega^2 \langle\omega_0\rangle^2\Big] + \sum_{n,\boldsymbol{k}} \psi^\dagger_{n,\boldsymbol{k}} \Big[\mathrm{i}\big(\omega_n + \gamma_0\boldsymbol{\gamma}\cdot\boldsymbol{k}\big) + \gamma_0 m^\star - \mu^\star\Big] \psi_{m,\boldsymbol{k}'}.\end{aligned} \tag{5.42}$$

Here we note some of the simplicity of the mean-field approximation. All terms in the Lagrangian that contain only meson fields, just contribute a factor βV times the mean of the Lagrangian to the action. Since we assume that the meson fields are classical, we do not integrate over them in the partition function. From (4.52), we then find

$$Z = \mathrm{e}^{-\frac{\beta V}{2}(m_\sigma^2\langle\sigma\rangle^2 - m_\omega^2\langle\omega_0\rangle^2)} \int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \mathrm{e}^{-\sum_{n,\boldsymbol{k}} \psi^\dagger_{n,\boldsymbol{k}} (\mathrm{i}\omega_n + \mathrm{i}\gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k} + \gamma^0 m^\star - \mu^\star)\psi_{n,\boldsymbol{k}}}. \tag{5.43}$$

This integral was solved in section 4.4-4.5 with the result

$$Z = \mathrm{e}^{-\frac{\beta V}{2}(m_\sigma^2\langle\sigma\rangle^2 - m_\omega^2\langle\omega_0\rangle^2)} \prod_{n,\boldsymbol{p}} \Big[\big(\omega_n + \mathrm{i}\mu^\star\big)^2 + E^2\Big], \tag{5.44}$$

$$\begin{aligned}\ln Z &= \frac{\beta V}{2}\Big(-m_\sigma^2\langle\sigma\rangle^2 + m_\omega^2\langle\omega_0\rangle^2\Big) + \sum_{n,\boldsymbol{p}} \ln\Big[\big(\omega_n + \mathrm{i}\mu^\star\big)^2 + E^2\Big] \\ &= \frac{\beta V}{2}\Big(-m_\sigma^2\langle\sigma\rangle^2 + m_\omega^2\langle\omega_0\rangle^2\Big) + 4\beta V \int \frac{\mathrm{d}^3k}{(2\pi)^3}\Big\{E + T\ln\big[1 + \mathrm{e}^{-\beta(E-\mu^\star)}\big] + T\ln\big[1 + \mathrm{e}^{-\beta(E+\mu^\star)}\big]\Big\},\end{aligned} \tag{5.45}$$

where E is given by (5.34). Note that in the second term, we have added an extra factor of two compared to the non-interacting Fermi gas expression (4.81). This is because the nucleon has two additional states compared to the Fermi gas we previously encountered: It can either be a proton or a neutron. We also see that in absence of the meson fields, the logarithm of the partition function is equal to (4.81), except for a factor $-\frac{\beta V}{2}$, as it should.

5.5 A small digression: Including an isospin chemical potential

We have assumed that we have a particle, the nucleon, with two possible isospin states: The neutron and the proton. The reason why we can do this, is because the free Dirac Lagrangian actually has an extra symmetry in addition to the global phase shift[5] $\psi \to \psi' = \mathrm{e}^{\mathrm{i}\alpha}\psi$: It also has an isospin symmetry under the transformation $\psi \to \psi' = \mathrm{e}^{-\mathrm{i}\alpha_i\tau_i/2}\psi$. Here τ_i denotes the Pauli matrices[6] given by (C.6). Since $\delta\psi = -\frac{1}{2}\mathrm{i}\alpha_i\tau_i\psi$, the conserved current is

$$j_{\mathrm{I}}^\mu = -\frac{\partial}{\partial(\partial_\mu\psi)}\bar{\psi}\Big(\mathrm{i}\partial\!\!\!/ + m\Big)\psi\frac{1}{2}\mathrm{i}\alpha_i\tau_i\psi + \frac{\partial}{\partial(\partial_\mu\bar{\psi})}\bar{\psi}\Big(\mathrm{i}\partial\!\!\!/ + m\Big)\psi\frac{1}{2}\mathrm{i}\alpha_i\tau_i\bar{\psi} = \frac{1}{2}\bar{\psi}\gamma^\mu\alpha_i\tau_i\psi. \tag{5.46}$$

If we let the rotation be infinitesimal, we obtain the conserved vector current

$$j_i^\mu = \frac{1}{2}\bar{\psi}\gamma^\mu\tau_i\psi. \tag{5.47}$$

The Pauli matrices do not commute, and thus it does only make sense to talk about one component of the conserved current. By convention, we choose to measure isospin along the third component. Since the Pauli matrices act in isospin space, the corresponding conserved charge is given by

$$Q_{\mathrm{I}} = \int \mathrm{d}^3x\, j_3^0 = \frac{1}{2}\int \mathrm{d}^3x\,\bar{\psi}\gamma^0\tau_3\psi = \frac{1}{2}\int \mathrm{d}^3x\,\begin{pmatrix}\bar{\psi}_{\mathrm{n}} & \bar{\psi}_{\mathrm{p}}\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\bar{\psi}_{\mathrm{n}}\\ \bar{\psi}_{\mathrm{p}}\end{pmatrix} = \frac{1}{2}\int \mathrm{d}^3x\big(\psi_{\mathrm{n}}^\dagger\psi_{\mathrm{n}} - \psi_{\mathrm{p}}^\dagger\psi_{\mathrm{p}}\big). \tag{5.48}$$

Again we stress that when we write τ_i, what we really mean is

$$\tau_i \otimes I_4 = \begin{pmatrix}I_4 & 0\\ 0 & -I_4\end{pmatrix}. \tag{5.49}$$

Since $\psi_{\mathrm{n}}^\dagger\psi_{\mathrm{n}}$ and $\psi_{\mathrm{p}}^\dagger\psi_{\mathrm{p}}$ are the baryon densities for neutrons and protons respectively, we know that the difference in number of protons and neutrons must be constant for this model. From this it is now possible to introduce a new chemical potential μ_{I} for the third component of the conserved isospin current. By writing

$$\mathscr{L} \to \mathscr{L} - \frac{1}{2}\mu_{\mathrm{I}}\psi^\dagger\tau_3\psi, \tag{5.50}$$

[5] It is important to keep in mind that we assume that the neutron and proton masses are the same. This is not entirely true as $m_{\mathrm{n}} = 939.56$MeV and $m_{\mathrm{p}} = 938.27$MeV

[6] It is usually convention to denote the Pauli matrices by τ^i when talking about isospin, and σ^i otherwise.

the partition function is from equation (4.71) and (5.45) given by

$$\ln Z = A + \ln\det\big[-\omega_n - \mathrm{i}\mu^\star - \frac{1}{2}\mathrm{i}\tau_3\mu_{\rm I} - \gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k} + \mathrm{i}\gamma^0 m^\star\big], \tag{5.51}$$

where we have denoted

$$A \equiv \beta V\Big(-\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle\Big). \tag{5.52}$$

In this notation, numbers are assumed to be multiplied with an identity matrix to match the dimensions of the matrices. For instance, we write

$$\omega_n + \gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k} \equiv \omega_n I_8 + \gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k}. \tag{5.53}$$

The fourth term in the determinant on the right hand side of (5.51) is equivalent to

$$-\gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k} = \begin{pmatrix} 0 & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & 0 & 0 \\ \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} \\ 0 & 0 & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & 0 \end{pmatrix}. \tag{5.54}$$

To simplify our expressions, we introduce the variables

$$\begin{aligned} a &= \mu^\star + \frac{1}{2}\mu_{\rm I} \qquad & b &= \mu^\star - \frac{1}{2}\mu_{\rm I} \\ c &= \omega_n + \mathrm{i}a \qquad & d &= \omega_n + \mathrm{i}b \end{aligned} \tag{5.55}$$

and use (4.70) to obtain

$$\begin{aligned} \ln Z &= A + \ln\det\begin{pmatrix} \big[-c+\mathrm{i}m^\star\big] & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & 0 & 0 \\ \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & \big[-c-\mathrm{i}m^\star\big] & 0 & 0 \\ 0 & 0 & \big[-d+\mathrm{i}m^\star\big] & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} \\ 0 & 0 & \mathrm{i}\boldsymbol{\sigma}\cdot\boldsymbol{k} & \big[-d-\mathrm{i}m^\star\big] \end{pmatrix} \\ &= A + \ln\det\begin{pmatrix} \big[c^2+(m^\star)^2+k^2\big] & 0 \\ 0 & \big[d^2+(m^\star)^2+k^2\big] \end{pmatrix} \\ &= A + \ln\Big\{\Big[c^2+(m^\star)^2+k^2\Big]^2\Big[d^2+(m^\star)^2+k^2\Big]^2\Big\} \\ &= A + \mathrm{Tr}\Big\{\ln\Big[c^2+(m^\star)^2+k^2\Big]^2 + \ln\Big[d^2+(m^\star)^2+k^2\Big]^2\Big\} \\ &= A + \sum_{n,\boldsymbol{k}}\ln\Big[\big(\omega_n+\mathrm{i}a\big)^2+E^2\Big] + \sum_{n,\boldsymbol{k}}\ln\Big[\big(\omega_n+\mathrm{i}b\big)^2+E^2\Big], \end{aligned} \tag{5.56}$$

where $E^2 = (m^\star)^2 + k^2$. Both terms are of the same form as (4.77), with the result

$$\begin{aligned} \ln Z = A &+ \sum_{n,\boldsymbol{k}}\ln\Big[\omega_n^2+\big(E-a\big)^2\Big] + \sum_{n,\boldsymbol{k}}\ln\Big[\omega_n^2+\big(E+a\big)^2\Big] \\ &+ \sum_{n,\boldsymbol{k}}\ln\Big[\omega_n^2+\big(E-b\big)^2\Big] + \sum_{n,\boldsymbol{k}}\ln\Big[\omega_n^2+\big(E+b\big)^2\Big]. \end{aligned} \tag{5.57}$$

Using the identity (4.78) in the continuum limit, we then obtain

$$
\begin{aligned}
T\ln Z =& A+\frac{1}{V}\int\frac{\mathrm{d}^3k}{(2\pi)^3}\Bigg\{\big(E-a\big)+\big(E-b\big)+\big(E+a\big)+\big(E+b\big)\\
&+2T\Big[\ln\Big(1+\mathrm{e}^{-\beta(E-a)}\Big)+\ln\Big(1+\mathrm{e}^{-\beta(E+a)}\Big)+\ln\Big(1+\mathrm{e}^{-\beta(E-b)}\Big)+\ln\Big(1+\mathrm{e}^{-\beta(E+b)}\Big)\Big]\Bigg\}\\
=& A+\frac{2}{V}\int\frac{\mathrm{d}^3k}{(2\pi)^3}\Big\{2E+T\ln\Big[1+\mathrm{e}^{-\beta(E-a)}\Big]+T\ln\Big[1+\mathrm{e}^{-\beta(E+a)}\Big]\\
&+T\ln\Big[1+\mathrm{e}^{-\beta(E-b)}\Big]+T\ln\Big[1+\mathrm{e}^{-\beta(E+b)}\Big]\Big\}. \qquad (5.58)
\end{aligned}
$$

We see that we have the same contribution to the vacuum energy, as we still have 8 possible states: Neutron/proton, spin up/spin down and particle/anti-particle. However, we now get two extra terms that take into account the forces due to the difference in number of protons and neutrons. Note also that in the case when $\mu_{\rm I}\to 0$, we find that $a\to\mu^\star$ and $b\to\mu^\star$, so that we recover (5.45), as we should.

When introducing the chemical potentials, we could have as an alternative approach, defined the chemical potential for the proton and the neutron density, $\mu_{\rm p}$ and $\mu_{\rm n}$, as oppose to the baryon density and the difference in proton and neutron densities, μ and $\mu_{\rm I}$. This would give us the partition function

$$\ln Z=\ln\det\big[-\omega_n-\mathrm{i}\mu+g_\omega\langle\omega_0\rangle-\gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k}+\mathrm{i}\gamma^0 m^\star\big], \qquad (5.59)$$

where we have defined the matrix

$$\mu=\begin{pmatrix}\mu_{\rm p} & 0\\ 0 & \mu_{\rm n}\end{pmatrix}. \qquad (5.60)$$

Then we obtain

$$
\begin{aligned}
\ln Z =& \frac{2}{V}\int\frac{\mathrm{d}^3k}{(2\pi)^3}\Big\{2E+T\ln\Big[1+\mathrm{e}^{-\beta(E-\mu^\star_{\rm p})}\Big]+T\ln\Big[1+\mathrm{e}^{-\beta(E+\mu^\star_{\rm p})}\Big]\\
&+T\ln\Big[1+\mathrm{e}^{-\beta(E-\mu^\star_{\rm n})}\Big]+T\ln\Big[1+\mathrm{e}^{-\beta(E+\mu^\star_{\rm n})}\Big]\Big\}, \qquad (5.61)
\end{aligned}
$$

with

$$\mu^\star_{\rm p}=\mu_{\rm p}-g_\omega\langle\omega_0\rangle, \qquad \mu^\star_{\rm n}=\mu_{\rm n}-g_\omega\langle\omega_0\rangle. \qquad (5.62)$$

Again we see that in the limit when $\mu^\star_{\rm p}=\mu^\star_{\rm n}\equiv\mu^\star$, (5.45) is obtained. For simplicity, we will as a first approximation assume that we have isospin symmetric matter. In other words we set $\mu_{\rm I}=0$ (or $\mu^\star_{\rm p}=\mu^\star_{\rm n}$), and save this subject for later.

5.6 The equation of state

Having found the partition function, we are now ready to calculate the EoS for the σ-ω model. Firstly, the pressure is from (4.76) given by

$$P=-\Omega=\frac{1}{\beta V}\ln Z=-\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2+\frac{1}{2}m_\omega^2\langle\omega_0\rangle^2+P_{\rm FG}. \qquad (5.63)$$

Here we have denoted the pressure for a free Fermi gas with chemical potential $\mu^\star$ and mass $m^\star$ as

$$P_{\rm FG}=f\int\frac{\mathrm{d}^3k}{(2\pi)^3}\Big\{2E+T\ln\Big[1+\mathrm{e}^{-\beta(E-\mu^\star)}\Big]+T\ln\Big[1+\mathrm{e}^{-\beta(E+\mu^\star)}\Big]\Big\}, \qquad (5.64)$$

where f is the degeneracy factor. Having assumed that the proton and neutron are two states of the same particle, the choice $f=4$ seems reasonable (neutron, anti-neutron, proton, anti-proton). However, since neutron stars mainly consist of neutrons, a model where the proton states are suppressed, might be a more realistic approach. We will in the following do all calculations for both cases $f = 2$ (neutron matter) and $f = 4$ (nuclear matter) to see their differences and similarities.

From (5.64) we can find the mean fields $\langle\sigma\rangle$ and $\langle\omega_0\rangle$. In equilibrium, the system will have minimized its energy. Then it follows that $\langle\sigma\rangle$ and $\langle\omega_0\rangle$ must minimize the pressure. Thus,

$$\frac{\partial P}{\partial m^\star} = \frac{\partial P}{\partial \mu^\star} = 0, \tag{5.65}$$

which gives

$$\frac{\partial P}{\partial m^\star} = -\frac{\partial}{\partial m^\star}\Big(\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2\Big) + \frac{\partial P_{\text{FG}}}{\partial m^\star} = -\frac{\partial\langle\sigma\rangle}{\partial m^\star}\frac{\partial}{\partial\langle\sigma\rangle}\Big(\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2\Big) + \frac{\partial P_{\text{FG}}}{\partial m^\star} = \frac{m_\sigma^2}{g_\sigma}\langle\sigma\rangle + \frac{\partial P_{\text{FG}}}{\partial m^\star} = 0, \tag{5.66}$$

$$\frac{\partial P}{\partial \mu^\star} = \frac{\partial}{\partial \mu^\star}\Big(\frac{1}{2}m_\omega^2\langle\omega_0\rangle^2\Big) + \frac{\partial P_{\text{FG}}}{\partial \mu^\star} = \frac{\partial\langle\omega_0\rangle}{\partial \mu^\star}\frac{\partial}{\partial\langle\omega_0\rangle}\Big(\frac{1}{2}m_\omega^2\langle\omega_0\rangle^2\Big) + \frac{\partial P_{\text{FG}}}{\partial \mu^\star} = -\frac{m_\omega^2}{g_\omega}\langle\omega_0\rangle + \frac{\partial P_{\text{FG}}}{\partial \mu^\star} = 0, \tag{5.67}$$

where we have used the relations

$$\frac{\partial\langle\sigma\rangle}{\partial m^\star} = \left(\frac{\partial m^\star}{\partial\langle\sigma\rangle}\right)^{-1} = \left[\frac{\partial(m - g_\sigma\langle\sigma\rangle)}{\partial\langle\sigma\rangle}\right]^{-1} = -g_\sigma^{-1} = -\frac{1}{g_\sigma}, \tag{5.68}$$

$$\frac{\partial\langle\omega_0\rangle}{\partial \mu^\star} = \left(\frac{\partial \mu^\star}{\partial\langle\omega_0\rangle}\right)^{-1} = \left[\frac{\partial(\mu - g_\omega\langle\omega_0\rangle)}{\partial\langle\omega_0\rangle}\right]^{-1} = -g_\omega^{-1} = -\frac{1}{g_\omega}. \tag{5.69}$$

Solving for the mean meson-fields, we find

$$\langle\sigma\rangle = -\frac{g_\sigma}{m_\sigma^2}\frac{\partial P_{\text{FG}}}{\partial m^\star}, \tag{5.70}$$

$$\langle\omega_0\rangle = \frac{g_\omega}{m_\omega^2}\frac{\partial P_{\text{FG}}}{\partial \mu^\star}. \tag{5.71}$$

Still ignoring the zero-point energy in the free Fermi gas expression[7], we find

$$\begin{aligned}\frac{\partial P_{\text{FG}}}{\partial m^\star} &= f\int\frac{\mathrm{d}^3k}{(2\pi)^3}\frac{\partial}{\partial m^\star}\Big\{T\ln\Big[1+\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}-\mu^\star\right)}\Big] + T\ln\Big[1+\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}+\mu^\star\right)}\Big]\Big\}\\ &= \frac{fT}{2\pi^2}\int\mathrm{d}k\,\frac{k^2}{\sqrt{k^2+(m^\star)^2}}\left[\frac{-\beta m^\star\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}-\mu^\star\right)}}{1+\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}-\mu^\star\right)}} + \frac{-\beta m^\star\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}+\mu^\star\right)}}{1+\mathrm{e}^{-\beta\left(\sqrt{k^2+(m^\star)^2}+\mu^\star\right)}}\right]\\ &= -\frac{fT}{2\pi^2}\int\mathrm{d}k\,\frac{k^2m^\star}{E}\left[\frac{1}{\mathrm{e}^{\beta(E-\mu^\star)}+1} + \frac{1}{\mathrm{e}^{\beta(E+\mu^\star)}+1}\right],\end{aligned} \tag{5.72}$$

and

$$\begin{aligned}\frac{\partial P_{\text{FG}}}{\partial \mu^\star} =& \frac{fT}{2\pi^2}\int\mathrm{d}k\left[\frac{\beta\mathrm{e}^{-\beta(E-\mu^\star)}}{1+\mathrm{e}^{-\beta(E-\mu^\star)}} + \frac{-\beta\mathrm{e}^{-\beta(E+\mu^\star)}}{1+\mathrm{e}^{-\beta(E+\mu^\star)}}\right]\\ =& \frac{f}{2\pi^2}\int\mathrm{d}k\left[\frac{1}{\mathrm{e}^{\beta(E-\mu^\star)}+1} - \frac{1}{\mathrm{e}^{\beta(E+\mu^\star)}+1}\right].\end{aligned} \tag{5.73}$$

In the zero-temperature limit, these expressions simplify to

$$\begin{aligned}\frac{\partial P_{\text{FG}}}{\partial m^\star} &= -\frac{f}{2\pi^2}\int\mathrm{d}k\,\frac{m^\star k^2}{E}\Theta(E-\mu^\star)\\ &= -\frac{f}{2\pi^2}\int_0^{k_{\text{F}}}\mathrm{d}k\,\frac{m^\star k^2}{\sqrt{k^2+(m^\star)^2}}\\ &= -\frac{fm^\star}{4\pi^2}\left[k_{\text{F}}\sqrt{k_{\text{F}}^2+(m^\star)^2} - (m^\star)^2\ln\!\left(\frac{\sqrt{k_{\text{F}}^2+(m^\star)^2}+k_{\text{F}}}{m^\star}\right)\right],\end{aligned} \tag{5.74}$$

[7] As mentioned before, we will take care of this in chapter 7.

and

$$\frac{\partial P_{\mathrm{FG}}}{\partial \mu^\star} = \frac{f}{2\pi^2}\int \mathrm{d}k\,\Theta(E-\mu^\star) = \frac{f}{2\pi^2}\int_0^{k_{\mathrm{F}}}\mathrm{d}k\,k^2 = \frac{f}{6\pi^2}k_{\mathrm{F}}^3. \tag{5.75}$$

Here we have denoted the Fermi momentum $k_F = \sqrt{(\mu^\star)^2-(m^\star)^2}$. We can then write the expectations of the meson fields as

$$\langle\sigma\rangle = \frac{fg_\sigma m^\star}{4\pi^2 m_\sigma^2}\left[k_{\mathrm{F}}\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}-(m^\star)^2\ln\!\left(\frac{\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}+k_{\mathrm{F}}}{m^\star}\right)\right], \tag{5.76}$$

$$\langle\omega_0\rangle = \frac{fg_\omega}{6\pi^2 m_\omega^2}k_{\mathrm{F}}^3. \tag{5.77}$$

From equation (4.91), we have the baryon density

$$\rho = \frac{\partial P}{\partial\mu} = \frac{\partial P_{\mathrm{FG}}}{\partial\mu} = \frac{\partial P_{\mathrm{FG}}}{\partial\mu^\star}. \tag{5.78}$$

Further, from the action (5.42) we see that taking derivatives of the partition function with respect to $m^\star$ brings down a factor of $-\beta V\bar{\psi}\psi$ from the exponential, and thus

$$\rho_{\mathrm{s}} \equiv \langle\bar{\psi}\psi\rangle = -\frac{1}{Z\beta V}\frac{\partial Z}{\partial m^\star} = -\frac{1}{\beta V}\frac{\partial \ln Z}{\partial m^\star} = -\frac{\partial P_{\mathrm{FG}}}{\partial m^\star}. \tag{5.79}$$

Thus, we may write the mean meson-fields in a simpler manner as

$$\langle\sigma\rangle = \frac{g_\sigma}{m_\sigma^2}\rho_{\mathrm{s}}, \tag{5.80}$$

$$\langle\omega_0\rangle = \frac{g_\omega}{m_\omega^2}\rho. \tag{5.81}$$

Going back to equation (5.63), and using the free Fermi gas pressure expression obtained in (4.96), we find the pressure in the zero-temperature limit

$$P = -\frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2}\right)\rho_{\mathrm{s}}^2 + \frac{1}{2}\left(\frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + P_{\mathrm{FG}}, \tag{5.82}$$

with

$$P_{\mathrm{FG}} = \frac{f}{48\pi^2}\left\{\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}\Big[2k_{\mathrm{F}}^3-3\,(m^\star)^2k_{\mathrm{F}}\Big]+3\,(m^\star)^4\ln\!\left[\frac{k_{\mathrm{F}}+\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}}{m^\star}\right]\right\}. \tag{5.83}$$

Further, the energy density is given by (4.94), with the result

$$\epsilon = \mu\rho - P = \mu^\star\rho + g_\omega\langle\omega_0\rangle\rho - P = \mu^\star\rho + \left(\frac{g_\omega^2}{m_\omega^2}\right)\rho^2 - P = \frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2}\right)\rho_{\mathrm{s}}^2 + \frac{1}{2}\left(\frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \epsilon_{\mathrm{FG}}, \tag{5.84}$$

where we have introduced the energy density of an ideal cold Fermi gas with mass $m^\star$ and chemical potential $\mu^\star$

$$\epsilon_{\mathrm{FG}} = \frac{f}{48\pi^2}\left\{\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}\Big[6k_{\mathrm{F}}^3+3\,(m^\star)^2k_{\mathrm{F}}\Big]-3\,(m^\star)^4\ln\!\left[\frac{k_{\mathrm{F}}+\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}}{m^\star}\right]\right\}. \tag{5.85}$$

It is important to note that we in equation (5.84) still have to use the "real" chemical potential μ, not $\mu^\star$, when we calculate the energy density. From these results, we see that in the limit where there are no interactions, the expressions for the pressure and energy density become the same as the ones for a free cold Fermi gas.

In the non-relativistic case $k_{\mathrm{F}} \ll m$, which is the same as low densities, we have

$$\begin{aligned}
\left(\rho_{\mathrm{s}}\right)_{\mathrm{NR}} &= \frac{fm^\star}{4\pi^2}\left[\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}\,k_{\mathrm{F}} - (m^\star)^2 \ln\!\left(\frac{\sqrt{k_{\mathrm{F}}^2+(m^\star)^2}+k_{\mathrm{F}}}{m^\star}\right)\right] \\
&\approx \frac{fm^\star}{4\pi^2}\left[\left(m^\star + \frac{k_{\mathrm{F}}^2}{2m^\star}\right)k_{\mathrm{F}} - (m^\star)^2\left(\frac{k_{\mathrm{F}}}{m^\star} - \frac{k_{\mathrm{F}}^3}{6\,(m^\star)^3}\right)\right] \\
&= \frac{f}{6\pi^2}k_{\mathrm{F}}^3 = \rho. \qquad (5.86)
\end{aligned}$$

We then see that in this limit, the effective mass and chemical potential become

$$m^\star_{\mathrm{NR}} \approx m - \frac{g_\sigma^2}{m_\sigma^2}\rho \approx m, \qquad \mu^\star_{\mathrm{NR}} = \mu - \frac{g_\omega^2}{m_\omega^2}\rho \approx \mu. \qquad (5.87)$$

In the previous chapter, we found the low-density limits for P_{FG} and ϵ_{FG}. Using that $\rho \sim k_F^3$, we then find

$$P_{\mathrm{NR}} = -\frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2} - \frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \left(P_{\mathrm{FG}}\right)_{\mathrm{NR}} = -\frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2} - \frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \frac{fk_{\mathrm{F}}^5}{30\pi^2 m} \approx \frac{fk_{\mathrm{F}}^5}{30\pi^2 m}, \qquad (5.88)$$

$$\epsilon_{\mathrm{NR}} = \frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2} + \frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \left(\epsilon_{\mathrm{FG}}\right)_{\mathrm{NR}} = \frac{1}{2}\left(\frac{g_\sigma^2}{m_\sigma^2} + \frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \frac{f}{6\pi^2}k_{\mathrm{F}}^3 m + \frac{k_{\mathrm{F}}^5}{20m\pi^2} \approx \frac{f}{6\pi^2}k_{\mathrm{F}}^3 m + \frac{fk_{\mathrm{F}}^5}{20m\pi^2}. \qquad (5.89)$$

Note that in the non-relativistic limit, the pressure and energy density are both independent of the coupling constants. Comparing this to the results obtained in section 4.6, we find that the gas behaves as a free Fermi gas in the non-relativistic limit. This is because, at low densities, the particles are so far apart that they hardly effect each other.

In the high-density limit $m \ll k_{\mathrm{F}}$, we expand around $k_{\mathrm{F}} = \infty$ to find the scalar density

$$(\rho_{\mathrm{s}})_{\mathrm{UR}} \approx \frac{fm^\star}{4\pi^2}k_{\mathrm{F}}^2. \qquad (5.90)$$

From this, the high-density limit for the effective mass is

$$m^\star_{\mathrm{UR}} \approx m - \left(\frac{g_\sigma^2}{m_\sigma^2}\right)\frac{fm^\star_{\mathrm{UR}}}{4\pi^2}k_{\mathrm{F}}^2. \qquad (5.91)$$

Solving for $m^\star_{\mathrm{UR}}$, we obtain

$$m^\star_{\mathrm{UR}} = \frac{m}{1 + \frac{fg_\sigma^2}{4\pi^2 m_\sigma^2}k_{\mathrm{F}}^2}. \qquad (5.92)$$

Note that this means that the effective mass goes to zero for high densities. Furthermore, inserting (5.92) into (5.90), we find that for $m \ll k_{\mathrm{F}}$, we have $(\rho_{\mathrm{s}})_{\mathrm{UR}} \sim 1$. To the fourth power in k_{F}, the pressure then becomes

$$P_{\mathrm{UR}} = \frac{1}{2}\left(\frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \frac{fk_{\mathrm{F}}^4}{24\pi^2} = \frac{f^2}{72\pi^4}\left(\frac{g_\omega^2}{m_\omega^2}\right)k_{\mathrm{F}}^6 + \frac{fk_{\mathrm{F}}^4}{24\pi^2}. \qquad (5.93)$$

In the same manner, the energy density is

$$\epsilon_{\mathrm{UR}} = \frac{1}{2}\left(\frac{g_\omega^2}{m_\omega^2}\right)\rho^2 + \frac{fk_{\mathrm{F}}^4}{8\pi^2} = \frac{f^2}{72\pi^4}\left(\frac{g_\omega^2}{m_\omega^2}\right)k_{\mathrm{F}}^6 + \frac{fk_{\mathrm{F}}^4}{8\pi^2} \qquad (5.94)$$

From equation (4.154), we then find that the speed of sound in this limit is

$$\frac{\partial P_{\mathrm{UR}}}{\partial \epsilon_{\mathrm{UR}}} = \frac{\partial P_{\mathrm{UR}}}{\partial k_{\mathrm{k}}}\frac{\partial k_{\mathrm{F}}}{\partial \epsilon_{\mathrm{UR}}} = \frac{\frac{f^2}{12\pi^4}\left(\frac{g_\omega^2}{m_\omega^2}\right)k_{\mathrm{F}}^5 + \frac{fk_{\mathrm{F}}^3}{6\pi^2}}{\frac{f^2}{12\pi^4}\left(\frac{g_\omega^2}{m_\omega^2}\right)k_{\mathrm{F}}^5 + \frac{fk_{\mathrm{F}}}{2\pi^2}} = 1 - \frac{4\pi^2}{f\left(\frac{g_\omega^2}{m_\omega^2}\right)k_{\mathrm{F}}^2 + 6\pi^2}. \qquad (5.95)$$

Hence, the speed of sound approaches the speed of light as the density increases. This is a factor of $\sqrt{3}$ larger than the maximum speed of sound in a free cold Fermi gas which we found earlier to be $\frac{c}{\sqrt{3}}$. The speed of sound is in general higher in an interacting gas, as the EoS becomes more stiff than in a free one. Also, we note that the speed of sound never surpasses the speed of light, which is consistent with special relativity.

5.6.1 The electron

We have assumed that the protons that are present in the case of nuclear matter come from the beta decay of the neutrons. We should therefore expect that there are equally many electrons as protons.[8] In other words, they must have the same density

$$\rho_{\mathrm{e}} = \rho_{\mathrm{p}} = \frac{\rho(f-2)}{4}. \tag{5.96}$$

However, in equilibrium, both processes

$$\mathrm{n} \to \mathrm{p} + \mathrm{e} + \bar{\nu}_{\mathrm{e}} \qquad \text{and} \qquad \mathrm{p} + \mathrm{e} + \bar{\nu}_{\mathrm{e}} \to \mathrm{n}, \tag{5.97}$$

where n, p, e and ν denotes the neutron, the proton, the electron and the neutrino, respectively, should be equally likely. Forgetting about the neutrino as it has a negligible mass compared to the other particles, this means that we can write

$$\mu_{\mathrm{n}} = \mu_{\mathrm{p}} + \mu_{\mathrm{e}}. \tag{5.98}$$

But since we have assumed isospin-symmetric matter, and that the neutron and proton masses are the same, this would mean that the chemical potential for the electron vanishes

$$\mu_{\mathrm{e}} = 0, \tag{5.99}$$

which would force us to set the electron density to zero. This means that as long as we are in isospin-symmetric matter, we cannot treat the beta decay in a consistent way. In the next chapter we will include an asymmetry in the proton and neutron densities that fixes this issue, but for now we just leave it as a remark that there still are simple features of matter that this model does not handle well.

5.7 Numerical solutions to the problem

We have found an expression for the EoS, and want to find the solutions of the TOV-equation. However, there are some minor issues we need to handle before we can proceed and calculate the mass-radius relation for this model.

5.7.1 Determining the coupling constants

Firstly, we need to determine the coupling constants. The coupling constants represent two free parameters of our model, and thus we must choose them so that they reproduce experimental result. We define nuclear matter as a system of equally many neutrons and protons that only interact through the nuclear forces. For infinite nuclear matter, that is, nuclear matter without boundaries, one observes that the radius of a nuclei scales approximately as $A^{1/3}$ where A is the number of nuclei. Since the neutron and proton masses are approximately the same, the total mass of the system increases proportional to A. This means that the nucleon density becomes a constant, which we will call the saturation density. The value we will use for the saturation density in this thesis is $\rho_0 = 0.153\mathrm{fm}^3$ [34].

Another property that we should incorporate in the model, is the binding energy. The negative of the binding energy is defined as the minimum energy needed to separate all the constituents of a system into free parts. In mathematical terms, the binding energy per nucleon is then defined as

$$B = \frac{\epsilon}{\rho} - m. \tag{5.100}$$

[8] This is only true if beta decay is the only source of particles different from neutrons.

By definition, the saturation density is the density that minimizes the binding energy. Experimentally, one finds that the binding energy at the saturation density is $B_0 = -16.3\,\mathrm{MeV}$ [34]. Thus, we must choose the coupling constants so that they satisfy

$$B_0 = \frac{\epsilon_0}{\rho_0} - m = -16.3\,\mathrm{MeV}, \tag{5.101}$$

$$\frac{\mathrm{d}B}{\mathrm{d}\rho}\Big|_{\rho=\rho_0} = \frac{\rho_0 \frac{\partial\epsilon}{\partial\rho}\big|_{\rho=\rho_0} - \epsilon_0}{\rho_0^2} = 0, \tag{5.102}$$

where the subscript 0 emphasizes that the values are at saturation.

In the code given by Appendix G.2, the function "couplingConstans" returns the coupling constants that satisfy the conditions (5.101) and (5.102) best. To use the program, we first need to choose a range we expect g_σ and g_ω to be within. In the calculations, we have looked at values between 0 and 200 for both g_σ and g_ω. The function then divides the domains for the couplings in N intervals with equal distance. For each of the N^2 sets of couplings, the program creates the variables

$$\text{derivativeEpsilon,} = \frac{\partial\epsilon(\rho_0, g_\sigma, g_\omega)}{\partial\rho} \qquad \text{and} \qquad \text{testBindingEnergy} = B(\rho_0, g_\sigma, g_\omega). \tag{5.103}$$

The coupling constants that minimizes the function

$$\text{bestFit} = \sqrt{\left(\rho_0 \frac{\text{derivativeEpsilon}}{\epsilon_0} - 1\right)^2 + \left(\frac{\text{testBindingEnergy}}{B_0} - 1\right)^2}, \tag{5.104}$$

is the best fit for the conditions (5.101) and (5.102). To increase accuracy, we run the function "couplings". This function calls "couplingConstants" multiple times, decreasing the interval we look for g_σ and g_ω by a factor of 10 each time. For instance, starting with the assumption g_σ and g_ω in $[0, 200]$, if "couplingConstants" returns $g_\sigma = 10$ and $g_\omega = 20$, the function runs "couplingConstants" again looking for g_σ in $[0, 20]$ and g_ω in $[10, 30]$. The intervals are still divided in N pieces, and thus increased accuracy is achieved faster than if one just increased N. When calculating the couplings, the value $N = 1000$ is used, and the function "couplings" is set to decrease the intervals 4 times.

The values obtained for the couplings are

$$g_\sigma = 10.94, \qquad g_\omega = 13.59, \tag{5.105}$$

which is consistent with Kapusta and Gale [25, p. 224]. In the calculations we have in the same manner as Kapusta and Gale assumed the particle masses $m_\omega = 783\,\mathrm{MeV}$ and $m_\sigma = 550\,\mathrm{MeV}$. It should be noted that there is a large uncertainty in the mass of the σ-meson. In the σ-ω model, the σ-meson is meant to represent an exchange of two pions, and corresponds to a resonance in π-π scattering. According to [25, p. 224] this resonance is between $500 - 600\,\mathrm{MeV}$ and so we must choose a value somewhere in-between to obtain a definite result. However, our choices of the meson masses in this model does not actually matter. This is because we could have expressed the energy density and pressure as a function of the variables

$$x = \left(\frac{g_\omega}{m_\omega}\right), \qquad y = \left(\frac{g_\sigma}{m_\sigma}\right). \tag{5.106}$$

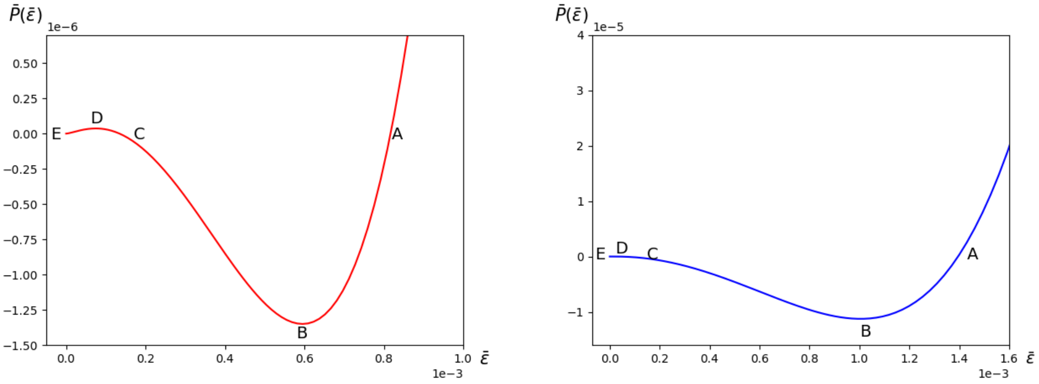
By doing so, the equations becomes consistent no matter what value m_σ has, because the coupling g_σ would always adjust so that y becomes a constant to ensure that (5.101) and (5.102) holds.

5.7.2 The pressure

Having found the coupling constants, we are ready to calculate the pressure and energy density. In a similar manner as before, we introduce the dimensionless quantities[9]

$$\bar{P} = \frac{P}{m^4}, \qquad \bar{\epsilon} = \frac{\epsilon}{m^4}, \qquad \bar{m}_\sigma = \frac{m_\sigma}{m}, \qquad \bar{m}_\omega = \frac{m_\omega}{m}, \qquad \bar{k}_{\mathrm{F}} = \frac{k_{\mathrm{F}}}{m}, \qquad \bar{m}^\star = \frac{m^\star}{m}. \tag{5.107}$$

[9] Remember that we, in contrast to the previous chapter, now use natural units. The scaling factor is the same; in units where $\hbar \neq c \neq 0$ we have for instance $\bar{P} = \frac{\hbar^3 P}{m^4 c^5}$.



(a) Neutron matter

(b) Nuclear matter

Figure 5.1: Dimensionless pressure as a function of the dimensionless energy density in the σ-ω model. The letters A-E are placed as references to make it easier to compare with other plots.

Figure 5.1 shows a plot of the dimensionless pressure as a function of the dimensionless energy density. We observe that the EoS has a region where

$$\frac{\mathrm{d}\overline{P}}{\mathrm{d}\bar{\epsilon}} < 0. \tag{5.108}$$

This region, that is the curve DB, is not stable according to our stability analysis in the section 4.7.4. This suggests that there is something wrong with our model. We know that the pressure must be continuous as a function of energy density, otherwise matter would flow from high pressure to low pressure until the pressure eventually becomes continuous. A stable solution must therefore connect the region before A and the region after E in a continuous way.

One possible explanation is that the matter undergoes a phase transition. To investigate this further, we look at a more familiar example, the Van der Waals gas. For a constant temperature below some critical temperature T_{crit}, the Van der Waals equation of state has the qualitative form given by Figure 5.2. Assuming first that there is no phase transition, we move along the path ABCDE. The total work done by this process is given by

$$W = \int_{V_{\mathrm{A}}}^{V_{\mathrm{E}}} \mathrm{d}V\, P(V), \tag{5.109}$$

since the temperature is constant. On the other hand, if a phase transition occurs, we will have a coexistence of two phases, which in the Van der Waals case is the coexistence of vapour and liquid. In the region with coexisting phases, the pressure stays constant. We can see this by imagining a liquid in contact with vapour through an interface. A liquid is in general much less compressible than vapour. If we then decrease the volume, the pressure will increase in the vapour while the pressure in the liquid remains approximately the same. The pressure difference will then force vapour molecules to liquefy until equilibrium is achieved. On the other hand, if we increase the volume, the vapor pressure decreases, and liquid molecules vaporizes until the pressure again is in equilibrium.

In the density regions where there is a coexistence between two phases, we insert a horizontal line segment between A and E. The work done during the phase transition must be the same as the work done along the non-transitioning path, otherwise the cycle ABCDEA would not be possible without changing the temperature. The work done along constant volume is zero. Thus, if a phase transition really occurs, we can choose a critical pressure P_{crit} so that the areas ABC and CDE are the same, and then substitute the curve ABCDE with a horizontal line AE. This procedure is called the Maxwell construction after J. C. Maxwell who first proposed it in 1875 [35]. For more about the Maxwell constrction and the phase transition in the Van der Waals gas, see for example [36].

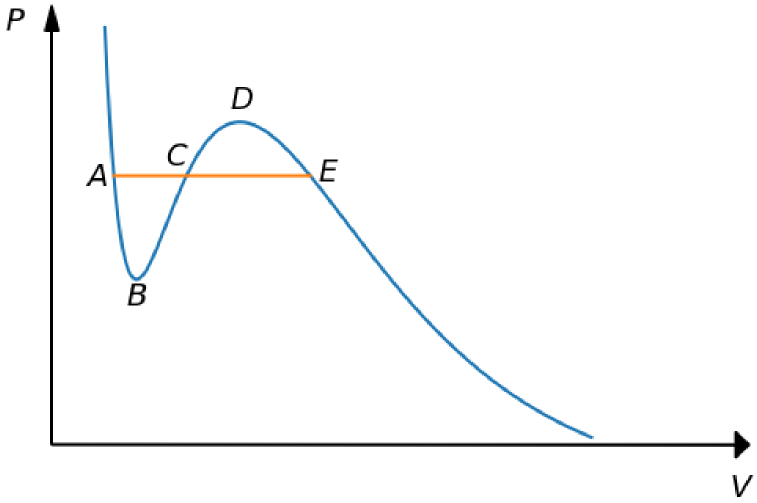


Figure 5.2: Sketch of an isotherm of the Van der Waals equation of state below the critical temperature. The height of the horizontal line segment ACE is chosen so that the area ABC is equal to the area CDE.

The question now arises whether we can apply the Maxwell construction to the σ-ω EoS. If we imagine keeping the energy constant, we can plot the pressure as a function of volume by inverting the energy density. In other words, we write

$$\bar{V} = \frac{\bar{E}}{\bar{\epsilon}}, \tag{5.110}$$

so that

$$\bar{P}(\bar{V}) = C\bar{P}\left(\frac{1}{\bar{\epsilon}}\right), \tag{5.111}$$

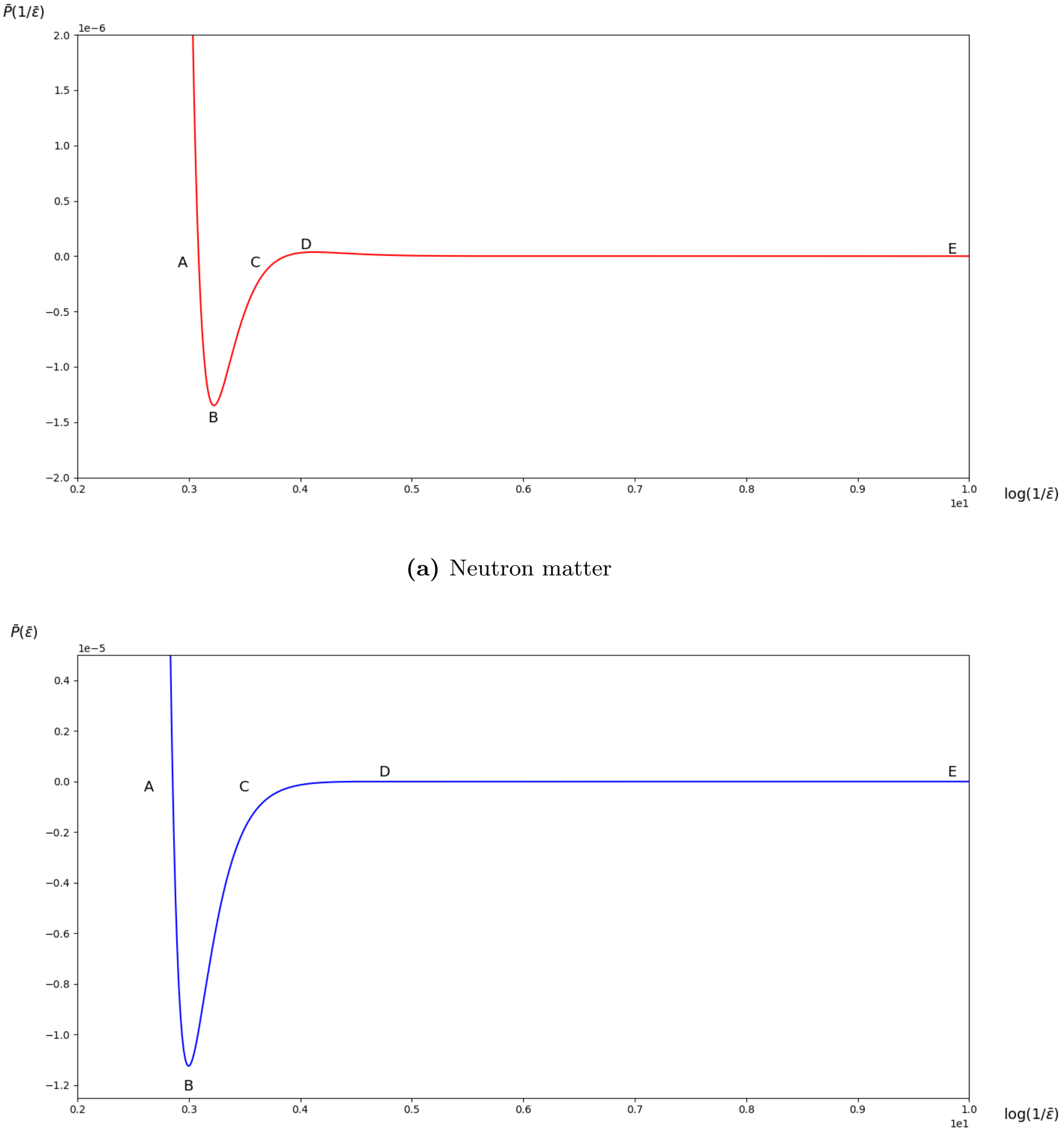
where C is some constant. Figure 5.3 shows the pressure as a function of the inverse energy density. This plot then represents the pressure as function of volume, up to some constant. We see that the pressure for small energy densities, and thus for large volumes, tends to zero. This means that the maximal area we can obtain under the curve ABC is given when the critical pressure is zero. Under closer inspection, we find by integrating that the maximal area under ABC is smaller than the area under CDE for both $f = 2$ and $f = 4$. The Maxwell construction is thus not possible, and a coexistence of liquid and vapor cannot solve the stability issues.

It should be mentioned that Walecka in his original paper [33] performs the Maxwell construction in the case $f = 2$. He can do this because he uses different coupling constants due to the fact that the binding energy for infinite nuclear matter at that time was measured to be $-15.7\,\text{MeV}$, and not $-16.3\,\text{MeV}$ as we have used here.

Having ruled out the possibility of a phase transition, we need another way to solve this issue. One possible answer might be that the model is just to simple to account for all the physics involved. When we determined the coupling constants, we fitted them to the observed values for the binding energy and saturation density. Only having two coupling constants at our disposal, we cannot expect to be able to fit all empirically known properties of nuclear matter. Especially, the compressibility, which is defined as the relative change in volume due to a change in the pressure, deviates badly from the observed value, which is somewhere in the range $K = 200 - 300$ MeV [37, 38, 39, 40] at saturation. In particular, one finds that for the σ-ω model

$$K = k_{\mathrm{F}}^2 \frac{\mathrm{d}^2}{\mathrm{d}\,k_{\mathrm{F}}^2}\left(\frac{\epsilon}{\rho}\right) = 563\,\text{MeV}, \tag{5.112}$$

at saturation. Later we will add more particles in this model to obtain extra degrees of freedom so that we become able to fix more properties of nuclear matter, and hopefully obtain a more accurate result.



(a) Neutron matter

(b) Nuclear matter

Figure 5.3: Dimensionless pressure as a function of the inverse dimensionless energy density in the σ-ω model. Letters A-E are placed so that they correspond to previous Figures, except that the area under ABC is not equal to the area under CDE for reasons explained in the text.

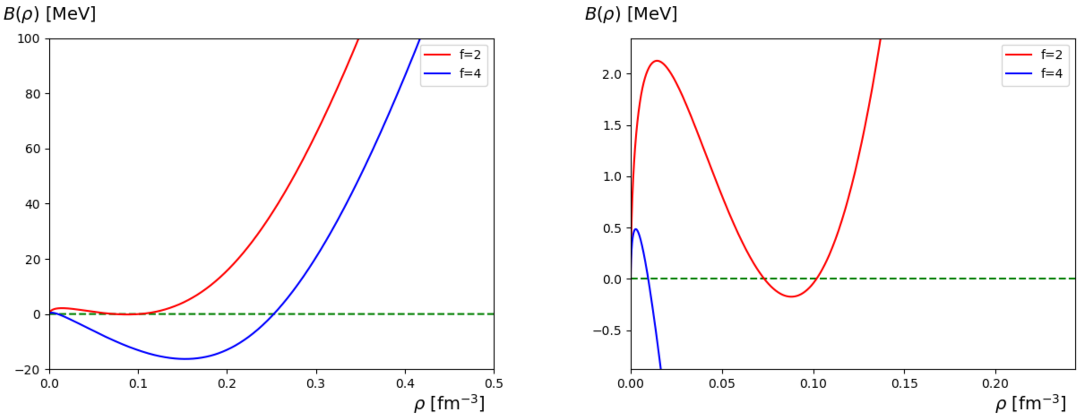


Figure 5.4: Binding energy as a function of nucleon density for neutron matter ($f = 2$) and nuclear matter ($f = 4$). Right panel is zoomed in on the in the minimum binding energy for neutron matter. Green line shows zero for reference.

However, for now, we will try to make sensible results from this simpler model.

Looking at Figure 5.4 we see that the binding energy has a negative minimum for both neutron and nuclear matter.[10] This means that there is some density where the neutron matter is bound. The pressure should then be zero at the density where the binding energy is at a minimum, that is, at the surface of the star [6, p. 194]. This then corresponds to letting the pressure vanish for all densities below this value. In other words, we should set the pressure to be zero once the saturation density is reached. However, it turns out that there is no evidence for bound matter in neutron stars today [6, p. 194-195]. Therefore, we will try other approaches to circumvent this problem.

Assuming that neutron matter is not bound, we can get rid of the ill-behaved part of the pressure in other ways. For one, we can approximate by assuming that at low energies, the interactions between the particles cease to exist. If that is the case, we can substitute the bad region with the free Fermi gas expression. Demanding that the pressure is continuous as a function of the energy density, we patch together the two solutions at the point where they coincide. This EoS is plotted in Figure 5.5. For future reference, we will call this solution the ”Fermi-σ-ω EoS”. Using this approximation, we should expect that the model predicts higher masses than the original one, since the pressure in the ill-behaved region always is lower than the pressure in the free Fermi gas.

Another way to settle this issue, is to alter the coupling constants. Having only two coupling constants, we cannot fit more than two experimental values to the EoS. There are, at least, four empirical values we wish to fit: The saturation density, the binding energy, the compressibility and the so called Landau mass defined by $m_{\text{L}} \equiv E = \sqrt{(m^\star)^2 + k_{\text{F}}^2}$ [25, p. 232]. It is then clear that the equation of state will differ, depending on which experiment we choose. This is of course not satisfactory in the search for a realistic, or at least consistent, model. However, we should remember that all calculations are done in the mean-field approximation. For small densities, that is in the non-relativistic limit, the average potential ”felt” by a particle is given by

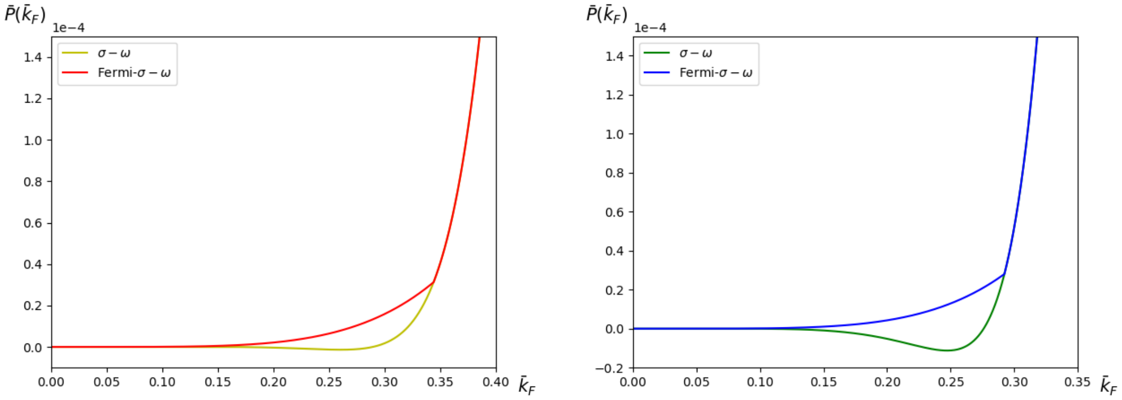
$$\langle V \rangle = \rho \int \mathrm{d}^3 r \, V. \tag{5.113}$$

Since $\rho \sim k_{\text{F}}^3$ and the average kinetic energy goes as $\sim k_{\text{F}}^2$, we must have that

$$\int \mathrm{d}^3 r \, V > 0, \tag{5.114}$$

otherwise the average energy would not be bounded from below, creating an unstable vacuum. This means that the average potential energy increases monotonically as a function of ρ. Now we see that we

[10] Of course, we already knew this about nuclear matter.



(a) Neutron matter

(b) Nuclear matter

Figure 5.5: Dimensionless pressure as a function of dimensionless Fermi momentum. The red/blue line has patched the free Fermi gas expression in the low density region together with the σ-ω EoS for higher densities. The yellow/green line represents pressure in the original σ-ω model.

have made an error in the assumptions earlier: The coupling constants in the mean field approximation should be chosen to satisfy the properties of $\langle V \rangle$, not V. We can then think of g_σ and g_ω as effective couplings that we chose in a way to mimic all the many-body interactions that we have not taken account for. One way of doing this is proposed by Machleidt, Holinde and Elstor in [41], where they fitted nucleon-nucleon phase shifts up to 300 MeV in a boson exchange model, using the π, ρ, ω and σ mesons. They then found the coupling constants

$$g_\sigma = 10.75, \qquad g_\omega = 15.85. \tag{5.115}$$

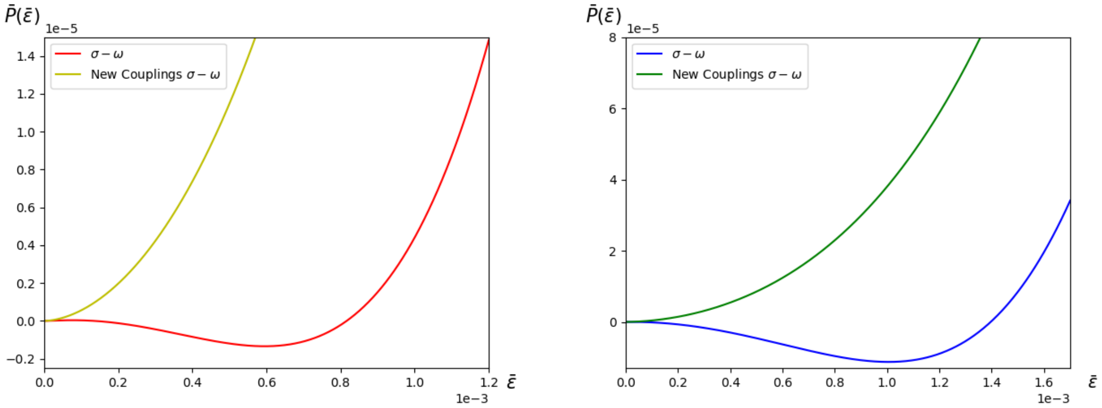
Looking at Figure 5.6 we see that by using these fine-tuned couplings, all stability issues vanishes. We will call the model with these new couplings for the "New Couplings σ-ω EoS". One should note, that the pressure is much higher in the low density region using the new couplings than the ones fitted for saturation density and binding energy. It is in any sense unreasonable to expect such a simple model to be accurate, and since the EoS is very sensitive to which experiments we choose to probe the couplings, we should already start looking for a better model.

5.7.3 The mass-radius relation

With the three equations of state obtained in the previous section (Figure 5.1[11], Figure 5.5 and Figure 5.6), a similar program to the one used in chapter 4 can be implemented to find the mass-radius relations. The result is seen in Figure 5.7. From the Figure, it is clear that fine-tuned couplings results in a significantly larger mass than the modified hybrid between the free Fermi gas and the σ-ω EoS. Further we note that even though different in shape, the limiting mass predicted for the bound matter and the Fermi σ–ω EoS, is practically the same. This is reasonable, since these two equations of state differ in the low energy region, while they coincide for larger energies. We also see that for equal central pressures, the radius of the $f = 2$ stars are in general larger than the ones with $f = 4$.

It is hard to tell if these results are good or not. As mentioned, the σ-ω model is a simple model with only two coupling constants, making it impossible to fit all empirically known values for nuclear matter. The maximum mass is calculated to be (for neutron matter) $M_{\max} = 2.99.M_\odot$ for the combined free Fermi gas and σ-ω EoS, while it yielded $M_{\max} = 3.63 M_\odot$ for the fine-tuned couplings. We note that this is substantially larger than the results obtained for the free Fermi gas. Adding this toy model for the strong force has thus increased the upper mass limit.

[11] Remember that in this case we set the pressure to be zero for all densities below the saturation density.



(a) Neutron matter

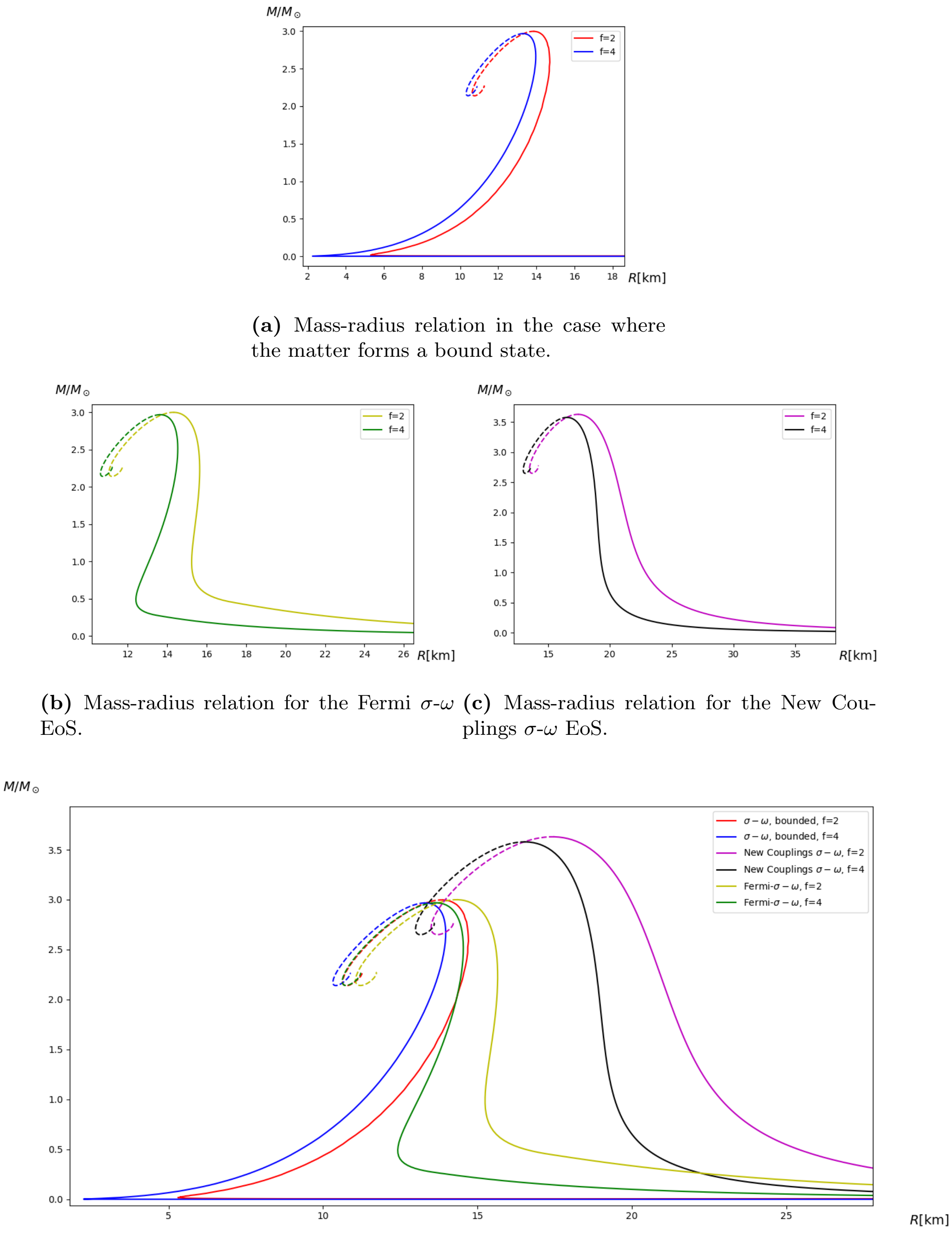
(b) Nuclear matter

Figure 5.6: Dimensionless pressure as a function of the dimensionless energy density. The red/blue line has couplings fitted to the nucleon-nucleon phase shifts according to [41], while the yellow/green line has couplings chosen to match saturation density and binding energy of nuclear matter.

We cannot falsify this model, in the sense that there has not been measured neutron stars with masses larger than this, but it is fair to assume that it is not accurate, due to the fact that we see a major difference in the limiting mass regarding which properties of nuclear matter we use to fix our coupling constants.

5.8 Summary

In this chapter we have explored the σ-ω model and the resulting EoS. We have used the mean-field approximation and assumed that the neutron and the proton are two different states of the same particle. Introducing a toy model for the strong force involving two mesons, we then found that the model predicted a maximum mass of about 3.6 solar masses. The Lagrangian we used was fairly simple and included only two interactions, making it impossible to fit all known properties of nuclear matter. The mass-radius relations is found highly dependent on the empirical values chosen to probe the coupling constants. We thus concluded that a more careful treatment is required.



(a) Mass-radius relation in the case where the matter forms a bound state.

(b) Mass-radius relation for the Fermi σ-ω EoS.

(c) Mass-radius relation for the New Couplings σ-ω EoS.

(d) Figures 5.7a, 5.7b and 5.7c shown in the same figure.

Figure 5.7: Mass-radius relations for the three equations of state discussed in the text. Dotted lines represent unstable solutions.

Chapter 6

Improving the σ–ω model

Having introduced the σ–ω model, we now have laid the foundation for a more realistic description of the equation of state for a neutron star. We are in particular interested in increasing the number of degrees of freedom, so that we can fit more of the observed properties of nuclear matter. As mentioned, the empirical values that we want to fix are the binding energy, the saturation density, the Landau mass and the compressibility. In addition, we want to consider a system where there is a difference in the neutron and proton densities. This will result in a change in the energy, which again gives rise to a symmetry-restoring force. We call this force the isospin force, and we will assume that it is mediated by a charged vector meson ρ_i^μ. This allows us to fix yet another experimental value, namely the symmetry-energy coefficient a_s.

We are still working in the relativistic mean-field approximation, but we will add more interactions and particles. In particular, we will include cubic and quartic self-interactions for the σ-meson, as well as an interaction term between the ρ-meson and the conserved isospin current. We will also see that it is required that any macroscopic object must be globally charge neutral. To enforce this, we include electrons and muons.

6.1 σ self-interactions

We want find a way to fix the Landau mass and the compression modulus. Here we follow the procedure first proposed by Boguta and Bodmer [42]. This involves adding cubic and quartic self-interaction terms for the scalar meson σ to the Lagrangian

$$\mathscr{L}_{\sigma\,\mathrm{self}} = \frac{1}{3}bm\big(g_\sigma\sigma\big)^3 + \frac{1}{4}c\big(g_\sigma\sigma\big)^4. \tag{6.1}$$

Here b and c are constants, while m is just a number with dimension mass, so that b becomes dimensionless. It is convention to set m to be the nucleon mass 939 MeV, but we could of course have chosen a completely arbitrary number since the product bm is constant.

The new terms in the Lagrangian do only affect the equation of motion for the σ-field. We thus focus on the part of the Lagrangian containing only σ-terms

$$\mathscr{L}_\sigma = \frac{1}{2}\big(\partial_\mu\sigma\big)\big(\partial^\mu\sigma\big) - \frac{1}{2}m_\sigma^2\sigma^2 + g_\sigma\sigma\bar{\psi}\psi + \frac{1}{3}bm\big(g_\sigma\sigma\big)^3 + \frac{1}{4}c\big(g_\sigma\sigma\big)^4. \tag{6.2}$$

Keep in mind that we have not said anything about the sign of the new terms; b and c can be both positive or negative. However, we should hope that the sign of c is negative to make sure that the energy is bounded from below, otherwise we would encounter an unstable vacuum.

Using the Euler-Lagrange equations, we find

$$\frac{\partial\mathscr{L}_\sigma}{\partial\sigma} = -m_\sigma^2\sigma + g_\sigma\bar{\psi}\psi + bmg_\sigma^3\sigma^2 + cg_\sigma^4\sigma^3 = \partial_\mu\frac{\partial\mathscr{L}_\sigma}{\partial(\partial_\mu\sigma)} = \partial_\mu\big(\partial^\mu\sigma\big) = \Box\,\sigma, \tag{6.3}$$

which can be rewritten as

$$g_\sigma \bar{\psi}\psi = \big(\Box + m_\sigma^2 - bmg_\sigma^3\sigma - cg_\sigma^4\sigma^2\big)\sigma. \tag{6.4}$$

The mean field $\langle\sigma\rangle$ is independent of the space-time coordinate x^μ. Thus we find that in the mean-field approximation, this equation becomes

$$g_\sigma \bar{\psi}\psi = \big(m_\sigma^2 - bmg_\sigma^3\langle\sigma\rangle - cg_\sigma^4\langle\sigma\rangle^2\big)\langle\sigma\rangle. \tag{6.5}$$

Since the right-hand side is only dependent on ground state expectation values, so must also the left-hand side. Thus we can rewrite this as

$$g_\sigma \langle\bar{\psi}\psi\rangle \equiv g_\sigma\rho_{\mathrm{s}} = \big(m_\sigma^2 - bmg_\sigma^3\langle\sigma\rangle - cg_\sigma^4\langle\sigma\rangle^2\big)\langle\sigma\rangle. \tag{6.6}$$

Note that since the terms we have added to the Lagrangian so far only contains the σ-field, the expression for the scalar density ρ_{s} is still given by (5.74) and (5.79). This also means that the energy spectrum and the effective mass have the same expressions as in the σ–ω model,

$$E = \sqrt{k_{\mathrm{F}}^2 + (m^\star)^2}, \tag{6.7}$$

$$m^\star = m - g_\sigma\langle\sigma\rangle, \tag{6.8}$$

we just have to remember that $\langle\sigma\rangle$ is given by (6.6). As we saw in the previous chapter, the terms containing solely meson fields contribute a factor βV times the mean of the Lagrangian. This means that the self-interacting terms add an extra factor $\exp\big\{-\beta V\big[\frac{1}{3}bm(g_\sigma\sigma)^3 + \frac{1}{4}c(g_\sigma\sigma)^4\big]\big\}$ to the partition function so that

$$Z = \mathrm{e}^{-\beta V\left[\frac{1}{3}bm(g_\sigma\langle\sigma\rangle)^2 + \frac{1}{4}c(g_\sigma\langle\sigma\rangle)\right]} Z_{\sigma\text{-}\omega}. \tag{6.9}$$

Here $Z_{\sigma\text{-}\omega}$ is the partition function for the σ–ω model given by (5.43). Taking the logarithm of Z, we find that the only difference in the pressure and the energy density due to the included self-interaction terms in the Lagrangian, is exactly these terms. We have thus found that

$$P = P_{\sigma\text{-}\omega} + \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 + \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4, \tag{6.10}$$

$$\epsilon = \epsilon_{\sigma\text{-}\omega} - \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 - \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4, \tag{6.11}$$

where $P_{\sigma\text{-}\omega}$ and $\epsilon_{\sigma\text{-}\omega}$ are given by (5.82) and (5.84) respectively.

It is not obvious from equation (6.6) that the scalar density ρ_{s} is the same as before. This we can now easily check. Demanding that the star is in equilibrium, we have[1]

$$0 = \frac{\partial P}{\partial m^\star} = \frac{\partial}{\partial m^\star}\Big[P_{\sigma\text{-}\omega} + \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 + \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4\Big] = \frac{\partial P_{\mathrm{FG}}}{\partial m^\star} + \frac{m_\sigma^2\langle\sigma\rangle}{g_\sigma} - bm\big(g_\sigma\langle\sigma\rangle\big)^2 - c\big(g_\sigma\langle\sigma\rangle\big)^3. \tag{6.12}$$

Inserted into (6.6), we then find

$$\rho_{\mathrm{s}} = -\frac{\partial P_{\mathrm{FG}}}{\partial m^\star}, \tag{6.13}$$

which is consistent with (5.79).

6.2 The isospin force

We are now ready to allow the proton and neutron densities to vary. This will result in a change in the binding energy, and this energy shift we will call the symmetry-energy [43]. The symmetry energy gives rise to a force which we will call the isospin force. This force manifest it self as an interaction term in

[1] Note that when taking derivatives of the term $bm\big(g_\sigma\langle\sigma\rangle\big)^3$ with respect to $m^\star$, we look at the product bm as a constant.

the Lagrangian which must involve the conserved isospin current j_i^μ from (5.47). Since we require that the Lagrangian is a scalar, the current j_i^μ must couple to a charged vector meson. By this we mean a vector meson with three isospin states. This is analogues to the nucleon which is a particle with two isospin states. Here we use the ρ_i^μ-meson, which represents an isospin triplet $\boldsymbol{\rho^\mu} = (\rho_1^\mu, \rho_2^\mu, \rho_3^\mu)$ [6, p. 183].

Each component of the triplet behaves as a vector boson, and thus the free Lagrangian is on the same form as it was for the ω-meson (5.6). We therefore write

$$\mathscr{L}_{\rho\,\mathrm{free}} = -\frac{1}{4}\rho_{i\mu\nu}\rho_i^{\mu\nu} + \frac{1}{2}m_\rho^2\rho_{i\mu}\rho_i^\mu, \tag{6.14}$$

where we have defined

$$\rho_{i\mu\nu} = \partial_\mu\rho_{i\nu} - \partial_\nu\rho_{i\mu}. \tag{6.15}$$

It turns out that this Lagrangian is invariant under the infinitesimal transformation[2] [6, p. 158]

$$\rho_{i\mu} \to \rho'_{i\mu} = \rho_{i\mu} - \epsilon_{ijk}\Lambda_j\rho_{k\mu}, \tag{6.16}$$

where ϵ_{ijk} is the 3-dimensional Levi-Civita-tensor defined by (C.11). We can show this by insertion:

$$\begin{aligned}
\mathscr{L}'_{\rho\,\mathrm{free}} &= -\frac{1}{4}\Big[\partial_\mu\big(\rho_{i\nu} - \epsilon_{ijk}\Lambda_j\rho_{k\nu}\big) - \partial_\nu\big(\rho_{i\mu} - \epsilon_{ilm}\Lambda_l\rho_{m\mu}\big)\Big]\Big[\partial^\mu\big(\rho_i^\nu - \epsilon_{ino}\Lambda_n\rho_o^\nu\big) - \partial^\nu\big(\rho_i^\mu - \epsilon_{ipq}\Lambda_p\rho_q^\mu\big)\Big] \\
&\qquad + \frac{1}{2}m_\rho^2\big(\rho_{i\mu} - \epsilon_{ijk}\Lambda_j\rho_{k\mu}\big)\big(\rho_i^\mu - \epsilon_{ilm}\Lambda_l\rho_m^\mu\big) \\
&= -\frac{1}{4}\Big[\big(\partial_\mu\rho_{i\nu} - \partial_\nu\rho_{i\mu}\big) - \big(\epsilon_{ijk}\Lambda_j\partial_\mu\rho_{k\nu} - \epsilon_{ilm}\Lambda_l\partial_\nu\rho_{m\mu}\big)\Big]\Big[\big(\partial^\mu\rho_i^\nu - \partial^\nu\rho_i^\mu\big) - \big(\epsilon_{ino}\Lambda_n\partial^\mu\rho_o^\nu - \epsilon_{ipq}\Lambda_p\partial^\nu\rho_q^\mu\big)\Big] \\
&\qquad + \frac{1}{2}m_\rho^2\Big[\rho_{i\mu}\rho_i^\mu - \epsilon_{ilm}\Lambda_l\rho_{i\mu}\rho_m^\mu - \epsilon_{ijk}\Lambda_j\rho_{k\mu}\rho_i^\mu - \mathcal{O}\big(\Lambda^2\big)\Big] \\
&= -\frac{1}{4}\Big[\rho_{i\mu\nu}\rho_i^{\mu\nu} - \big(\partial_\mu\rho_{i\nu} - \partial_\nu\rho_{i\mu}\big)\big(\epsilon_{ino}\Lambda_n\partial^\mu\rho_o^\nu - \epsilon_{ipq}\Lambda_p\partial^\nu\rho_q^\mu\big) \\
&\qquad - \big(\partial^\mu\rho_i^\nu - \partial^\nu\rho_i^\mu\big)\big(\epsilon_{ijk}\Lambda_j\partial_\mu\rho_{k\nu} - \epsilon_{ilm}\Lambda_l\partial_\nu\rho_{m\mu}\big) + \mathcal{O}\big(\Lambda^2\big)\Big] \\
&\qquad + \frac{1}{2}m_\rho^2\Big[\rho_{i\mu}\rho_i^\mu - \epsilon_{ilm}\Lambda_l\rho_{i\mu}\rho_m^\mu + \epsilon_{kji}\Lambda_j\rho_{k\mu}\rho_i^\mu + \mathcal{O}\big(\Lambda^2\big)\Big] \\
&= -\frac{1}{4}\Big[\rho_{i\mu\nu}\rho_i^{\mu\nu} - \epsilon_{ino}\Lambda_n\partial_\mu\rho_{i\nu}\partial^\mu\rho_o^\nu + \epsilon_{ino}\Lambda_n\partial_\nu\rho_{i\mu}\partial^\mu\rho_o^\nu + \epsilon_{ipq}\Lambda_p\partial_\mu\rho_{i\nu}\partial^\nu\rho_q^\mu - \epsilon_{ipq}\Lambda_p\partial_\nu\rho_{i\mu}\partial^\nu\rho_q^\mu \\
&\qquad - \epsilon_{ijk}\Lambda_j\partial^\mu\rho_i^\nu\partial_\mu\rho_{k\nu} + \epsilon_{ijk}\Lambda_j\partial^\nu\rho_i^\mu\partial_\mu\rho_{k\nu} - \epsilon_{ilm}\Lambda_l\partial^\mu\rho_i^\nu\partial_\nu\rho_{m\mu} + \epsilon_{ilm}\Lambda_l\partial^\nu\rho_i^\mu\partial_\nu\rho_{m\nu}\Big] \\
&\qquad + \frac{1}{2}m_\rho^2\rho_{i\mu}\rho_i^\mu + \mathcal{O}\big(\Lambda^2\big) \\
&= -\frac{1}{4}\Big[\rho_{i\mu\nu}\rho_i^{\mu\nu} - \epsilon_{ino}\Lambda_n\partial_\mu\rho_{i\nu}\partial^\mu\rho_o^\nu + \epsilon_{ino}\Lambda_n\partial_\nu\rho_{i\mu}\partial^\mu\rho_o^\nu + \epsilon_{ipq}\Lambda_p\partial_\mu\rho_{i\nu}\partial^\nu\rho_q^\mu - \epsilon_{ipq}\Lambda_p\partial_\nu\rho_{i\mu}\partial^\nu\rho_q^\mu \\
&\qquad + \epsilon_{kji}\Lambda_j\partial^\mu\rho_i^\nu\partial_\mu\rho_{k\nu} - \epsilon_{kji}\Lambda_j\partial^\nu\rho_i^\mu\partial_\mu\rho_{k\nu} - \epsilon_{mli}\Lambda_l\partial^\mu\rho_i^\nu\partial_\nu\rho_{m\mu} + \epsilon_{mli}\Lambda_l\partial^\nu\rho_i^\mu\partial_\nu\rho_{m\nu}\Big] \\
&\qquad + \frac{1}{2}m_\rho^2\rho_{i\mu}\rho_i^\mu + \mathcal{O}\big(\Lambda^2\big) \\
&= -\frac{1}{4}\rho_{i\mu\nu}\rho_i^{\mu\nu} + \frac{1}{2}\rho_{i\mu}\rho_i^\mu + \mathcal{O}\big(\Lambda^2\big) \\
&= \mathscr{L}_{\rho\,\mathrm{free}} + \mathcal{O}\big(\Lambda^2\big).
\end{aligned} \tag{6.17}$$

Since the transformation is infinitesimal, the second order term in Λ vanishes, and we obtain the original Lagrangian.

[2] Just to be clear, the sign of the transformation is arbitrary. The Lagrangian is of course also invariant if we write $\rho'_{i\mu} \to \rho_{i\mu} + \epsilon_{ijk}\Lambda_j\rho_{k\mu}$

The transformation $\rho_{i\mu} \to \rho'_{i\mu}$ has the variation

$$\delta\rho_{i\mu} = -\sum_j \epsilon_{ijk}\rho_{k\mu}, \tag{6.18}$$

and the corresponding conserved isospin current

$$\begin{aligned}
J_i^\mu &= \frac{\partial \mathscr{L}_{\rho\,\text{free}}}{\partial(\partial_\mu \rho_{h\nu})}\big(-\epsilon_{hij}\rho_{j\nu}\big) \\
&= \frac{1}{4}\epsilon_{hij}\rho_{j\nu}\frac{\partial}{\partial(\partial_\mu\rho_{h\nu})}\Big[\big(\partial_\alpha\rho_{k\beta} - \partial_\beta\rho_{k\alpha}\big)\big(\partial^\alpha\rho_k^\beta - \partial^\beta\rho_k^\alpha\big)\Big] \\
&= \frac{1}{4}\epsilon_{hij}\rho_{j\nu}\Big[\big(\delta^\mu_\alpha\delta^\nu_\beta - \delta^\mu_\beta\delta^\nu_\alpha\big)\big(\partial^\alpha\rho_k^\beta - \partial^\beta\rho_k^\alpha\big) + \big(\eta^{\alpha\mu}\eta^{\beta\nu} - \eta^{\beta\mu}\eta^{\alpha\nu}\big)\big(\partial_\alpha\rho_{k\beta} - \partial_\beta\rho_{k\alpha}\big)\Big]\delta_{kh} \\
&= \frac{1}{4}\epsilon_{hij}\rho_{j\nu}\Big[\big(\partial^\mu\rho_h^\nu - \partial^\nu\rho_h^\mu\big) - \big(\partial^\nu\rho_h^\mu - \partial^\mu\rho_h^\nu\big) + \big(\partial^\mu\rho_h^\nu - \partial^\nu\rho_h^\mu\big) - \big(\partial^\nu\rho_h^\mu - \partial^\mu\rho_h^\nu\big)\Big] \\
&= \epsilon_{hij}\rho_{j\nu}\rho_h^{\mu\nu} \\
&= \epsilon_{ijh}\rho_{j\nu}\rho_h^{\mu\nu}.
\end{aligned} \tag{6.19}$$

Note that if one prefers, it is possible to write this in vector notation as

$$\boldsymbol{J}^\mu = \boldsymbol{\rho}_\nu \times \boldsymbol{\rho}^{\mu\nu}. \tag{6.20}$$

Using (6.19) and (5.47) we obtain the total conserved isospin current

$$I_i^\mu = j_i^\mu + J_i^\mu = \frac{1}{2}\bar{\psi}\gamma^\mu\tau_i\psi + \epsilon_{ijk}\rho_{j\nu}\rho_k^{\mu\nu}. \tag{6.21}$$

However, when we couple $\rho_{i\mu}$ to this current, we obtain a term in the Lagrangian that contains derivatives of ρ_i^μ. Since derivatives contributes to conserved currents through Noether's theorem, introducing an interaction term[3] $-g_\rho\rho_{i\mu}I_i^\mu$ will also give a contribution to the conserved isospin current. Thus we must make the substitution

$$\begin{aligned}
I_i^\mu \to \big(I_i^\mu\big)' &= I_i^\mu + \frac{\partial(-g_\rho\rho_{k\alpha}I_k^\alpha)}{\partial(\partial_\mu\rho_{h\nu})}\big(-\epsilon_{hij}\rho_{j\nu}\big) \\
&= I_i^\mu + g_\rho\epsilon_{hij}\epsilon_{klm}\rho_{j\nu}\rho_{l\beta}\rho_{k\alpha}\frac{\partial}{\partial(\partial_\mu\rho_{h\nu})}\big(\partial^\alpha\rho_m^\beta - \partial^\beta\rho_m^\alpha\big) \\
&= I_i^\mu + g_\rho\epsilon_{hij}\epsilon_{klm}\rho_{j\nu}\rho_{l\beta}\rho_{k\alpha}\big(\eta^{\alpha\mu}\eta^{\beta\nu} - \eta^{\beta\mu}\eta^{\alpha\nu}\big)\delta_{hm} \\
&= I_i^\mu + g_\rho\epsilon_{hij}\rho_{j\nu}\big(\epsilon_{klh}\rho_k^\mu\rho_l^\nu - \epsilon_{klh}\rho_k^\mu\rho_l^\nu\big) \\
&= I_i^\mu + g_\rho\epsilon_{hij}\rho_{j\nu}\big(-\epsilon_{hlk}\rho_k^\mu\rho_l^\nu - \epsilon_{hkl}\rho_k^\mu\rho_l^\nu\big) \\
&= \frac{1}{2}\bar{\psi}\gamma^\mu\tau_i\psi + \epsilon_{ijk}\rho_{j\nu}\rho_k^{\mu\nu} + 2g_\rho\epsilon_{ihj}\epsilon_{hlk}\rho_{j\nu}\rho_l^\nu\rho_k^\mu.
\end{aligned} \tag{6.22}$$

Again, it is possible to write this in vector notation as

$$\big(I_i^\mu\big)' = \frac{1}{2}\bar{\psi}\gamma^\mu\boldsymbol{\tau}\psi + \boldsymbol{\rho}_\nu \times \boldsymbol{\rho}^{\mu\nu} + 2g_\rho\big(\boldsymbol{\rho}^\nu \times \boldsymbol{\rho}^\mu\big) \times \boldsymbol{\rho}_\nu, \tag{6.23}$$

where we have defined $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3)$.

Consider now the free Lagrangian for the ρ-meson

$$\mathscr{L}_{\rho\,\text{free}} = -\frac{1}{4}\rho_{i\mu\nu}\rho_i^{\mu\nu} + \frac{1}{2}m_\rho^2\rho_{i\mu}\rho_i^\mu. \tag{6.24}$$

[3] We remind the reader that the sign of the interaction term is completely arbitrary. The sign of the coupling constant g_ρ will change depending on our choice, so that the overall sign of the interaction term always is the same.

We then see that if we write the charged ρ-fields in terms of the isospin ladder operators[4] ρ_+^μ and ρ_-^μ, that is

$$\rho_1^\mu = \frac{1}{\sqrt{2}}\big(\rho_-^\mu + \rho_+^\mu\big), \tag{6.25}$$

$$\rho_2^\mu = \frac{\mathrm{i}}{\sqrt{2}}\big(\rho_-^\mu - \rho_+^\mu\big), \tag{6.26}$$

the Lagrangian becomes

$$\begin{aligned}\mathscr{L}_{\rho\,\mathrm{free}} = -\frac{1}{4}\bigg[&\frac{1}{2}\big(\rho_{-\mu\nu}\rho_-^{\mu\nu} + 2\rho_{+\mu\nu}\rho_-^{\mu\nu} + \rho_{+\mu\nu}\rho_+^{\mu\nu}\big) + \frac{1}{2}m_\rho^2\big(\rho_{-\mu\nu}\rho_-^{\mu\nu} - 2\rho_{+\mu\nu}\rho_-^{\mu\nu} + \rho_{+\mu\nu}\rho_+^{\mu\nu}\big) + \rho_{3\mu\nu}\rho^{\mu\nu}\bigg] \\ &+ \frac{1}{2}m_\rho^2\bigg[\frac{1}{2}\big(\rho_{-\mu}\rho_-^{\mu} + 2\rho_{+\mu}\rho_-^{\mu} + \rho_{+\mu}\rho_+^{\mu}\big) + \frac{1}{2}\big(\rho_{-\mu}\rho_-^{\mu} - 2\rho_{+\mu}\rho_-^{\mu} + \rho_{+\mu}\rho_+^{\mu}\big) + \rho_{3\mu}\rho^{\mu}\bigg] \\ = -\frac{1}{4}&\big(\rho_{-\mu\nu}\rho_-^{\mu\nu} + \rho_{+\mu\nu}\rho_+^{\mu\nu} + \rho_{3\mu\nu}\rho^{\mu\nu}\big) + \frac{1}{2}m_\rho^2\big(\rho_{-\mu}\rho_-^{\mu} + \rho_{+\mu}\rho_+^{\mu} + \rho_{3\mu}\rho^{\mu}\big) \\ = -\frac{1}{4}&\rho'_{i\mu\nu}\big(\rho_i^{\mu\nu}\big)' + \frac{1}{2}m_\rho^2\rho'_{i\mu}\big(\rho_i^\mu\big)',\end{aligned} \tag{6.27}$$

where we have defined $(\rho_i^\mu)' = (\rho_-^\mu, \rho_+^\mu, \rho_3^\mu)$. The Lagrangian is thus of the same form as (6.24), which means that it describes the same physics. The isospin ladder-operators $\rho_\pm$ increase or lower the third component of the isospin for a particle. For instance, if we let ρ_- work on a neutron, which has total isospin 1/2 with a third component yielding 1/2, then the third component changes to $-1/2$ and the particle becomes a proton

$$\rho_-\psi_{\mathrm{n}} = \psi_{\mathrm{p}}. \tag{6.28}$$

In other words, if $\rho_\pm$ is working on a nucleon field, the numbers of neutrons and protons will change. However, in the mean-field approximation, we assume that the system is in its ground state. Thus we expect that the proton and neutron densities are constants, and therefore we must have that $\langle\rho_\pm^\mu\rangle = 0$. The only way this is possible, is if the expectation value of both fields ρ_1^μ and ρ_2^μ vanish.

We can now find the equation of motion for the ρ-field. Since only $\langle\rho_3^\mu\rangle$ is non-zero, we have that the last term in the conserved isospin current (6.22) vanishes, because we obtain a cross product between ρ_i^μ and ρ_i^ν, and they are obviously parallel. The equation of motion for the ρ-field is then

$$\frac{\partial}{\partial\rho_i^\mu}\Big(\frac{1}{2}m_\rho^2\rho_{3\alpha}\rho_3^\alpha - \frac{1}{4}\rho_{3\alpha\beta}\rho_3^{\alpha\beta} - \frac{1}{2}g_\rho\rho_{3\alpha}\bar{\psi}\gamma^\alpha\tau_3\psi - g_\rho\epsilon_{3jk}\rho_{j\beta}\rho_k^{\alpha\beta}\Big) = m_\rho^2\rho_3^\mu - \frac{1}{2}g_\rho\bar{\psi}\gamma^\mu\tau_3\psi + \text{derivatives} = 0. \tag{6.29}$$

Introducing the mean-field approximation, all derivatives vanish and the equation of motion for the ρ-field becomes

$$m_\rho^2\langle\rho_3^\mu\rangle - \frac{1}{2}g_\rho\bar{\psi}\gamma^\mu\tau_3\psi = 0. \tag{6.30}$$

Since these two terms must be equal, we have that the current $g_\rho\bar{\psi}\gamma^\mu\tau_3\psi$ is given only by ground state expectation values. Hence the current must also be a ground state expectation value, so we write

$$m_\rho^2\langle\rho_3^\mu\rangle = \frac{1}{2}g_\rho\langle\bar{\psi}\gamma^\mu\tau_3\psi\rangle. \tag{6.31}$$

Having assumed that there is rotational symmetry at each point in space, there should be no preferred direction for the expectation value of the current $\langle\bar{\psi}\gamma^\mu\tau_3\psi\rangle$. Thus, all spatial components must be zero. We may then simplify even further

$$\langle\rho_i^\mu\rangle = \begin{cases}\langle\rho_3^0\rangle = \frac{g_\rho}{2m_\rho^2}\langle\bar{\psi}\gamma^0\tau_3\psi\rangle = \frac{g_\rho}{2m_\rho^2}\langle\psi^\dagger\tau_3\psi\rangle = \frac{g_\rho}{2m_\rho^2}(\rho_{\mathrm{n}} - \rho_{\mathrm{p}}), & \text{for } i = 3, \mu = 0, \\ 0, & \text{otherwise,}\end{cases} \tag{6.32}$$

[4] One may ask why we can write operators in terms of fields. The answer is that in field theory, the fields are actually field *operators*, but we just call them fields because we are lazy.

where ρ_{n} and ρ_{p} denotes the neutron and the proton density, respectively.

Now that we have the ground state expectation values for the meson fields, we can compute the equation of motion for the nucleon field

$$\begin{aligned}0 =& \frac{\partial}{\partial\bar{\psi}}\Big[\bar{\psi}(\mathrm{i}\not\partial - m)\psi + g_\sigma\langle\sigma\rangle\bar{\psi}\psi - g_\omega\langle\omega_0\rangle\bar{\psi}\gamma^0\psi - g_\rho\langle\rho_{30}\rangle I_3^0\Big]\\ =& \Big(\mathrm{i}\not\partial - m\Big)\psi + g_\sigma\langle\sigma\rangle\psi - g_\omega\langle\omega_0\rangle\gamma^0\psi - g_\rho\langle\rho_{30}\rangle\frac{\partial}{\partial\bar{\psi}}\Big(\frac{1}{2}\bar{\psi}\gamma^0\tau_3\psi\Big)\\ =& \Big[\gamma^0\Big(\mathrm{i}\partial_0 - g_\omega\langle\omega_0\rangle - \frac{1}{2}g_\rho\langle\rho_{30}\rangle\tau_3\Big) + \mathrm{i}\gamma^i\partial_i - \Big(m - g_\sigma\langle\sigma\rangle\Big)\Big]\psi. \qquad (6.33)\end{aligned}$$

In the same manner as in (5.28), we Fourier transform this expression to obtain

$$\Big[\gamma^0\Big(k_0 - g_\omega\langle\omega_0\rangle - \frac{1}{2}g_\rho\langle\rho_{30}\rangle\tau_3\Big) + \gamma^i k_i - \Big(m - g_\sigma\langle\sigma\rangle\Big)\Big]\psi(k^\mu). \qquad (6.34)$$

If we define the momenta

$$K^\mu = k^\mu - g_\omega\langle\omega^\mu\rangle - \frac{1}{2}g_\rho\langle\rho_3^\mu\rangle\tau_3, \qquad (6.35)$$

and the effective mass

$$m^\star = m - g_\sigma\langle\sigma\rangle, \qquad (6.36)$$

we find

$$\Big(\not K - m^\star\Big)\psi = 0. \qquad (6.37)$$

The nucleon field can thus be represented by a free field with the same effective mass as in the σ–ω model, but with a shifted field-momenta. The energy spectrum is still the same as before

$$E(\boldsymbol{k}) \equiv K_0(\boldsymbol{k}) = \sqrt{\boldsymbol{k}^2 + (m^\star)^2}. \qquad (6.38)$$

6.3 Charge neutrality

The last property that we want to incorporate in this model is the fact that any macroscopic object in equilibrium must be globally, electrically charge neutral. This is because in equilibrium, the energy is minimized. The lowest possible energy contribution from the Coulomb forces is then when the total charge of the object is zero. This does not mean that each point inside the object must be charge neutral. There are possible configurations which are globally charge neutral, but still have finite charge distributions, that are energetically favourable. Some examples are the atom or a molecule. Both have a vanishing net charge, but if we examine them closer we find regions of positive and negative charges.

As we have mentioned before, free neutrons are unstable and decay through the process [5]

$$\mathrm{n} \rightarrow \mathrm{p} + \mathrm{e} + \bar{\nu}_{\mathrm{e}}, \qquad (6.39)$$

where n, p, e and ν_{e} denote the neutron, the proton, the electron and the electron-neutrino respectively. It is then natural to introduce the electrons e to ensure global charge neutrality. However, at high densities we also expect that there is a presence of leptons with higher mass. We therefore also include the negative charged μ, since their main decay mode is

$$\mu \rightarrow \mathrm{e} + \bar{\nu}_{\mathrm{e}} + \nu_\mu. \qquad (6.40)$$

Neutrinos have a negligible mass compared to all other particles in this model. We will therefore as an approximation assume that they do not contribute much to the energy density, and forget them.

[5] All decay modes can be found in the "Particle Physics Booklet" from the Particle Data Group.

To ensure our system is charge neutral, we then imagine a charged background field of electrons and muons. That is, we neglect their interactions, and assume that they are free fields

$$\mathscr{L}_{\text{leptons}} = \sum_{i=\text{e},\mu} \bar{\psi}_i \big(\mathrm{i}\partial\!\!\!/ - m\big)\psi_i. \tag{6.41}$$

Since both μ^- and e^- have the opposite sign of the proton, global charge neutrality yields that the electron and the muon density add up to the proton density

$$\rho_\text{e} + \rho_\mu = \rho_\text{p}. \tag{6.42}$$

Be aware that whenever ρ is written with only one subscript without boldface, then ρ_i refers to a density for a field of type i and not the ρ-field.

Since the star is static, we must also have β-stability. By that we mean that the decay (6.39) is equally likely as the inverse process

$$\text{p} + \text{e} + \bar{\nu}_\text{e} \to \text{n}. \tag{6.43}$$

Demanding β-stability is then the same as writing

$$\mu_\text{n} = \mu_\text{p} + \mu_\text{e}. \tag{6.44}$$

Furthermore, we expect that muon decay is in chemical equilibrium. That is, the processes

$$\mu^- \longleftrightarrow \text{e}^- + \bar{\nu}_\text{e} + \nu_\mu, \tag{6.45}$$

are both equally likely. However, since the muon has a (much) larger mass than the electron, the inverse muon decay is impossible unless

$$\boldsymbol{k}_\text{e}^2 > m_\mu^2 - m_\text{e}^2. \tag{6.46}$$

This means that muons do not appear at low densities. Then we write

$$\mu_\mu = \mu_\text{e}\Theta\big(\boldsymbol{k}_\text{e}^2 - m_\mu^2 + m_\text{e}^2\big), \tag{6.47}$$

where $\Theta(x)$ is the Heaviside step function.

Now we have all the constraints needed to develop a consistent equation of state. We proceed in the following by calculating the partition function.

6.4 The partition function and the equation of state

We have in this chapter included self-interactions for the σ-meson, as well as the isospin force mediated by the ρ-meson. We have also added a charge neutrality condition by introducing free electrons and muons. It is now time to put it all together and find the partition function.

The total Lagrangian for this system is in Euclidean space given by[6]

$$\mathscr{L} = \mathscr{L}_{\text{baryon}} + \mathscr{L}_{\text{lepton}}, \tag{6.48}$$

with

$$\mathscr{L}_{\text{baryon}} = \frac{1}{2}\big(\partial_\mu\sigma\big)^2 + \frac{1}{4}\omega_{\mu\nu}^2 + \frac{1}{4}\rho_{i\mu}^2 + \frac{1}{2}m_\sigma^2\sigma^2 - \frac{1}{2}m_\omega^2\omega_\mu^2 - \frac{1}{2}m_\rho^2\rho_{i\mu}^2 \tag{6.49}$$

$$- \frac{1}{3}bm\big(g_\sigma\sigma\big)^3 - \frac{1}{4}c\big(g_\sigma\sigma\big)^4 + \bar{\psi}\big(\partial\!\!\!/ + m - g_\sigma\sigma + g_\omega\omega_\mu\gamma_0 + \frac{1}{2}g_\rho\rho_{i\mu}\tau_i\gamma_0\big)\psi, \tag{6.50}$$

$$\mathscr{L}_{\text{lepton}} = \sum_{i=\text{e},\mu} \bar{\psi}_i\big(\partial\!\!\!/ + m_i\big)\psi_i. \tag{6.51}$$

[6] As before, once introduced, we assume Euclidean space and corresponding γ-matrices throughout the chapter without denoting the subscript E.

Then we add the chemical potentials so that

$$\mathscr{L}_{\text{baryon}} \to \mathscr{L}_{\text{baryon}} - \psi^\dagger \mu \psi, \tag{6.52}$$

$$\mathscr{L}_{\text{lepton}} \to \mathscr{L}_{\text{lepton}} - \psi^\dagger \psi \mu_{\text{e}} - \psi^\dagger \psi \mu_{\mu}. \tag{6.53}$$

where we have defined the μ-matrix as

$$\begin{pmatrix} \mu_{\text{n}} & 0 \\ 0 & \mu_{\text{p}} \end{pmatrix}. \tag{6.54}$$

Firstly, we calculate the nucleon part of the partition function. By expanding the action around the expectation values of the fields just as we did in (5.38)-(5.43) we obtain

$$Z_{\text{baryon}} = Z_{\text{nucleon}} Z_{\text{meson}}, \tag{6.55}$$

with

$$Z_{\text{nucleon}} = \int \mathcal{D}\mathrm{i}\psi^\dagger \mathcal{D}\psi \, \mathrm{e}^{\sum_{n,\boldsymbol{k}} \mathrm{i}\psi^\dagger_{n,\boldsymbol{k}} \left[-\omega_n - \gamma^0 \boldsymbol{\gamma}\cdot\boldsymbol{k} + \mathrm{i}\gamma^0 m^\star - \mathrm{i}(\mu - g_\omega \langle\omega_0\rangle) + \frac{1}{2}\mathrm{i} g_\rho \langle\rho_{30}\rangle \tau_3 \right] \psi_{n,\boldsymbol{k}}}, \tag{6.56}$$

$$Z_{\text{meson}} = \mathrm{e}^{\beta V \left[-\frac{1}{2} m_\sigma^2 \langle\sigma\rangle^2 + \frac{1}{2} m_\omega^2 \langle\omega_0\rangle^2 + \frac{1}{2} m_\rho^2 \langle\rho_{30}\rangle^2 + bm(g_\sigma\langle\sigma\rangle)^3 + c(g_\sigma\langle\sigma\rangle)^4 \right]}. \tag{6.57}$$

According to (4.68), the logarithm of the nucleon part can be rewritten as

$$\ln Z_{\text{nucleon}} = \ln\det\Big[-\omega_n - \gamma^0\boldsymbol{\gamma}\cdot\boldsymbol{k} + \mathrm{i}\gamma^0 m^\star - \mathrm{i}\big(\mu - g_\omega\langle\omega_0\rangle\big) + \frac{1}{2}\mathrm{i}g_\rho\langle\rho_{30}\rangle\tau_3 \Big]. \tag{6.58}$$

We have calculated similar determinants earlier, and the steps are almost identical. If we define the variables

$$\mu^\star_{\text{n}} = \mu_{\text{n}} - g_\omega\langle\omega_0\rangle - \frac{1}{2}g_\rho\langle\rho_{30}\rangle, \qquad \mu^\star_{\text{p}} = \mu_{\text{p}} - g_\omega\langle\omega_0\rangle + \frac{1}{2}g_\rho\langle\rho_{30}\rangle, \tag{6.59}$$

$$C_\pm = \Big[1 + \mathrm{e}^{-\beta(E \pm \mu^\star_{\text{n}})}\Big], \qquad D_\pm = \Big[1 + \mathrm{e}^{-\beta(E \pm \mu^\star_{\text{p}})}\Big], \tag{6.60}$$

we find that

$$\ln Z_{\text{nucleon}} = \frac{2V}{T}\int \frac{\mathrm{d}^3k}{(2\pi)^3}\big(2E + T\ln C_+ + T\ln C_- + T\ln D_+ + T\ln D_-\big). \tag{6.61}$$

Assuming that μ_{n} and μ_{p} are positive, and ignoring the zero-point energy, the partition function becomes

$$\ln Z_{\text{nucleon}} = \frac{\beta V}{\pi^2}\int \mathrm{d}k \, \frac{k^4}{\sqrt{k^2 + (m^\star)^2}} \Big\{ \Theta\big[\mu^\star_{\text{n}} - \sqrt{k^2 + (m^\star)^2}\big] + \Theta\big[\mu^\star_{\text{p}} - \sqrt{k^2 + (m^\star)^2}\big] \Big\}, \tag{6.62}$$

in the zero-temperature limit. Performing the integral (6.62), we obtain

$$\ln Z_{\text{nucleon}} = \sum_{i=\text{n},\text{p}} \frac{\beta V}{24\pi^2} \left\{ \sqrt{k_i^2 + (m_i^\star)^2}\Big[2k_i^3 - 3(m_i^\star)^2 k_i\Big] + 3(m_i^\star)^4 \ln\left[\frac{k_i + \sqrt{k_i^2 + (m_i^\star)^2}}{m_i^\star}\right] \right\}, \tag{6.63}$$

where k_i represents the Fermi energy for a field of type i so that

$$k_i = \sqrt{(\mu_i^\star)^2 - (m_i^\star)^2}. \tag{6.64}$$

Lastly, we are left with the lepton part which is just the partition function for a free fermion field. The partition function will then have the same form as (6.63) if we substitute the effective nucleon mass with the lepton mass:

$$\ln Z_{\text{lepton}} = \sum_{i=\text{e},\mu} \frac{\beta V}{24\pi^2} \left\{ \sqrt{k_i^2 + (m_i)^2}\Big[2k_i^3 - 3(m_i)^2 k_i\Big] + 3(m_i)^4 \ln\left[\frac{k_i + \sqrt{k_i^2 + (m_i)^2}}{m_i}\right] \right\}, \tag{6.65}$$

where k_i satisfies

$$k_i = \sqrt{\mu_i^2 - m_i^2}. \tag{6.66}$$

It is somewhat intriguing that the partition function for the interacting nucleon field has the exact same form as the partition function for the free fermion field. This shows the simplicity of the mean-field approximation, as it removes the derivatives of the meson fields.

Note that the relations (6.64) and (6.66) can be rewritten as

$$\begin{aligned} \mu_{\mathrm{n}} &= g_\omega\langle\omega_0\rangle + \frac{1}{2}g_\rho\langle\rho_{30}\rangle + \sqrt{k_{\mathrm{n}}^2 + (m^\star)^2}, & \mu_{\mathrm{p}} &= g_\omega\langle\omega_0\rangle - \frac{1}{2}g_\rho\langle\rho_{30}\rangle + \sqrt{k_{\mathrm{p}}^2 + (m^\star)^2}, \\ \mu_{\mathrm{e}} &= \sqrt{k_{\mathrm{e}}^2 + m_{\mathrm{e}}^2}, & \mu_\mu &= \sqrt{k_\mu^2 + m_\mu^2}. \end{aligned} \tag{6.67}$$

Combined with (6.44), this gives

$$\mu_{\mathrm{n}} - \mu_{\mathrm{p}} = g_\rho\langle\rho_{30}\rangle + \sqrt{k_{\mathrm{n}}^2 + (m^\star)^2} - \sqrt{k_{\mathrm{p}}^2 + (m^\star)^2} = \mu_{\mathrm{e}} = \sqrt{k_{\mathrm{e}}^2 + m_{\mathrm{e}}^2} \tag{6.68}$$

Rearranging we can then express the Fermi momentum for the neutron as

$$k_{\mathrm{n}}^2 = \left[g_\rho\langle\rho_{30}\rangle - \sqrt{k_{\mathrm{p}}^2 + (m^\star)^2} - \sqrt{k_{\mathrm{e}}^2 + m_{\mathrm{e}}^2}\right]^2 - (m^\star)^2. \tag{6.69}$$

Combining (6.57),(6.63) and (6.65), we find the expressions for the pressure and the energy density

$$\begin{aligned} P = &-\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle^2 + \frac{1}{2}m_\rho^2\langle\rho_{30}\rangle^2 + \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 + \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4 \\ &+ \sum_{i=\mathrm{n,p,e},\mu} \frac{1}{24\pi^2}\left\{\sqrt{k_i^2 + (m_i^\star)^2}\Big[2k_i^3 - 3(m_i^\star)^2 k_i\Big] + 3(m_i^\star)^4 \ln\left[\frac{k_i + \sqrt{k_i^2 + (m_i^\star)^2}}{m_i^\star}\right]\right\}, \end{aligned} \tag{6.70}$$

$$\begin{aligned} \epsilon = &\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle^2 + \frac{1}{2}m_\rho^2\langle\rho_{30}\rangle^2 - \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 - \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4 \\ &+ \sum_{i=\mathrm{n,p,e},\mu} \frac{1}{24\pi^2}\left\{\sqrt{k_i^2 + (m_i^\star)^2}\Big[6k_i^3 + 3(m_i^\star)^2 k_i\Big] - 3(m_i^\star)^4 \ln\left[\frac{k_i + \sqrt{k_i^2 + (m_i^\star)^2}}{m_i^\star}\right]\right\}, \end{aligned} \tag{6.71}$$

where $m_i^\star$ denotes the effective mass for a field of type i.[7] Looking at the energy density, we hope that the sign of c is negative. Otherwise, the energy would not be bounded from below. Had our theory been fundamental, this would be catastrophic, but it is not. This model handles hadrons as point particles, and does thus not account for the effects that occur at densities so high that the quarks manifest themselves. However, should c be positive, we must still check that the pressure and energy density are well behaved in the sense that the energy density is strictly increasing as a function of the Fermi momenta and that they are both continuous, within the region of interest. In the case of a neutron star, this means densities up to 10 times the saturation density for nuclear matter [6, p. 182].

[7] The effective mass of a free field is of course the mass of the field it self. For example is $m_{\mathrm{e}}^\star = m_{\mathrm{e}}$.

6.5 Numerical solutions

We have so far written a large set of equations concerning the EoS. To make things clearer, we list the most important ones

$$m_\sigma^2\langle\sigma\rangle = g_\sigma\rho_{\rm s} + bmg_\sigma^3\langle\sigma\rangle^2 + cg_\sigma^4\langle\sigma\rangle^3, \tag{6.72}$$

$$\rho_{\rm s} = \sum_{i={\rm n,p}}\frac{m^\star}{2\pi^2}\left\{\sqrt{k_i^2+(m^\star)^2}k_i - (m^\star)^2\ln\left[\frac{\sqrt{k_i^2+(m^\star)^2}+k_i}{m^\star}\right]\right\}, \tag{6.73}$$

$$m^\star = m - g_\sigma\langle\sigma\rangle, \tag{6.74}$$

$$\langle\omega_0\rangle = \frac{g_\omega}{m_\omega^2}\rho, \tag{6.75}$$

$$\rho = \rho_{\rm n} + \rho_{\rm p}, \tag{6.76}$$

$$\langle\rho_{30}\rangle = \frac{g_\rho}{2m_\rho^2}\big(\rho_{\rm n} - \rho_{\rm p}\big), \tag{6.77}$$

$$\rho_i = \frac{k_i^3}{3\pi^2}, \tag{6.78}$$

$$\rho_{\rm p} = \rho_{\rm e} + \rho_\mu, \tag{6.79}$$

$$k_\mu^2 = \big(m_{\rm e}^2 + k_{\rm e}^2 - m_\mu^2\big)\Theta\big(m_{\rm e}^2 + k_{\rm e}^2 - m_\mu^2\big), \tag{6.80}$$

$$k_{\rm n}^2 = \left[g_\rho\langle\rho_{30}\rangle - \sqrt{k_{\rm p}^2 + (m^\star)^2} - \sqrt{k_{\rm e}^2 + m_{\rm e}^2}\right]^2 - (m^\star)^2. \tag{6.81}$$

We want to express the pressure and energy density in terms of one variable, which we will choose to be the Fermi momentum for the electron $k_{\rm e}$. In particular, we want to write the variables

$$\langle\sigma\rangle,\quad \langle\omega_0\rangle,\quad \langle\rho_{30}\rangle,\quad k_i,\quad m^\star, \tag{6.82}$$

as a function of $k_{\rm e}$. If we combine (6.72)-(6.74), we obtain

$$m - m^\star = \frac{g_\sigma^2}{m_\sigma^2}\sum_{i={\rm n,p}}\frac{m^\star}{2\pi^2}\left\{k_i\sqrt{k_i^2+(m^\star)^2} - (m^\star)^2\ln\left[\frac{\sqrt{k_i^2+(m^\star)^2}+k_i}{m^\star}\right] + bm\big(m-m^\star\big)^2 + c\big(m-m^\star\big)^3\right\}. \tag{6.83}$$

We can use this equation to determine the effective mass $m^\star$ for a set of Fermi-momenta $k_{\rm n}$ and $k_{\rm p}$. The result can then be inserted in (6.74) to determine $\langle\sigma\rangle$. Further, we can rewrite equations (6.78)–(6.81) in such a way that that the Fermi-momenta $k_{\rm n}$, $k_{\rm p}$ and k_μ only depend on the Fermi-momentum of the electron

$$k_\mu^2 = \big(k_{\rm e}^2 + m_{\rm e}^2 - m_\mu^2\big)\Theta\big(m_{\rm e}^2 + k_{\rm e}^2 - m_\mu^2\big), \tag{6.84}$$

$$k_{\rm p}^3 = k_{\rm e}^3 + \big(k_{\rm e}^2 + m_{\rm e}^2 - m_\mu^2\big)^{3/2}\Theta\big(m_{\rm e}^2 + k_{\rm e}^2 - m_\mu^2\big), \tag{6.85}$$

$$k_{\rm n}^2 = \left\{g_\rho\langle\rho_{30}\rangle - \sqrt{\left[k_{\rm e}^3 + \big(k_{\rm e}^2 + m_{\rm e}^2 - m_\mu^2\big)^{3/2}\Theta\big(m_{\rm e}^2 + k_{\rm e}^2 - m_\mu^2\big)\right]^{\frac{2}{3}} + (m^\star)^2} - \sqrt{k_{\rm e}^2 + m_{\rm e}^2}\right\}^2 - (m^\star)^2. \tag{6.86}$$

The equations for $k_{\rm n}$ and $m^\star$ can now be solved with a built in root finder in Python[8] for any electron Fermi-momentum. This closes our system, and the only thing that is left before we can study the mass-radius relation for this model, is to determine the coupling constants.

6.5.1 Determining the coupling constants

Surprisingly, it turns out that it is actually possible to determine the coupling constants analytically as shown in [6, p. 178-181]. However, this calculation is long and tedious, and even though analytic

[8] Or any other programming language of your desire.

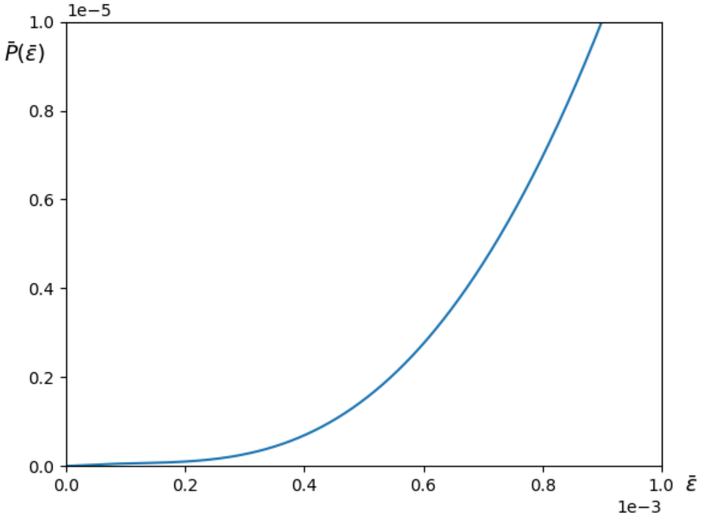


Figure 6.1: Dimensionless pressure as a function of dimensionless energy density for the improved σ–ω model.

expressions for the couplings can be useful due to their energy dependence, it will for our use suffice with a numerical treatment. The empirical values we wish to fit our couplings against will be the same as used in [25, p. 232], namely the saturation density

$$\rho_0 = 0.153\,\text{fm}^{-3}, \tag{6.87}$$

the binding energy

$$B_0 = \frac{\epsilon_0}{\rho_0} - m = -16.3\,\text{MeV}, \tag{6.88}$$

the Landau mass

$$m_{\text{L}} \equiv \sqrt{(m_0^\star)^2 + (k_{\text{F}})_0^2} = 0.83m, \tag{6.89}$$

and the compressibility

$$K_0 = (k_{\text{F}})_0^2 \frac{\mathrm{d}^2}{\mathrm{d}k_{\text{k}}^2} \frac{\epsilon}{\rho}\bigg|_{\rho=\rho_0} = 250\,\text{MeV}. \tag{6.90}$$

In addition, we use the symmetry-energy coefficient from [6, p. 187]

$$a_{\text{s}} = \left(\frac{g_\rho}{m_\rho}\right)^2 \frac{(k_{\text{F}})_0^3}{12\pi^2} + \frac{(k_{\text{F}})_0^2}{6\sqrt{(k_{\text{F}})_0^2 + (m^\star)^2}} = 32.5\,\text{MeV}. \tag{6.91}$$

In all expressions $(k_{\text{F}})_0$ denotes the Fermi momentum for nucleons in isospin symmetric matter at saturation density.

Firstly, g_ρ is given by rearranging (6.91). For the four other couplings, we use the function ”couplingConstants” in Appendix G.3. This function takes as input a range of possible guesses for the coupling constants, some parameter N, and the empirical values for the nuclear matter properties we wish to fit. The program runs four loops with N iterations that go through the range of possible guesses for each of the couplings g_σ, g_ω, b and c. For each iteration, the program uses a built in root-finding routine from scipy that solves the four equations (6.87)−(6.90), using the current coupling guesses. If the root-finding routine fails to converge, it tries a new set of couplings. When the routine converges, the function calls ”runAllChecks”. This function calculates the numbers (6.87)−(6.90), and compares them to the experimental ones. We do this because there were some stability issues with the root-finding

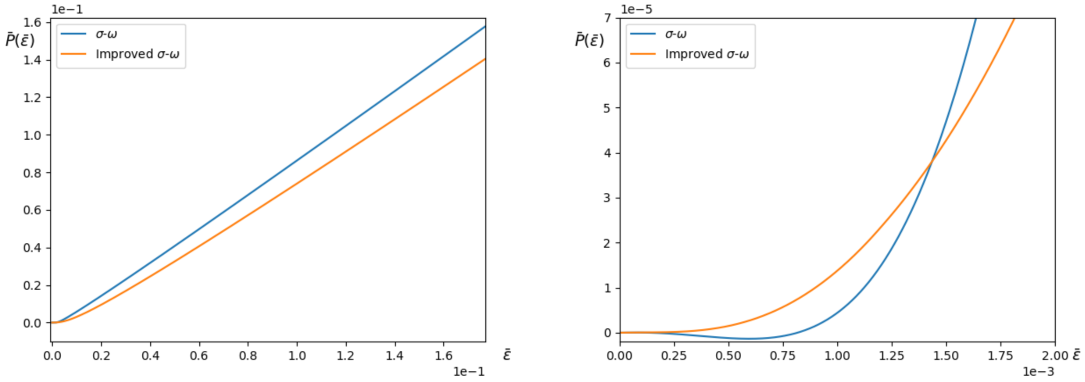


Figure 6.2: Dimensionless pressure $\bar{P}$ as a function of dimensionless energy density $\bar{\epsilon}$ for the σ-ω model in chapter 5.8 and the improved version. Figure on the right zooms in on the low density region of the figure on the left. The coupling constants are fitted to bulk properties of nuclear matter given in the text.

routine so that it sometimes converged towards wrong values. However, when the routine fails, it does so badly, so we just have to check that the values are not way off to be sure that we got the right numbers. The program used here checks if the values matches within about 5%, and if they do, the loop terminates and the coupling constants are returned. The built in root-finding routine improves the run time of the program compared to the method used to obtain the couplings in the σ–ω model, where a more brute force trial and error method was used. As a result, even though we have four loops instead of two, the program is much faster.

The experimental values used in this thesis are similar to the ones used in [44]. It is then reasonable that our couplings will not differ much. Thus we choose our guesses so that they are close to those used there. In these calculations we have used $N = 10$, and the guesses for the couplings are given by table 6.1. The coupling constants obtained from this are

	g_σ	g_ω	b	c
Guess	$[0, 20]$	$[0, 20]$	$[-0.01, 0.01]$	$[-0.01, 0.01]$

Table 6.1: The coupling constants are assumed to be within these intervals

$$g_\sigma = 8.711, \qquad g_\omega = 8.646, \qquad g_\rho = 8.610, \qquad b = -7.950 \cdot 10^{-3}, \qquad c = -6.947 \cdot 10^{-4}. \tag{6.92}$$

As we see, both b and c are negative, and so we do not need to worry about an unstable vacuum.

6.5.2 The equation of state and the mass-radius relation

Now that we have the coupling constants, we are able to plot the equation of state. In Figure 6.1 we see the dimensionless pressure as a function of dimensionless energy density. In this case, there are no unstable regions as we saw for the σ–ω model. We also observe that by comparing with the σ–ω model (Figure 6.2), the EoS is less "stiff" in this improved model. That is, the pressure increases slower as a function of the energy density. Then we should expect a smaller mass and radius. As shown by Figure 6.3 the maximum mass for this model is $2.03M_\odot$ with a radius $R = 10.8\,\text{km}$.

6.5.3 Population density

Figure 6.4 shows the relative population densities for the particles involved in this model. From the plot we see that there are clearly more neutrons than other particles. However, we also see that we are

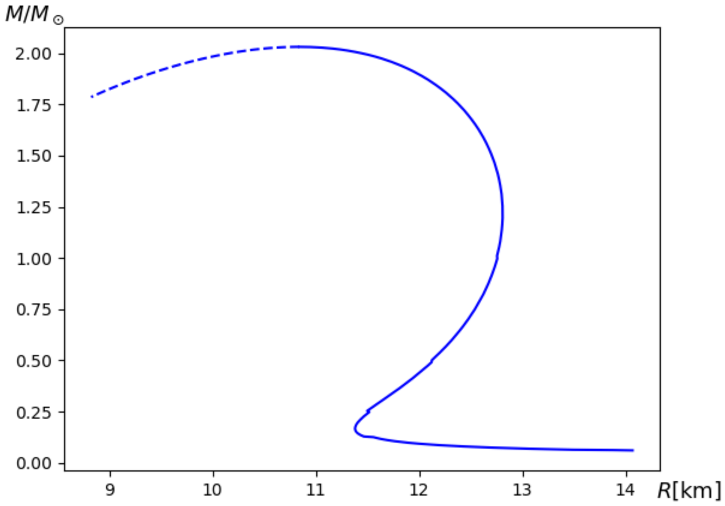


Figure 6.3: Mass-radius relation for the improved σ–ω model. Coupling constants are fitted to the bulk properties of nuclear matter given in the text. Dotted lines indicate unstable solutions.

far from having pure neutron matter, as we have assumed in the previous chapters. At the beginning there are equally many electrons and protons. When the density is high enough, muons appear and the symmetry in electron and proton densities breaks to satisfy the overall charge neutrality condition. This is the reason why we obtain the "kink" in the neutron density at around $\bar{\rho} \approx 0.1$.

6.6 Summary

We have used the σ–ω model as a foundation for a more complete model. Two self-interaction terms have been added in the same manner as Botuga and Bodmer did in [42] to fix the compressibility and the Landau mass. We have also introduced an isospin asymmetry, which allowed us to match the symmetry-energy coefficient. From this, we have found that the upper mass limit for neutron stars in this model is 2.03 solar masses, just above the lower limit set by the pulsar PSR J0348+0432.

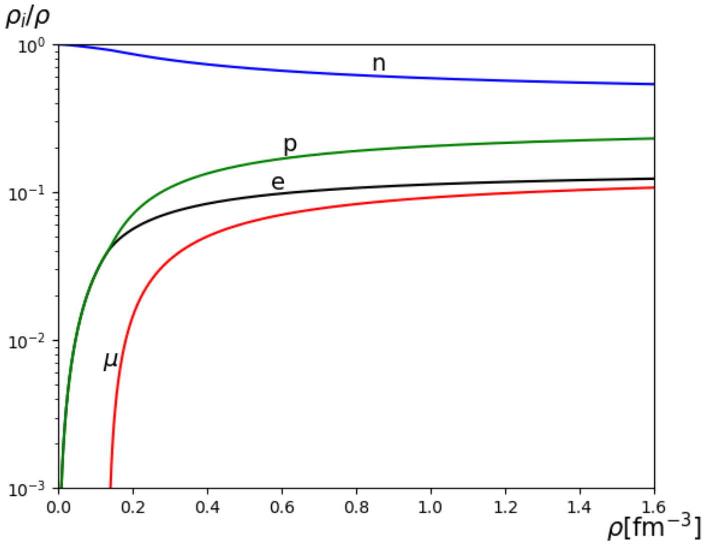


Figure 6.4: Relative population density ρ_i/ρ as a function of nucleon density ρ for the improved σ–ω model.

Chapter 7

Renormalization

It is time to settle the issue with the infinite zero-point energy term in the partition function. Infinite contributions to the pressure and energy density are not as unsettling as they first may seem. Whenever we measure a quantity in nature, we always have to compare it to something else. To say that an object is big is ambiguous, but to say that it is has five times the diameter of a proton is useful information. In the same manner, even though the zero-point energy is infinite, so is also the vacuum energy. Thus, the effect that we actually observe is the difference between these energies, which we will call the vacuum-energy shift.[1]

7.1 Vacuum-energy shift from fermions

The vacuum energy is given by the zero-point energy for all particles present with their original mass. This means that for the improved σ–ω model, the vacuum-energy shift is given by

$$\begin{aligned} V &= \left[-\sum_{i=\mathrm{e},\mu,\mathrm{n},\mathrm{p}} 2\int \frac{\mathrm{d}^3 k_i}{(2\pi)^3}\sqrt{k_i^2+(m_i^\star)^2}\right] - \left[-\sum_{i=\mathrm{e},\mu,\mathrm{n},\mathrm{p}} 2\int \frac{\mathrm{d}^3 k_i}{(2\pi)^3}\sqrt{k_i^2+m_i^2}\right] \\ &= -2\sum_{i=\mathrm{n},\mathrm{p}}\int \frac{\mathrm{d}^3 k_i}{(2\pi)^3}\left[\sqrt{k_i^2+(m_i^\star)^2}-\sqrt{k_i^2+m_i^2}\right] \\ &= -4\int \frac{\mathrm{d}^3 k}{(2\pi)^3}\left[\sqrt{k^2+(m^\star)^2}-\sqrt{k^2+m^2}\right], \end{aligned} \tag{7.1}$$

where we have used that $m_i^\star = m_i$ for $i=$ e, μ, and the fact that neutrons and protons in this thesis are regarded to have the same mass. We now see that our calculation boils down to evaluating the integral

$$I(m) = 4\int \frac{\mathrm{d}^3 k}{(2\pi)^3}\sqrt{k^2+m^2}. \tag{7.2}$$

This integral is divergent, but we may temporarily move over to $d = 3-2\epsilon$ dimensions where it is finite, and then expand it continuously to $d=3$. This will allow us to isolate the divergences so that we can add counterterms to the Lagrangian that cancel them.[2] In $d = 3-2\epsilon$ dimensions, the integral becomes

$$I(m) = 4\Lambda^{3-d}\int \frac{\mathrm{d}^d k}{(2\pi)^d}\sqrt{k^2+m^2} = 4\Lambda^{3-d}\int \mathrm{d}\Omega_d \int \frac{\mathrm{d}k}{(2\pi)^d}k^{d-1}\sqrt{k^2+m^2}. \tag{7.3}$$

[1] This is not entirely true in the case of gravity as the EH-Lagrangian is not invariant under the addition of a constant to the stress-energy tensor [45].

[2] It might seem ad hoc to include counterterms so that we can get rid of infinities, but in essence what we do is just redefining the parameters of our model. If the energy is some huge number plus one, we might as well call that energy one instead of using the large number. The physics is the same.

Here, Λ is some constant with dimension mass that makes sure that I has the right dimensions. The surface of a sphere in d dimensions is given by [46]

$$\int \mathrm{d}\Omega_d = \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}, \tag{7.4}$$

where $\Gamma(x)$ denotes the gamma function defined by (D.3). Then we find that

$$I(m) = 4\frac{2\pi^{\frac{d}{2}}\Lambda^{3-d}}{(2\pi)^d\Gamma(\frac{d}{2})}\int \mathrm{d}k\, k^{d-1}\sqrt{k^2+m^2} = \frac{8\Lambda^{3-d}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\int \mathrm{d}k\, k^{d-1}\sqrt{k^2+m^2}. \tag{7.5}$$

Substituting $k^2 = um^2$, which gives $\mathrm{d}k = \frac{m^2}{2k}\mathrm{d}u$, we obtain

$$I(m) = \frac{8\Lambda^{3-d}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\int \mathrm{d}u\frac{(um^2)^{\frac{d-2}{2}}}{2}m^2\sqrt{um^2+m^2} = \frac{4\Lambda^{3-d}m^{d+1}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\int \mathrm{d}u\, u^{\frac{d-2}{2}}\sqrt{u+1}. \tag{7.6}$$

Introducing the constants $x = \frac{d}{2}$ and $y = -\frac{1+d}{2}$, the expression can be rewritten as

$$I(m) = \frac{4\Lambda^{3-d}m^{d+1}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\int \mathrm{d}u\frac{u^{x-1}}{(u+1)^{x+y}}. \tag{7.7}$$

This we recognize as the beta function (D.5) times some constant. The integral (7.7) is then equivalent to

$$I(m) = \frac{4\Lambda^{3-d}m^{d+1}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\beta(x,y) = \frac{4\Lambda^{3-d}m^{d+1}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}. \tag{7.8}$$

Inserting values for x and y gives

$$I(m) = \frac{4\Lambda^{3-d}m^{d+1}}{(4\pi)^{\frac{d}{2}}\Gamma(\frac{d}{2})}\frac{\Gamma(\frac{d}{2})\Gamma(-\frac{d+1}{2})}{\Gamma(-\frac{1}{2})} = -\frac{4\Lambda^{3-d}m^{d+1}\Gamma(-\frac{d+1}{2})}{(4\pi)^{\frac{d}{2}}2\pi^{\frac{1}{2}}} = -\frac{4\Lambda^{3-d}m^{d+1}\Gamma(-\frac{d+1}{2})}{(4\pi)^{\frac{d+1}{2}}}, \tag{7.9}$$

where we have used that $\Gamma(-\frac{1}{2}) = -2\sqrt{\pi}$. Inserting $d = 3-2\epsilon$ yields

$$I(m) = -\frac{4m^4}{(2\pi)^{2-\epsilon}}\Big(\frac{m}{\Lambda}\Big)^{-2\epsilon}\Gamma(-2+\epsilon). \tag{7.10}$$

Expanding the gamma function around $\epsilon = 0$ we have [25, p. 71]

$$\Gamma(-2+\epsilon) = \frac{1}{2\epsilon} + \frac{3}{4} - \gamma + \mathcal{O}(\epsilon), \tag{7.11}$$

where we have introduced the Euler-Mascheroni constant defined by

$$\gamma \equiv -\frac{\mathrm{d}}{\mathrm{d}x}\Gamma(x)\Big|_{x=1} \approx 0.5772. \tag{7.12}$$

Using the power series

$$a^{x+b} = a^b + xa^b\ln a + \mathcal{O}(x^2), \tag{7.13}$$

we find that to first order in ϵ

$$I(m) = -\frac{m^4}{8\pi^2}\Big[-\gamma + \frac{1}{\epsilon} - \frac{3}{2} + \ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\Big] + \mathcal{O}(\epsilon). \tag{7.14}$$

Going back to (7.1), we obtain the vacuum-energy shift

$$\begin{aligned}
V =& \frac{(m^\star)^4}{8\pi^2}\bigg\{-\gamma+\frac{1}{\epsilon}-\frac{3}{2}+\ln\bigg[\frac{4\pi\Lambda^2}{(m^\star)^2}\bigg]\bigg\}-\frac{m^4}{8\pi^2}\bigg\{-\gamma+\frac{1}{\epsilon}-\frac{3}{2}+\ln\bigg[\frac{4\pi\Lambda^2}{m^2}\bigg]\bigg\}+\mathcal{O}(\epsilon)\\
=& \frac{1}{8\pi^2}\bigg\{\bigg[(m^\star)^4-m^4\bigg]\bigg[-\gamma+\frac{1}{\epsilon}-\frac{3}{2}\bigg]\\
&+(m^\star)^4\ln\bigg[\frac{4\pi\Lambda^2}{(m^\star)^2}\bigg]-(m^\star)^4\ln\bigg[\frac{4\pi\Lambda^2}{m^2}\bigg]+(m^\star)^4\ln\bigg[\frac{4\pi\Lambda^2}{m^2}\bigg]-m^4\ln\bigg[\frac{4\pi\Lambda^2}{m^2}\bigg]\bigg\}+\mathcal{O}(\epsilon)\\
=& \frac{1}{8\pi^2}\bigg\{\bigg[(m^\star)^4-m^4\bigg]\bigg[-\gamma+\frac{1}{\epsilon}-\frac{3}{2}+\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg]+2(m^\star)^4\ln\Big(\frac{m}{m^\star}\Big)\bigg\}+\mathcal{O}(\epsilon).
\end{aligned} \tag{7.15}$$

Note that we now have isolated the divergence to the terms containing $\frac{1}{\epsilon}$. Remembering that $m^\star = m - g_\sigma\langle\sigma\rangle$, we find

$$\begin{aligned}
V =& \frac{1}{8\pi^2}\bigg\{\bigg[-4m^3(g_\sigma\langle\sigma\rangle)+6m^2(g_\sigma\langle\sigma\rangle)^2-4m(g_\sigma\langle\sigma\rangle)^3+(g_\sigma\langle\sigma\rangle)^4\bigg]\bigg[-\gamma+\frac{1}{\epsilon}-\frac{3}{2}+\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg]\\
&+2\bigg[m^4-4m^3(g_\sigma\langle\sigma\rangle)+6m^2(g_\sigma\langle\sigma\rangle)^2-4m(g_\sigma\langle\sigma\rangle)^3+(g_\sigma\langle\sigma\rangle)^4\bigg]\ln\bigg[1-\frac{(g_\sigma\langle\sigma\rangle)}{m}\bigg]\bigg\}+\mathcal{O}(\epsilon).
\end{aligned} \tag{7.16}$$

Since this expression and the Lagrangian is of order $\mathcal{O}(\langle\sigma\rangle^4)$, it will be convenient to expand the last term using

$$\ln(1-x) = -x-\frac{x^2}{2}-\frac{x^3}{3}-\frac{x^4}{4}-\mathcal{O}(x^5), \tag{7.17}$$

to obtain

$$\begin{aligned}
(m^\star)^4\ln\bigg[1-\frac{(g_\sigma\langle\sigma\rangle)}{m}\bigg] =& \bigg[m^4-4m^3(g_\sigma\langle\sigma\rangle)+6m^2(g_\sigma\langle\sigma\rangle)^2-4m(g_\sigma\langle\sigma\rangle)^3+(g_\sigma\langle\sigma\rangle)^4\bigg]\ln\bigg[1-\frac{(g_\sigma\langle\sigma\rangle)}{m}\bigg]\\
=&-\bigg[m^4-4m^3(g_\sigma\langle\sigma\rangle)+6m^2(g_\sigma\langle\sigma\rangle)^2-4m(g_\sigma\langle\sigma\rangle)^3+(g_\sigma\langle\sigma\rangle)^4\bigg]\\
&\times\bigg[\frac{(g_\sigma\langle\sigma\rangle)}{m}+\frac{(g_\sigma\langle\sigma\rangle)^2}{2m^2}+\frac{(g_\sigma\langle\sigma\rangle)^3}{3m^3}+\frac{(g_\sigma\langle\sigma\rangle)^4}{4m^4}+\mathcal{O}\big(\langle\sigma\rangle^5\big)\bigg]\\
=&-\bigg[m^3(g_\sigma\langle\sigma\rangle)+\frac{1}{2}m^2(g_\sigma\langle\sigma\rangle)^2-4m^2(g_\sigma\langle\sigma\rangle)^2+\frac{1}{3}m(g_\sigma\langle\sigma\rangle)^3-2m(g_\sigma\langle\sigma\rangle)^3\\
&+6m(g_\sigma\langle\sigma\rangle)^3+\frac{1}{4}(g_\sigma\langle\sigma\rangle)^4-\frac{4}{3}(g_\sigma\langle\sigma\rangle)^4+3(g_\sigma\langle\sigma\rangle)^4-4(g_\sigma\langle\sigma\rangle)^4\\
&+(g_\sigma\langle\sigma\rangle)^4+\mathcal{O}\big(\langle\sigma\rangle^5\big)\bigg]\\
=&-m^3(g_\sigma\langle\sigma\rangle)+\frac{7}{2}m^2(g_\sigma\langle\sigma\rangle)^2-\frac{13}{3}m(g_\sigma\langle\sigma\rangle)^3+\frac{25}{12}(g_\sigma\langle\sigma\rangle)^4+\mathcal{O}\big(\langle\sigma\rangle^5\big).
\end{aligned} \tag{7.18}$$

This means that we can express the vacuum-energy shift as

$$
\begin{aligned}
V =& \frac{1}{8\pi^2}\Bigg\{\bigg[-4m^3(g_\sigma\langle\sigma\rangle)+6m^2(g_\sigma\langle\sigma\rangle)^2-4m(g_\sigma\langle\sigma\rangle)^3+(g_\sigma\langle\sigma\rangle)^4\bigg]\bigg[-\gamma+\frac{1}{\epsilon}-\frac{3}{2}+\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg] \\
&-2m^3(g_\sigma\langle\sigma\rangle)+7m^2(g_\sigma\langle\sigma\rangle)^2-\frac{26}{3}m(g_\sigma\langle\sigma\rangle)^3+\frac{25}{6}(g_\sigma\langle\sigma\rangle)^4\Bigg\}+\mathcal{O}(\langle\sigma\rangle^5) \\
=& \frac{1}{8\pi^2}\Bigg\{m^3(g_\sigma\langle\sigma\rangle)\bigg[-4\gamma+\frac{4}{\epsilon}-4+4\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg]+m^2(g_\sigma\langle\sigma\rangle)^2\bigg[6\gamma-\frac{6}{\epsilon}+2-6\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg] \\
&+m(g_\sigma\langle\sigma\rangle)^3\bigg[-4\gamma+\frac{4}{\epsilon}+\frac{8}{3}+4\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg]+(g_\sigma\langle\sigma\rangle)^4\bigg[\gamma-\frac{1}{\epsilon}-\frac{8}{3}-\ln\Big(\frac{4\pi\Lambda^2}{m^2}\Big)\bigg]\Bigg\}+\mathcal{O}(\langle\sigma\rangle^5).
\end{aligned}
\tag{7.19}
$$

If we now consider the part of the Lagrangian containing the σ-field at tree level

$$
\mathscr{L}_{\sigma,\mathrm{tree}} = m\bar\psi\psi - g_\sigma\langle\sigma\rangle\bar\psi\psi - \frac{1}{2}m_\sigma^2\langle\sigma\rangle^2+\frac{1}{3}bm(g_\sigma\langle\sigma\rangle)^3+\frac{1}{4}c(g_\sigma\langle\sigma\rangle)^4, \tag{7.20}
$$

and inserting the equation of motion for $\bar\psi\psi$ given by (6.5), we see that we can write

$$
\begin{aligned}
\mathscr{L}_{\sigma,\mathrm{tree}} =& -\frac{m}{g_\sigma}(m_\sigma^2\langle\sigma\rangle-bmg_\sigma^3\langle\sigma\rangle^2-cg_\sigma^4\langle\sigma\rangle^3)+m_\sigma^2\langle\sigma\rangle^2-bm(g_\sigma\langle\sigma\rangle)^3-c(g_\sigma\langle\sigma\rangle)^4 \\
&-\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2+\frac{1}{3}mb(g_\sigma\langle\sigma\rangle)^3+\frac{1}{4}c(g_\sigma\langle\sigma\rangle)^4 \\
=& -\frac{mm_\sigma^2}{g_\sigma}\langle\sigma\rangle+\Big(bm^2g_\sigma^2+\frac{1}{2}m_\sigma^2\Big)\langle\sigma\rangle^2+m\Big(c-\frac{2b}{3}\Big)(g_\sigma\langle\sigma\rangle)^3-\frac{3}{4}c\,(g\sigma\langle\sigma\rangle)^4.
\end{aligned}
\tag{7.21}
$$

This is a polynomial in $\langle\sigma\rangle$ containing all orders up to $\mathcal{O}(\langle\sigma\rangle^5)$. Then we are able to dispose of all the infinities by adding counterterms to the coupling constants. In other words, we write

$$
g_\sigma \to g_\sigma+\delta g_\sigma, \qquad m_\sigma^2\to m_\sigma^2+\delta m_\sigma^2, \qquad b\to b+\delta b, \qquad c\to c+\delta c, \tag{7.22}
$$

where δa_i are counterterms chosen to cancel the divergences. However, having established that we can render the energy density finite, we do not need to find the counterterms for the coupling constants. Instead, we can directly cancel the divergent parts in the energy density by adding the sum $\sum_{i=1}^4 c_i\langle\sigma\rangle^i$ where the constants c_i are such that all divergent i-th order terms in $\langle\sigma\rangle$ vanish. One may argue that by doing so we do not know which values to assign the coupling constants. However, we can just fit them according to the properties of nuclear matter after we have added the vacuum shift anyway, and this is easier. Choosing the on-shell renormalization scheme, that is, we renormalize in such a way that all particles have the mass we measure in the laboratory, we can choose the counterterms so that they cancel all divergent and finite remainders of V up to $\mathcal{O}(\langle\sigma\rangle^5)$. Thus, the renormalized vacuum shift is given by the fifth-order contribution from the logarithm (7.18):

$$
V_{\mathrm{r}} = -\frac{1}{4\pi^2}\Bigg\{(m^\star)^4\ln\Big(\frac{m^\star}{m}\Big)-\bigg[-m^3(g_\sigma\langle\sigma\rangle)+\frac{7}{2}m^2(g_\sigma\langle\sigma\rangle)^2-\frac{13}{3}m(g_\sigma\langle\sigma\rangle)^3+\frac{25}{12}(g_\sigma\langle\sigma\rangle)^4\bigg]\Bigg\}. \tag{7.23}
$$

Then we find that the renormalized pressure and energy density is

$$
P_{\mathrm{r}} = P - V_{\mathrm{r}}, \qquad \epsilon_{\mathrm{r}} = \epsilon + V_{\mathrm{r}}, \tag{7.24}
$$

where P and ϵ are the pressure and energy density obtained in the improved σ–ω model.

Using the equilibrium condition, we note that the vacuum-energy shift also alters the self-consistency equation for the σ-meson:

$$
0 = \frac{\partial P_{\mathrm{r}}}{\partial m^\star} = \frac{\partial P}{\partial m^\star}-\frac{\partial V_{\mathrm{r}}}{\partial m^\star} = -\rho_{\mathrm{s}}+\frac{m_\sigma^2}{g_\sigma}(m-m^\star)-bm(m-m^\star)^2-c(m-m^\star)^3-\frac{\partial V_{\mathrm{r}}}{\partial m^\star}, \tag{7.25}
$$

where we in the last step inserted (6.12). The derivative of the vacuum energy shift is given by

$$\begin{aligned}
\frac{\partial V_{\mathrm{r}}}{\partial m^\star} =& -\frac{1}{4\pi^2}\frac{\partial}{\partial m^\star}\bigg\{(m^\star)^4\ln\Big(\frac{m^\star}{m}\Big) - \bigg[-m^3(m-m^\star) + \frac{7}{2}(m-m^\star)^2 - \frac{13}{3}m(m-m^\star)^3 + \frac{25}{12}(m-m^\star)^4\bigg]\bigg\}\\
=& -\frac{1}{4\pi^2}\bigg\{4(m^\star)^3\ln\Big(\frac{m^\star}{m}\Big) + (m^\star)^3 - \bigg[m^3 - 7m^2(m-m^\star) + 13(m-m^\star)^2 - \frac{25}{3}(m-m^\star)^3\bigg]\bigg\}\\
=& -\frac{1}{4\pi^2}\bigg\{4(m^\star)^3\ln\Big(\frac{m^\star}{m}\Big) - \bigg[m^3 - 3m^2m^\star + 3m(m^\star)^2 - (m^\star)^3\bigg] - 3m^2m^\star + 3m(m^\star)^2\\
&+ 7m^2(m-m^\star) - 13m(m-m^\star)^2 + \frac{25}{3}(m-m^\star)^3\bigg\}\\
=& -\frac{1}{4\pi^2}\bigg\{4(m^\star)^3\ln\Big(\frac{m^\star}{m}\Big) + 10m^2(m-m^\star) + \bigg[3m^3 - 6m^2m^\star + 3m(m^\star)^2\bigg]\\
&+ 6m^2m^\star - 6m^3 - 13m(m-m^\star)^2 + \frac{22}{3}(m-m^\star)^3\bigg\}\\
=& -\frac{1}{4\pi^2}\bigg[4(m^\star)^3\ln\Big(\frac{m^\star}{m}\Big) + 10m^2(m-m^\star) - 6m^3 - 10m(m-m^\star)^2 + 6m(m^\star)^2 + \frac{22}{3}(m-m^\star)^3\bigg]\\
=& -\frac{1}{\pi^2}\bigg[(m^\star)^3\ln\Big(\frac{m^\star}{m}\Big) + m^2(m-m^\star) - \frac{5}{2}m(m-m^\star)^2 + \frac{11}{6}(m-m^\star)^3\bigg]. \qquad (7.26)
\end{aligned}$$

We have so far found the change in the vacuum energy due to the σ-meson's interaction with the baryon current. But having introduced the renormalization procedure, we may just as well account for more effects that alters the vacuum energy. Next we will look at the effect self-interactions have on the vacuum energy.

7.2 Vacuum-energy shift from self-interactions

Self-interactions induce a shift in the vacuum energy. We will in this section show this. We begin by dividing the σ-field in a classical part σ_0 and a quantum part $\tilde{\sigma}$ by writing

$$\sigma = \sigma_0 + \tilde{\sigma}, \qquad (7.27)$$

since it is the only field that we have included self-interactions for. The Lagrangian containing the σ-field in Euclidean space then changes to

$$\mathscr{L}_\sigma = \mathscr{L}_{\sigma_0} + \mathscr{L}_1 + \mathscr{L}_2, \qquad (7.28)$$

where we have defined

$$\begin{aligned}
\mathscr{L}_{\sigma_0} &= \frac{1}{2}(\partial_\mu\sigma_0)^2 + \frac{1}{2}m_\sigma^2\sigma_0^2 - g_\sigma\sigma_0\bar{\psi}\psi - \frac{1}{3}mb(g_\sigma\sigma_0)^3 - \frac{1}{4}c(g_\sigma\sigma_0)^4,\\
\mathscr{L}_1 &= (\partial\sigma_0)(\partial\tilde{\sigma}) + m_\sigma^2\sigma_0\tilde{\sigma} - g_\sigma\tilde{\sigma}\bar{\psi}\psi - bm\sigma_0^2\tilde{\sigma} - cg_\sigma^4\big(\sigma_0^3\tilde{\sigma} + \sigma_0\tilde{\sigma}^3\big),\\
\mathscr{L}_2 &= \frac{1}{2}(\partial_\mu\tilde{\sigma})^2 + \frac{1}{2}m_\sigma^2\tilde{\sigma}^2 - bmg_\sigma^3\sigma_0\tilde{\sigma}^2 - \frac{3c}{2}g_\sigma^4\sigma_0^2\tilde{\sigma}^2 - \frac{1}{3}mb(g_\sigma\tilde{\sigma})^3 - \frac{1}{4}c(g_\sigma\tilde{\sigma})^4.
\end{aligned}$$

For simplicity, we will define the function

$$f(x) = \frac{1}{3}mb(g_\sigma x)^3 + \frac{1}{4}c(g_\sigma x)^4, \qquad (7.29)$$

which will allow us to simplify these expressions

$$\mathscr{L}_{\sigma_0} = \frac{1}{2}(\partial_\mu\sigma_0)^2 + \frac{1}{2}m_\sigma^2\sigma_0 - g_\sigma\sigma_0\bar{\psi}\psi - f(\sigma_0), \qquad (7.30)$$

$$\mathscr{L}_1 = (\partial\sigma_0)(\partial\tilde{\sigma}) + m_\sigma^2\sigma_0\tilde{\sigma} - g_\sigma\tilde{\sigma}\bar{\psi}\psi - \tilde{\sigma}f'(\sigma_0), \qquad (7.31)$$

$$\mathscr{L}_2 = \frac{1}{2}(\partial_\mu\tilde{\sigma})^2 + \frac{1}{2}m_\sigma^2\tilde{\sigma}^2 - \frac{\tilde{\sigma}^2}{2}f''(\sigma_0) - cg_\sigma^4\sigma_0\tilde{\sigma}^3 - \frac{1}{3}mb(g_\sigma\tilde{\sigma})^3 - \frac{1}{4}c(g_\sigma\tilde{\sigma})^4. \qquad (7.32)$$

We note that $\mathscr{L}_{\sigma_0}$ is just the classical Lagrangian we used to calculate the original equation of state, and does thus not contribute to the vacuum-energy shift. We proceed by calculating the action integrals for $\mathscr{L}_1$ and $\mathscr{L}_2$. Firstly we have

$$\begin{aligned} S_1 &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \, \mathscr{L}_1 \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \Big[(\partial\sigma_0)(\partial\tilde{\sigma}) + m_\sigma^2 \sigma_0 \tilde{\sigma} - g_\sigma \tilde{\sigma} \bar{\psi}\psi - \tilde{\sigma} f'(\sigma_0) \Big] \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \, \tilde{\sigma} \Big[\Big(-\Box + m_\sigma^2 \Big) \sigma_0 - g_\sigma \bar{\psi}\psi - f'(\sigma_0) \Big], \end{aligned} \tag{7.33}$$

where we in the last step have performed one partial integration and ignored the surface term as it by definition vanishes at the boundary. If we transform equation (6.4) to Euclidean space, we see that this integral is zero. The only part that contributes to the vacuum-energy shift is then the action

$$\begin{aligned} S_2 &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \Big[\frac{1}{2}(\partial_\mu \tilde{\sigma})^2 + \frac{1}{2} m_\sigma^2 \tilde{\sigma}^2 - \frac{\tilde{\sigma}^2}{2} f''(\sigma_0) - c g_\sigma^4 \sigma_0 \tilde{\sigma}^3 - \frac{1}{3} m b (g_\sigma \tilde{\sigma})^3 - \frac{1}{4} c (g_\sigma \tilde{\sigma})^4 \Big] \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \Big[-\frac{1}{2} \tilde{\sigma} \Box \tilde{\sigma} + \frac{1}{2} m_\sigma^2 \tilde{\sigma}^2 - \frac{\tilde{\sigma}^2}{2} f''(\sigma_0) - c g_\sigma^4 \sigma_0 \tilde{\sigma}^3 - \frac{1}{3} m b (g_\sigma \tilde{\sigma})^3 - \frac{1}{4} c (g_\sigma \tilde{\sigma})^4 \Big] \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \, \tilde{\sigma} \Big[-\frac{1}{2} \Box + \frac{1}{2} m_\sigma^2 - \frac{1}{2} f''(\sigma_0) - c g_\sigma^4 \sigma_0 \tilde{\sigma} - \frac{1}{3} m b g_\sigma \tilde{\sigma} - \frac{1}{4} c g_\sigma^4 \tilde{\sigma}^2 \Big] \tilde{\sigma}. \end{aligned} \tag{7.34}$$

Assuming that the vacuum fluctuations are small, we neglect all $\mathcal{O}(\tilde{\sigma}^3)$ contributions, and expand the field in its frequency-momentum eigenstates

$$\tilde{\sigma} = \sum_n \sum_{\boldsymbol{k}} \tilde{\sigma}_{n,\boldsymbol{k}} \mathrm{e}^{\mathrm{i}(\omega_n \tau + \boldsymbol{k}\cdot\boldsymbol{x})}, \tag{7.35}$$

so that

$$\begin{aligned} S_2 &= \frac{1}{\beta V} \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3x \sum_n \sum_m \sum_{\boldsymbol{k}} \sum_{\boldsymbol{k}'} \tilde{\sigma}_{n,\boldsymbol{k}} \mathrm{e}^{-\mathrm{i}(\omega_n \tau + \boldsymbol{k}\cdot\boldsymbol{x})} \Big[-\frac{1}{2}\Box + \frac{1}{2} m_\sigma^2 - \frac{1}{2} f''(\sigma_0) \Big] \tilde{\sigma}_{m,\boldsymbol{k}'} \mathrm{e}^{\mathrm{i}(\omega_n \tau + \boldsymbol{k}\cdot\boldsymbol{x})} \\ &= \frac{1}{2} \sum_n \sum_{\boldsymbol{k}} \tilde{\sigma}_{n,\boldsymbol{k}} \Big[\omega_n^2 + \boldsymbol{k}^2 + m_\sigma^2 - f''(\sigma_0) \Big] \tilde{\sigma}_{n,\boldsymbol{k}}. \end{aligned} \tag{7.36}$$

The partition function then becomes[3]

$$Z_2 = \int \mathcal{D}\tilde{\sigma} \, \mathrm{e}^{-\frac{1}{2} \sum_n \sum_{\boldsymbol{k}} \tilde{\sigma}_{n,\boldsymbol{k}} \left[\omega_n^2 + \boldsymbol{k}^2 + m_\sigma^2 - f''(\sigma_0) \right] \tilde{\sigma}_{n,\boldsymbol{k}}}. \tag{7.37}$$

This is a standard Gaussian integral with the solution

$$\ln Z_2 = -\frac{1}{2} \mathrm{Tr} \ln \Big[\omega_n^2 + \boldsymbol{k}^2 + m_\sigma^2 - f''(\sigma_0) \Big] = -\frac{1}{2} \sum_n \sum_{\boldsymbol{k}} \ln \Big[\omega_n^2 + \boldsymbol{k}^2 + (m_\sigma^\star)^2 \Big], \tag{7.38}$$

where we have defined

$$(m_\sigma^\star)^2 = m_\sigma^2 - f''(\sigma_0). \tag{7.39}$$

Using (4.78) and taking the continuum limit, we obtain

$$\ln Z_2 = -\beta V \int \frac{\mathrm{d}^3 k}{(2\pi)^3} \Big[\frac{1}{2} \sqrt{k^2 + (m_\sigma^\star)^2} + T \ln \Big(1 - \mathrm{e}^{-\beta \sqrt{k^2 + (m_\sigma^\star)^2}} \Big) \Big]. \tag{7.40}$$

[3] Actually, we should also integrate over the conjugate momenta $\pi = \frac{\partial \mathscr{L}_2}{\partial(\frac{\partial \tilde{\sigma}}{\partial \tau})} = \frac{\partial \tilde{\sigma}}{\partial \tau}$, but there are no such factors in the action.

If we now only consider the contribution to the zero-point energy, we see that the vacuum-energy shift is given by

$$U = -\frac{1}{2}\int \frac{\mathrm{d}^3k}{(2\pi)^3}\Big[\sqrt{k^2+(m_\sigma^\star)^2} - \sqrt{k^2+m_\sigma^2}\Big]. \tag{7.41}$$

From this it is clear why the self-interacting σ-particles contribute to the vacuum-energy shift. If we remove the self-interaction terms in the Lagrangian, then we would have $f(\sigma_0) = 0$, and $m_\sigma = m_\sigma^\star$, so that $U = 0$. Thus, we do not need to do the same procedure with the ω-field. From here on we will now invoke the mean-field approximation, setting $\sigma_0 \to \langle\sigma\rangle$.

The integral (7.41) was solved in the previous section. We thus know that the renormalized contribution to the vacuum energy shift from U is given by the fifth order contribution from the logarithm

$$\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big). \tag{7.42}$$

Expanding (7.42) to fourth order in $\langle\sigma\rangle$ we obtain

$$\begin{aligned}
(m_\sigma^\star)^4\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big) &= \frac{1}{2}\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big)^2 \\
&=\frac{1}{2}\Big[m_\sigma^2 - f''(\langle\sigma\rangle)\Big]\ln\Big[1 - \frac{f''(\langle\sigma\rangle)}{m_\sigma^2}\Big] \\
&=\frac{1}{2}\Big\{m_\sigma^4 - 2m_\sigma^2 f''(\langle\sigma\rangle) + \big[f''(\langle\sigma\rangle)\big]^2\Big\} \\
&\quad\times\Big\{-\frac{f''(\langle\sigma\rangle)}{m_\sigma^2} - \frac{\big[f''(\langle\sigma\rangle)\big]^2}{2m_\sigma^4} - \frac{\big[f''(\langle\sigma\rangle)\big]^3}{3m_\sigma^6} - \frac{\big[f''(\langle\sigma\rangle)\big]^4}{4m_\sigma^8} + \mathcal{O}(\langle\sigma\rangle^5)\Big\} \\
&=\frac{1}{2}\Big\{-m_\sigma^2 f''(\langle\sigma\rangle) - \frac{\big[f''(\langle\sigma\rangle)\big]^2}{2} + 2\big[f''(\langle\sigma\rangle)\big]^2 - \frac{\big[f''(\langle\sigma\rangle)\big]^3}{3m_\sigma^2} + \frac{\big[f''(\langle\sigma\rangle)\big]^3}{m_\sigma^2} \\
&\quad - \frac{\big[f''(\langle\sigma\rangle)\big]^3}{m_\sigma^2} - \frac{\big[f''(\langle\sigma\rangle)\big]^4}{4m_\sigma^4} + \frac{2\big[f''(\langle\sigma\rangle)\big]^4}{3m_\sigma^4} - \frac{\big[f''(\langle\sigma\rangle)\big]^4}{2m_\sigma^4}\Big\} \\
&=\frac{1}{2}\Big\{-m_\sigma^2 f''(\langle\sigma\rangle) + \frac{3}{2}\big[f''(\langle\sigma\rangle)\big]^2 - \frac{\big[f''(\langle\sigma\rangle)\big]^3}{3m_\sigma^2} - \frac{\big[f''(\langle\sigma\rangle)\big]^4}{12m_\sigma^4} + \mathcal{O}(\langle\sigma\rangle^5)\Big\}.
\end{aligned} \tag{7.43}$$

Using the definition of f, we find

$$\big[f''(\langle\sigma\rangle)\big]^3 = (2bmg_\sigma^3\langle\sigma\rangle)^3 + 3(2bmg_\sigma^3\langle\sigma\rangle)^2(3cg_\sigma^4\langle\sigma\rangle^2) + \mathcal{O}(\langle\sigma\rangle^5) \tag{7.44}$$

$$\big[f''(\langle\sigma\rangle)\big]^4 = (2bmg_\sigma^3\langle\sigma\rangle)^4 + \mathcal{O}(\langle\sigma\rangle^5), \tag{7.45}$$

which inserted into (7.43) gives

$$16\pi^2 U_\mathrm{r} = (m_\sigma^\star)^4\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big) - \frac{1}{2}\Big\{-m_\sigma^2 f(\langle\sigma\rangle) + \frac{3}{2}\big[f''(\langle\sigma\rangle)\big]^2 - \frac{a^3+3a^2b}{3m_\sigma^2} - \frac{a^4}{12m_\sigma^4}\Big\}, \tag{7.46}$$

where we have defined the variables $a = 2bmg_\sigma^3\langle\sigma\rangle$ and $b = 3cg_\sigma^4\langle\sigma\rangle^2$. That the shift in the vacuum energy is given by the fifth order correction to the logarithms $\ln(\frac{m^\star}{m})$ and $\ln(\frac{m_\sigma^\star}{m_\sigma})$, is also reported in [44]. Combining this with the results from the previous section, we find

$$\epsilon_\mathrm{r} = \epsilon + V_\mathrm{r} + U_\mathrm{r} \qquad P_\mathrm{r} = P - V_\mathrm{r} - U_\mathrm{r} \tag{7.47}$$

and the self-consistency condition becomes

$$m^\star = m + \frac{g_\sigma^2}{m_\sigma^2}\Big[\rho_\mathrm{s} - bm(m-m^\star)^2 - c(m-m^\star)^3 - \frac{\partial V_\mathrm{r}}{\partial m^\star} - \frac{\partial U_\mathrm{r}}{\partial m^\star}\Big] \tag{7.48}$$

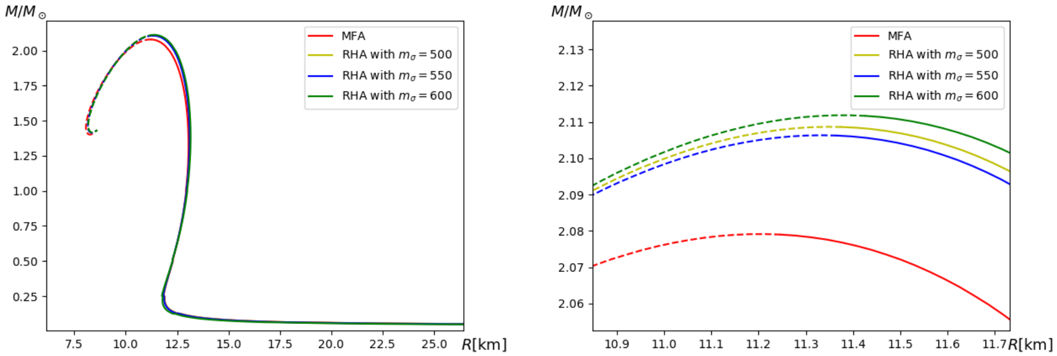


Figure 7.1: Mass-radius relation for the renormalized version of the improved σ–ω model (RHA) versus the regular mean-field approximation (MFA). The coupling constants are fitted for same properties of nuclear matter as in chapter 6, except that the compression modulus is set to 300 MeV. Right panel shows the same curves as the left, but is zoomed in near the maximum mass.

What remains is then to find the derivative of U_r, which is given by

$$\begin{aligned}16\pi^2\frac{\partial U_\mathrm{r}}{\partial m^\star} =&\frac{\partial\langle\sigma\rangle}{\partial m^\star}\frac{\partial m_\sigma^\star}{\partial\langle\sigma\rangle}\frac{\partial}{\partial m_\sigma^\star}\left[(m_\sigma^\star)^4\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big)\right]\\ &-\frac{1}{2}\frac{\partial\langle\sigma\rangle}{\partial m^\star}\frac{\partial}{\partial\langle\sigma\rangle}\left\{-m_\sigma^2 f''(\langle\sigma\rangle)+\frac{3}{2}\big[f''(\langle\sigma\rangle)\big]^2-\frac{a^3+a^2b}{3m_\sigma^2}-\frac{a^4}{12m_\sigma^4}\right\}\\ =&\frac{f'''(\langle\sigma\rangle)(m_\sigma^\star)^2}{g_\sigma}\left[4\ln\Big(\frac{m_\sigma^\star}{m_\sigma}\Big)+1\right]\\ &+\frac{1}{2g_\sigma}\left[-m_\sigma^2 f'''(\langle\sigma\rangle)+3f'''(\langle\sigma\rangle)f''(\langle\sigma\rangle)-\frac{a^3+4a^2b}{\langle\sigma\rangle m_\sigma^2}-\frac{a^4}{3\langle\sigma\rangle m_\sigma^4}\right],\end{aligned}\tag{7.49}$$

where we have used

$$\frac{\partial m_\sigma^\star}{\partial\langle\sigma\rangle}=\frac{\partial}{\partial\langle\sigma\rangle}\sqrt{m_\sigma^2-f''(\langle\sigma\rangle)}=-\frac{f'''(\langle\sigma\rangle)}{\sqrt{m_\sigma^2-f''(\langle\sigma\rangle)}}=-\frac{f'''(\langle\sigma\rangle)}{m_\sigma^\star}.\tag{7.50}$$

It is important to note that since

$$f'''(\langle\sigma\rangle)=2bmg_\sigma^3+6cg_\sigma^4\langle\sigma\rangle,\tag{7.51}$$

we are no longer able to express the EoS in terms of the ratio $\frac{g_\sigma}{m_\sigma}$. In other words, the mass of the σ-field is now relevant. Since the uncertainty in m_σ is quite large, we would be in trouble if the renormalized model differs much from the original model. This we will now investigate further.

7.3 Numerical solutions and summary

Figure 7.1 shows a plot of the original mass-radius relation, and the renromalized one for $m_\sigma = 500$, 550 and 600 MeV. From this it is clear that renormalization does not alter the EoS much, and we also see that the dependence of m_σ is small. It should be noted that we have here used the compression modulus $K = 300$ MeV. This is because it proved hard to find the coupling constants in the renormalized case. The convergence was much better for some values of the bulk properties of nuclear matter. Thus, to be able to find the couplings on my laptop within reasonable amount of computation time, I changed the compression modulus. However, the plot still proves an important point: Renormalization hardly matters in this model. Furthermore, renormalization does not strongly depend on the mass of the σ-meson, which is good considering the great uncertainty in this quantity. This result is similar to the one obtained by [44].

We have in this chapter looked at what happens to the infinite terms in the pressure and energy density that arose in the previous chapters. We used dimensional regularization to isolate the divergent terms, and added counterterms to the Lagrangian to absorb the infinities. We then found that the contribution to the EoS from the shift in the zero-point energy was small, and only affected the mass-radius relation by a few percent.

Chapter 8

Conclusions and outlook

In this master's thesis we have looked at models describing neutron star matter using the relativistic mean-field approximation. We started out by deriving Einsten's field equation for general relativity, which then was used to find the Tolmann-Oppenheimer-Volkoff equations. This equation describes the rate of change in pressure as we move along the radius of a spherically symmetric mass-distribution, given the equation of state. Using the path-integral formalism, we then looked at a model consisting of an ideal cold neutron gas which resulted in an upper mass limit of 0.7 solar masses in accordance with [19]. Then we introduced scalar and vector mesons to mimic the long-range attraction and short-range repulsion of the strong force analogous to the way Walecka and Chin did in [33]. This gave us two coupling constants between the mesons and the nucleons. These couplings serve as two free parameters of the model, allowing the fitting of two bulk properties of nuclear matter. However, the EoS was highly dependent on which properties we chose to fit the couplings, making the model inconsistent. For some couplings, it even predicts that neutron star matter is bound, which is not compatible with observations [6, p. 194-195]. This motivated the introduction of scalar self-interactions and an isospin asymmetry induced by the ρ-meson as first done by Botuga and Bodmer [42]. This resulted in a limiting mass just above two solar masses. It was then shown that renormalizing the model had little impact when the coupling constants were chosen to match the properties of nuclear matter.

8.1 Hyperons and the hyperon puzzle

Even though the improved σ–ω model provided us with a satisfactory limiting mass, this is not the complete picture. At densities above 2-3 times the saturation density of nuclear matter, it becomes energetically favourable for the Σ^- to replace a neutron and one of the leptons (the e or the μ) [44]. As the energy density increases even further, more massive particles becomes stable and if it is high enough, the whole baryon octet (n, p, Λ, Σ^-, Σ^0, Σ^+, Ξ^0 and Ξ^-) might be present. The six new baryons are strange particles that do not have any charm, bottom or top quarks.[1] The procedure for including these so called hyperons in the EoS is briefly discussed in Appendix F. To solve the equations, we need some coupling constants for the meson-hyperon interactions, which we denote g_{mB} where $m = \{\sigma, \omega, \rho\}$ and $B = \{\Lambda, \Sigma, \Xi\}$. Determining the hyperon couplings is not as straightforward as it was for the nucleons. After all, since they are not present until 2-3 times nucleon density, we cannot describe them using the properties of nuclear matter at saturation. We will now briefly discuss how we can solve this.

We start by defining the ratios

$$x_{\omega\mathrm{B}} = \frac{g_{\omega\mathrm{B}}}{g_\omega}, \qquad x_{\sigma\mathrm{B}} = \frac{g_{\sigma\mathrm{B}}}{g_\sigma}, \qquad x_{\rho\mathrm{B}} = \frac{g_{\rho\mathrm{B}}}{g_\rho}. \tag{8.1}$$

[1] A strange particle is constituent of one or more strange quarks. They are not strange in the normal sense of the word.

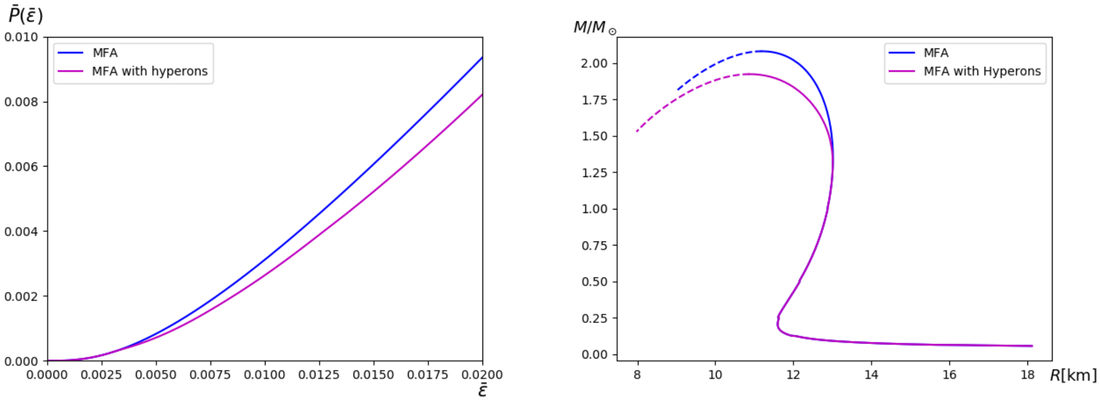


Figure 8.1: EoS and mass-radius relation for the improved σ–ω model with and without hyperons using the mean-field approximation.

According to [47], experimental data of the binding energy for each of the types of particles Λ, Σ and Ξ exist to some degree. By writing down the expression for the binding energies we find a relation between $x_{\sigma \mathrm{B}}$ and $x_{\omega \mathrm{B}}$ as shown in [6, p. 259-260]. This does not determine the couplings, but at least it sets some constraints on $g_{\omega \mathrm{B}}$ and $g_{\sigma \mathrm{B}}$. The ρ-coupling can be chosen using symmetry considerations [48]. However, we will start this discussion by setting

$$x_{\omega \mathrm{B}} = x_{\sigma \mathrm{B}} = x_{\rho \mathrm{B}} = 1, \qquad \text{for all B.} \tag{8.2}$$

There is no reasoning behind this assumption, but the result obtained will serve to illustrate the effect hyperons have on the EoS. Figure 8.1 shows a plot of the equation of state for the improved σ–ω model with and without hyperons, as well as the mass-radius relation for the two. We see that the hyperon EoS is softer and yields a lower maximum mass. This is a consequence of the Pauli principle. We have more types of fermions that can make up the density, which lowers the Fermi momentum for each species, thereby decreasing the overall pressure. Also, because the leptons have small masses compared to the baryons, it is expensive energy wise to have lepton densities large enough to maintain the charge neutrality condition. When the hyperons form, the leptons are no longer the only source of negative charge. The hyperons then overtake the responsibility as the enforcers of the charge neutrality, and the relative lepton densities are substantially lowered which we see from Figure 8.2.

The strong softening of the EoS imposes a serious problem. The improved σ–ω model predicted a maximum mass just above the most massive neutron star ever measured. Unless the hyperon effects are negligible, which there is no reason to believe they are, we cannot avoid that the maximum mass falls below the minimum 2.01 solar masses. However, invoking hyperons in the EoS seems unavoidable since the energy density in the core of the most massive neutron stars is way above three times the saturation density. This problem is today popularly known as the *hyperon puzzle*, and is still a hot topic in nuclear physics today [3]. We will now briefly discuss some of the possible resolutions to this issue.

1. There is no problem: we can fine tune the coupling constants

The simplest solution is to just fine tune the coupling constants. In an attempt to overcome the hyperon problem Zhao found in 2017 [47] by using constraints from known properties of hyper nuclei that the choices

$$x_{\sigma\Lambda} = 0.8, \qquad x_{\omega\Lambda} = 0.9319 \qquad x_{\sigma\Sigma} = 0.4 \qquad x_{\omega\Sigma} = 0.825, \qquad x_{\sigma\Xi} = 0.7, \qquad x_{\omega\Xi} = 0.804, \tag{8.3}$$

results in a mass compatible with observations. However, to achieve this, he sets the saturation density and binding energy to be $\rho_0 = 0.145\,\mathrm{fm}^{-3}$ and $B_0 = -15.95\,\mathrm{MeV}$, respectively. These values are

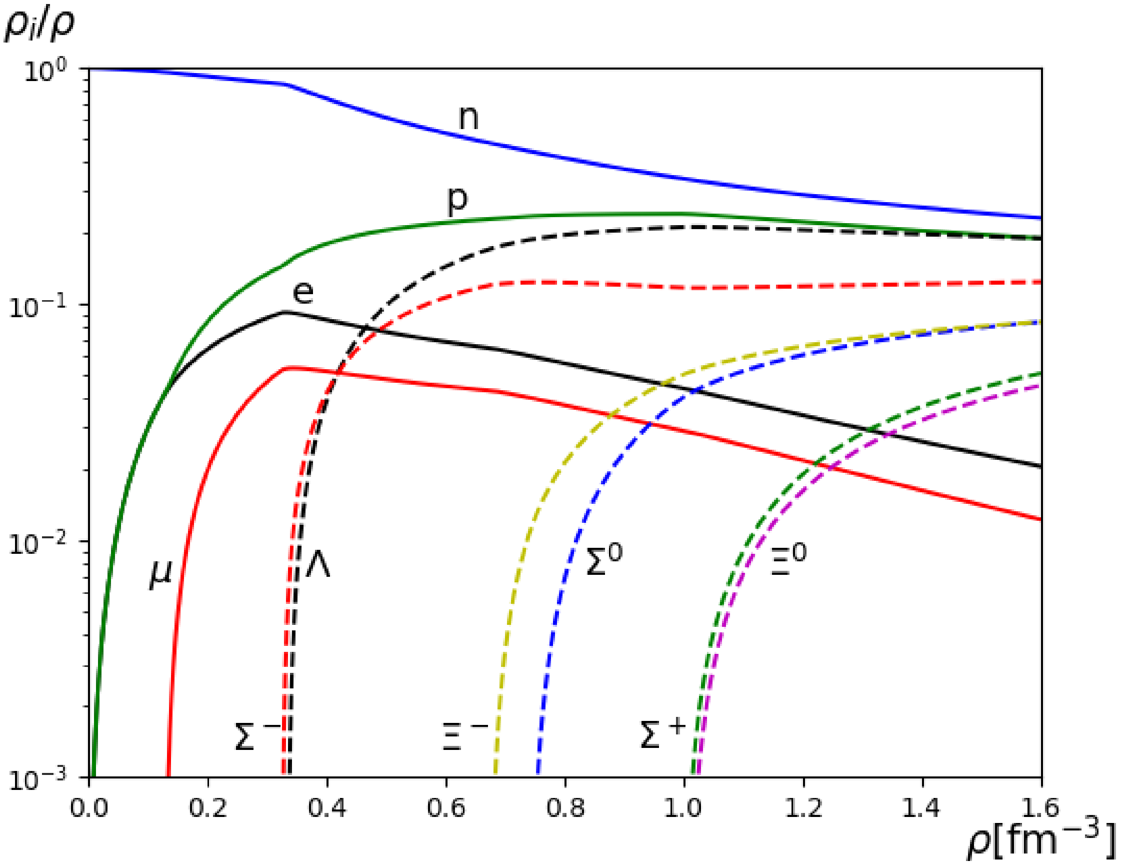


Figure 8.2: Relative population density ρ_i/ρ as a function of baryon density ρ for the improved σ–ω model when hyperons are included.

picked from Glendenning's article [49] published in 1985. Glendenning has in his later works such as [6] corrected these values to be $\rho_0 = 0.153\,\mathrm{fm}^{-3}$ and $B_0 = -16.3\,\mathrm{MeV}$ like the ones used here. I have not been able to reproduce a maximum mass above two solar masses with these saturation values, despite choosing the upper limit for the compression modulus 300 MeV.

Even if one should find it reasonable to set the saturation properties of nuclear matter the way Zhao did, there are still some issues with this fine tuning approach. As discussed in for instance [6, p. 268-271], it is likely that pions condense in neutron star matter. From (F.6) we find that

$$\mu_{\pi^-} = \mu_\mathrm{e}, \qquad \mu_{\pi^0} = 0, \qquad \mu_{\pi^+} = -\mu_\mathrm{e}, \tag{8.4}$$

which means that only the π^- will be present.[2] Since pions are bosons, they can condense in the ground state, making them much more energetically cheap than the leptons. The leptons are then no longer required to withhold the charge neutrality condition and the degeneracy pressure shrinks as the pions emerge. This lowers the neutron star's mass, opening for the possibility that the fine tuned couplings are not enough to satisfy the observed masses. Also, if a pion condensate is present, there is no reason why there should not be a kaon condensate as well, resulting in a stronger softening of the EoS.

2. Repulsive hyperon-hyperon interactions

Repulsive interactions stiffens the EoS. This is fairly intuitive. In equilibrium, a gas of particles that all repel one another takes up more space than one where the particles attract each other, resulting in a higher pressure. If there were some repulsive interactions between the hyperons, this could counter the softening of the EoS. A repulsive force between the hyperons themselves can be added by including the two hidden-strangeness mesons $\sigma^\star$ and ϕ as shown in for instance [50]. Another approach is to use density dependent couplings such as in [51]. Hopefully, one day high-energy experiments might give a clue whether or not this is the case.[3]

3. A phase transition to quark matter occurs

At the extreme densities inside the core of a neutron star, it is possible that the energy is so high that quarks no longer are confined to mesons and baryons. There would then be a phase transition to an exotic state known as quark matter [2]. In the relativistic mean-field approximation it is suggested that the transition from hadronic to quark matter is of first order [52]. This result is also supported using the framework of QCD [53]. If this is indeed the case, then there must be a coexistence of hadronic matter and quark matter at some density just as there is a coexistence of liquid and vapour during the phase transition of the Van der Waals gas as discussed in section 5.7.2.

8.2 Other effects that could be included

Lastly, we remark other effects that may impact the EoS. For one, we have throughout this thesis ignored the rotation of the neutron star. As mentioned in the introduction, the rotation should be extremely fast due to the conservation of angular momentum from the collapsing red giant. The rotation will make the star's radius smaller at the poles, and larger at the equator. Then our assumption of rotational symmetry at each point breaks down, since we now only have an axial symmetry. This should show up in the EoS as the pressure gradient will be dependent on the polar angle. Perhaps even more interestingly is how gravity changes as we include rotation. In such a picture, the TOV equation is no longer valid, and we would have to develop new equilibrium equations.

As we also mentioned in the introduction, neutron stars can have large magnetic fields. In general, the magnetic field does not align with the rotation axis [54]. This means that a complete description of

[2] This is only true while assuming beta equilibrium.

[3] It is well established that the coupling constants are not really constants at all. The question is if the density dependence of the couplings is large enough to make any difference on the energy scales we are working on.

neutron stars considering both rotation and a magnetic field, cannot make use of any spatial symmetry whatsoever, except perhaps a mirroring about the equatorial plane.

Appendix A

Notation and conventions

Units

Unless otherwise specified, natural units are used. Hence Boltzmann constant k_B, the reduced Planck constant $\hbar$ and the speed of light c are set to unity:

$$k_B = \hbar = c = 1. \tag{A.1}$$

Einsteins summation convention

In each term where two indices are repeated, summation is implied over all possible indices:

$$a^\mu b_\mu = \sum_{i=0}^{n} a^i b_i, \quad \mu \in [0, n]. \tag{A.2}$$

Greek indices are implied to be in the range $[0, 3]$ while Latin indices are in the range $[1, 3]$.

Differentiation

Differentiation of a function: Differentiation of a function f with respect to some variable a is written as

$$\frac{\partial f}{\partial a} = \partial_a f. \tag{A.3}$$

Should the function only be dependent of a variable r, Newton's notation may be used:

$$f'(r) = \partial_r f(r) \tag{A.4}$$

Differentiation of a tensor: Differentiation of a tensor $T_{\mu\nu}$ can be written with the comma convention:

$$\partial_\sigma T_{\mu\nu} = T_{\mu\nu,\sigma}. \tag{A.5}$$

Tensor notation

Trace: The trace of a tensor $T^{\mu\nu}$ is denoted either with equal lower and upper index, or with no index at all

$$\mathrm{Tr}[T] = T^\mu_\mu = T. \tag{A.6}$$

The context should make it clear whether T is the trace of a tensor or a scalar.

Determinant: The determinant of a tensor $A_{\mu\nu}$ is denoted

$$\det(A_{\mu\nu}) = |A|. \tag{A.7}$$

Metric tensor

The sign convention used for the metric tensor $g_{\mu\nu}$ is $(+,-,-,-)$. It is assumed that the metric tensor is symmetric and that the determinant $|g|$ is invariant under coordinate transformations. In this thesis the metric in flat space is mostly denoted by $\eta_{\mu\nu}$.

Matrices and numbers

If a number ω is summed with a matrix A, the number is assumed to be the identity matrix of same dimensions as A times ω. For example, if A is a 4×4-matrix, when we write

$$\omega + A, \tag{A.8}$$

what we really mean is

$$I_4\omega + A, \tag{A.9}$$

where I_4 is the 4×4-identity matrix.

Feynman slash notation

When writing the contraction of a four-vector A_μ with the gamma matrices γ^μ, the Feynman slash notation is used:

$$\gamma^\mu A_\mu = \not{A}. \tag{A.10}$$

Bar notation for spinors

For a spinor ψ we define the bar notation

$$\bar{\psi} = \gamma^0\psi^\dagger \tag{A.11}$$

where the dagger means complex transpose and γ^0 is given by (C.9)-(C.10).

Appendix B

Grassmann variables

Anti-commuting numbers, also called Grassman numbers [24, p. 299], have some special properties that will be derived here. For a Grassman number θ, we have by definition that the anti-commutator with itself vanishes:

$$0 = \{\theta, \theta\} = \theta\theta + \theta\theta = 2\theta^2. \tag{B.1}$$

If we now want to make a function out of θ, the most general differentiable form would be

$$f(\theta) = a + b\theta, \tag{B.2}$$

where a and b are constants, because all the higher order terms of the function's Taylor expansion would vanish.

Further, we would like to define some sort of integration over the Grassman variables. The integral

$$\int \mathrm{d}\theta f(\theta), \tag{B.3}$$

should be invariant under a shift of integration variable [24, p. 299]. Invariance during the shift $\theta \to \nu + \theta$ gives

$$\int \mathrm{d}\theta (a + b\theta) = \int \mathrm{d}\theta \left[(a + b\nu) + b\theta\right], \tag{B.4}$$

which changes the constant term, but leaves the first order term untouched. The only way this can happen is if the integral is a constant, which we chose to be b:

$$\int \mathrm{d}\theta (a + b\theta) \equiv b. \tag{B.5}$$

It then follows that

$$\begin{aligned} \int \mathrm{d}\theta a = \int \mathrm{d}\theta &= 0, \\ \int \mathrm{d}\theta\theta &= 1. \end{aligned} \tag{B.6}$$

Appendix C

Tensor definitions

Riemann tensor

The Riemann tensor defines how the curvature behaves on a manifold and is given by

$$R^{\rho}_{\sigma\mu\nu} = \Gamma^{\rho}_{\nu\sigma,\mu} - \Gamma^{\rho}_{\mu\sigma,\nu} + \Gamma^{\rho}_{\mu\lambda}\Gamma^{\lambda}_{\nu\sigma} - \Gamma^{\rho}_{\nu\lambda}\Gamma^{\lambda}_{\mu\sigma}, \tag{C.1}$$

where $\Gamma^{\rho}_{\mu\nu}$ are the Christoffel symbols.

Christoffel symbols

Assuming that the metric is torsion free, the Christoffel symbols are uniquely defined as

$$\Gamma^{\rho}_{\mu\nu} = \frac{1}{2} g^{\rho\sigma} \left(g_{\sigma\mu,\nu} + g_{\sigma\nu,\mu} - g_{\mu\nu,\sigma} \right). \tag{C.2}$$

The Ricci tensor

The Ricci tensor is a special case of the Riemann tensor and is defined as

$$R_{\mu\nu} = R^{\rho}_{\mu\rho\nu,\rho} = \Gamma^{\rho}_{\mu\nu,\rho} - \Gamma^{\rho}_{\rho\mu,\nu} + \Gamma^{\rho}_{\rho\sigma}\Gamma^{\sigma}_{\mu\nu} - \Gamma^{\rho}_{\nu\sigma}\Gamma^{\sigma}_{\rho\mu}. \tag{C.3}$$

The Ricci scalar

The Ricci scalar is a number associated with how much the volume of a geodesic ball in the Riemann manifold deviates from a ball in the Euclidean space. It is defined as

$$R = g^{\mu\nu} R_{\mu\nu}. \tag{C.4}$$

The stress-energy tensor

The stress-energy tensor describes the density and flux of energy and momentum in space-time and is by definition

$$T_{\mu\nu} = \kappa \left(\mathscr{L}_{\mathrm{M}} g_{\mu\nu} - 2 \frac{\delta \mathscr{L}_{\mathrm{M}}}{\delta g^{\mu\nu}} \right), \tag{C.5}$$

where $\mathscr{L}_{\mathrm{M}}$ is the matter Lagrangian, while $\kappa = 8\pi G$ and G is Newton's gravitational constant.

Pauli matrices

The Pauli matrices are a set of 2×2 hermitian and unitary matrices σ_i defined by

$$\sigma_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{C.6}$$

These matrices satisfy the anti-commutation relation

$$\{\sigma_i, \sigma_j\} = \delta_{ij} I_2, \tag{C.7}$$

where I_2 is the 2×2 identity matrix.

Gamma matrices

The γ-matrices γ^μ are defined so that they satisfy the anti-commutation relation

$$\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu}, \tag{C.8}$$

where $\eta^{\mu\nu}$ is the metric tensor for flat space. In Minkowski space, we have in the Dirac basis

$$\gamma^0 = \begin{pmatrix} I_2 & 0 \\ 0 & -I_2 \end{pmatrix}, \quad \gamma^i = \begin{pmatrix} 0 & \sigma^i \\ -\sigma^i & 0 \end{pmatrix}. \tag{C.9}$$

In Euclidean space it is often convention to use

$$\gamma^0_{\mathrm{E}} = \gamma^0 \quad \gamma^i_{\mathrm{E}} = -\mathrm{i}\gamma^i, \tag{C.10}$$

where I_2 denotes the 2×2 identity matrix.

Levi-Civita-symbol

The Levi-Civita-symbol is defined as a tensor $\epsilon_{\dots i_p \dots i_q}$ that is anti-symmetric in all of its indices. This means that the tensor changes sign each time we switch two indices, so that

$$\epsilon_{\dots i_p \dots i_q} = -\epsilon_{\dots i_q \dots i_p}. \tag{C.11}$$

Should any of the indices be equal, then the symbol is zero.

Appendix D

Other definitions

Matsubara frequencies

The Matsubara frequencies are assumed in the fourer expansion of bosonic(fermionic) fields so that they become periodic(antiperiodic), and is defined by

$$\omega_n = \begin{cases} 2\pi n T & \text{bosons,} \\ (2n+1)\pi T & \text{fermions,} \end{cases} \tag{D.1}$$

where n is an integer between $-\infty$ and ∞.

Gamma function

The gamma function is defined as a generalization of the factorial function. It is defined by the two properties

$$\Gamma(1) = 1, \qquad x\Gamma(x) = \Gamma(x+1), \tag{D.2}$$

and can be expressed as the integral

$$\Gamma(x) = \int_0^\infty \mathrm{d}z\, z^{x-1} \mathrm{e}^{-z}. \tag{D.3}$$

Beta function

The beta function is defined by the property

$$\beta(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}, \tag{D.4}$$

and can be expressed as the integral

$$\beta(x,y) = \int_0^\infty \mathrm{d}z\, \frac{z^{x-1}}{(1+z)^{x+y}}. \tag{D.5}$$

Appendix E

Useful theorems

The Euler-Lagrange equation

The Euler-Lagrange equation is derived in most textbooks on classical mechanics and field theory, for instance [16, p. 119-124]. Here we only give a brief overview. Hamilton's principle states that if a system does a transition from a state at $t = t_0$ to another state at $t = t_1$, the system would take the path that extremizes the action

$$S = \int_{t_0}^{t_1} \mathrm{d}t L[q, \dot{q}], \tag{E.1}$$

where L is the systems Lagrangian and q and $\dot{q}$ is some coordinate and its time derivative, respectively. This is the same as saying that the variation should be zero

$$0 = \delta S = \int_{t_0}^{t_1} \mathrm{d}t \bigg(\frac{\partial L[q, \dot{q}]}{\partial q} \delta q + \frac{\partial L[q, \dot{q}]}{\partial \dot{q}} \delta \dot{q} \bigg). \tag{E.2}$$

By definition, there is no variation at the boundaries, and so an integration by parts on the second term gives

$$0 = \int_{t_0}^{t_1} \mathrm{d}t \bigg(\frac{\partial L[q, \dot{q}]}{\partial q} \delta q - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L[q, \dot{q}]}{\partial \dot{q}} \delta q \bigg) = \int_{t_0}^{t_1} \mathrm{d}t \bigg(\frac{\partial L[q, \dot{q}]}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L[q, \dot{q}]}{\partial \dot{q}} \bigg) \delta q, \tag{E.3}$$

where we have used that

$$\delta \dot{q} = \frac{\mathrm{d}}{\mathrm{d}t} \delta q. \tag{E.4}$$

The variation δq is arbitrary and hence we arrive at the Euler-Lagrange equation

$$\frac{\partial L[q, \dot{q}]}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial L[q, \dot{q}]}{\partial \dot{q}} = 0. \tag{E.5}$$

In the same manner, one finds that for a field $\phi(x)$ with Lagrangian density $\mathscr{L}[\phi(x), \partial_\mu \phi(x)]$, the Euler-Lagrange equation becomes [6, p. 151]

$$\frac{\partial \mathscr{L}}{\partial \phi(x)} - \partial_\mu \frac{\partial \mathscr{L}}{\partial(\partial_\mu \phi(x))} = 0. \tag{E.6}$$

Noether's theorem

Noether's theorem is derived in most textbooks on quantum field theory, for instance [24]. The theorem states that for every continuous symmetry the Lagrangian of a system has, there is a corresponding conserved current. If the Lagrangian density $\mathcal{L}(\phi_a)$ is invariant under the transformation $\phi_a \to \phi_a + \delta\phi_a$ then the conserved current is given by

$$j^\mu = \frac{\delta \mathcal{L}}{\delta(\partial_\mu \phi_a)} \delta\phi_a. \tag{E.7}$$

If we define the charge

$$Q = \int \mathrm{d}^3 x j^0, \tag{E.8}$$

we see that by integrating the conservation equation

$$\partial_\mu j^\mu = 0, \tag{E.9}$$

over three-space, results in

$$0 = \int \mathrm{d}^3 x \; \partial_\mu j^\mu = \int \mathrm{d}^3 x \left(\frac{\mathrm{d}}{\mathrm{d}t} j^0 + \boldsymbol{\nabla} \cdot \boldsymbol{j} \right) = \frac{\mathrm{d}}{\mathrm{d}t} \int \mathrm{d}^3 x \; j^0 + \int_S \boldsymbol{j} \cdot \mathrm{d}\boldsymbol{S}. \tag{E.10}$$

Assuming that $\boldsymbol{j}$ goes to zero sufficiently fast enough as $\boldsymbol{x}$ goes to infinity so that the surface integral vanish, we find that the charge Q is conserved.

Appendix **F**

Including hyperons in the improved σ–ω model

We will here develop the equation of state as well as the the constraints that follow from including hyperons in the improved σ–ω model. The steps involved are completely analogous to ones we did in chapter 6. Therefore we will here settle with a brief overview of the derivation.

In order to account for the effects of the hyperons, we generalize the Lagrangian (6.48) by writing

$$\begin{aligned}\mathscr{L} =& \sum_{\mathrm{B}} \Big[\bar{\psi}_{\mathrm{B}}\big(\mathrm{i}\partial\!\!\!/ - m_{\mathrm{B}}\big)\psi_{\mathrm{B}} + g_{\sigma\mathrm{B}}\sigma\bar{\psi}_{\mathrm{B}}\psi_{\mathrm{B}} - g_{\omega\mathrm{B}}\omega_\mu\bar{\psi}_{\mathrm{B}}\gamma^\mu\psi_{\mathrm{B}} - \frac{1}{2}g_{\rho\mathrm{B}}\rho_{i\mu}\bar{\psi}_{\mathrm{B}}\gamma^\mu\tau_i\psi_{\mathrm{B}} + \mu_{\mathrm{B}}\psi^\dagger\psi\Big] \\ & - \frac{1}{2}m_\sigma^2\sigma^2 + \frac{1}{2}m_\omega^2\omega_\mu\omega^\mu + \frac{1}{2}m_\rho^2\rho_{i\mu}\rho^{i\mu} + bm\big(g_\sigma\sigma\big)^3 + c\big(g_\sigma\sigma\big)^4 + \sum_{i=\mathrm{e},\mu}\bar{\psi}_i\big(\mathrm{i}\partial\!\!\!/ - m_i + \mu_i\gamma^0\big)\psi_i. \quad \text{(F.1)}\end{aligned}$$

Here B runs over all the particles in the baryon octet. The partition function in the mean-field approximation changes accordingly to

$$\begin{aligned}\frac{1}{\beta V}\ln Z =& -\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle^2 + \frac{1}{2}m_\rho^2\langle\rho_{30}\rangle^2 + \frac{1}{3}bm(g_\sigma\langle\sigma\rangle)^3 + \frac{1}{4}c(g_\sigma\langle\sigma\rangle)^4 \\ & + \sum_{i=\mathrm{B},\mathrm{l}}\frac{1}{24\pi^2}\left\{\sqrt{k_i^2+(m_i^\star)^2}\Big[2k_i^3 - 3(m_i^\star)^3\Big] + 3(m_i^\star)^4\ln\left[\frac{k_i+\sqrt{k_i^2+(m_i^\star)^2}}{m_i}\right]\right\}, \qquad \text{(F.2)}\end{aligned}$$

where l vary over all lepton species and $m^\star$ is the effective mass of the particle of species i.

Before we continue, we remark that τ_3 gives us the third component of the isospin projection for the baryon involved. As an example, we saw that for the nucleons

$$\langle\rho_{30}\rangle = \frac{g_\rho}{2m_\rho^2}\bar{\psi}\gamma^0\tau_3\psi = \frac{g_\rho}{m_\rho^2}(\rho_{\mathrm{n}} - \rho_{\mathrm{p}}) = \frac{g_\rho}{m_\rho^2}\big(\rho_{\mathrm{n}}I_{3\mathrm{n}} + \rho_{\mathrm{p}}I_{3\mathrm{p}}\big), \qquad \text{(F.3)}$$

where $I_{3\mathrm{B}}$ is the isospin projection ($\frac{1}{2}$ for the neutron and $-\frac{1}{2}$ for the proton) for particle species B, along its third component. If we as an approximation set $m_{\Sigma^-} = m_{\Sigma^0} = m_{\Sigma^+} \equiv m_\Sigma$ and $m_{\Xi^0} = m_{\Xi^-} \equiv m_\Xi$, we find an isospin symmetry[1] that allows us to write

$$\langle\rho_{30}\rangle = \sum_{\mathrm{B}}\frac{g_{\rho\mathrm{B}}}{m_\rho^2}I_{3\mathrm{B}}\rho_{\mathrm{B}}. \qquad \text{(F.4)}$$

[1] This is the exact same thing we did for the neutron and the proton. For instance we picture that the Σ particle has a total isospin one with three isospin states: Σ^- ($I_3 = 1$), Σ^0 ($I_3 = 0$) and Σ^+ ($I_3 = -1$).

By the same reasoning as in section 6.4, the effective chemical potential for each baryon is then

$$\mu_{\mathrm{B}}^{\star} = \mu_{\mathrm{B}} - g_{\omega \mathrm{B}}\langle\omega_0\rangle - g_{\rho \mathrm{B}} I_{3\mathrm{B}}\langle\rho_{30}\rangle = \sqrt{k_{\mathrm{B}}^2 + (m_{\mathrm{B}}^{\star})^2}. \tag{F.5}$$

Still assuming beta equilibrium, the general expression for the chemical potential is given by[2] [55]

$$\mu_{\mathrm{B}} = b_{\mathrm{B}}\mu_{\mathrm{n}} - q_{\mathrm{B}}\mu_{\mathrm{e}}. \tag{F.6}$$

Here, q_{B} is the charge of the baryon, and b_{B} is the baryon number defined by

$$b_{\mathrm{B}} = \frac{1}{3}(n_{\mathrm{q}} + \bar{n}_{\mathrm{q}}), \tag{F.7}$$

where n_{q} and $\bar{n}_q$ denotes the number of quarks and anti quarks inside the baryon, respectively. We immediately see that the formula (F.6) is consistent with the chemical potential used for the proton in (6.44) by inserting plus one for both baryon number and proton charge.

All baryons concerned have plus one as their baryon number. Inserting (F.6) into (F.5) then yields

$$\mu_{\mathrm{n}} - q_{\mathrm{B}}\mu_{\mathrm{e}} - g_{\omega \mathrm{B}}\langle\omega_0\rangle - g_{\rho \mathrm{B}} I_{3\mathrm{B}}\langle\rho_{30}\rangle = \sqrt{k_{\mathrm{B}}^2 + (m_{\mathrm{B}}^{\star})^2} \tag{F.8}$$

which determines the Fermi momentum for each baryon

$$k_{\mathrm{B}}^2 = (\mu_{\mathrm{n}} - q_{\mathrm{B}}\mu_{\mathrm{e}} - g_{\omega \mathrm{B}}\langle\omega_0\rangle - g_{\rho \mathrm{B}} I_{3\mathrm{B}}\langle\rho_{30}\rangle)^2 - (m_{\mathrm{B}}^{\star})^2. \tag{F.9}$$

Of course, this equation is only valid when it is energetically favourable for the baryon B to be stable, which means that we only consider real solutions for k_{B}.

The expectation value of the ω-field is dependent on the baryon density, so it has to be modified

$$\langle\omega_0\rangle = \sum_{\mathrm{B}} \frac{g_{\omega \mathrm{B}}}{m_\omega^2}\rho_{\mathrm{B}}. \tag{F.10}$$

Furthermore, we should add the scalar densities for each baryon to the self-consistency equation so that

$$g_\sigma\langle\sigma\rangle = \frac{g_\sigma^2}{m_\sigma^2}\left[bm(g_\sigma\langle\sigma\rangle)^2 + c(g_\sigma\langle\sigma\rangle)^3 + \frac{1}{g_\sigma}\sum_{\mathrm{B}} g_{\sigma \mathrm{B}}\rho_{\mathrm{sB}}\right], \tag{F.11}$$

where ρ_{sB} is given by

$$\rho_{\mathrm{sB}} = \frac{m^\star}{2\pi^2}\left[k_{\mathrm{B}}\sqrt{k_{\mathrm{B}}^2 + (m_{\mathrm{B}}^\star)^2} - (m_{\mathrm{B}}^\star)^2 \ln\left(\frac{\sqrt{k_{\mathrm{B}}^2 + (m_{\mathrm{B}}^\star)^2} + k_{\mathrm{B}}}{m_{\mathrm{B}}^\star}\right)\right]. \tag{F.12}$$

Now that all the equations of motion are established, we close our system by imposing the global charge neutrality constraint, which gives us

$$\sum_{\mathrm{B}} q_{\mathrm{B}}\rho_{\mathrm{B}} + \sum_{\mathrm{l}} q_{\mathrm{l}}\rho_{\mathrm{l}} = 0, \tag{F.13}$$

where q_{l} and ρ_{l} are the lepton charges and densities respectively.

We are now ready to compute the EoS. The pressure and energy density will in the mean-field approximation have the same form as we saw in the case of the improved σ–ω model, except that we

[2] This result holds for mesons as well.

must sum over all the hyperons as well:

$$\begin{aligned} P =& -\frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle^2 + \frac{1}{2}m_\rho^2\langle\rho_{30}\rangle^2 + \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 + \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4 \\ &+ \sum_{i=\mathrm{l,B}} \frac{1}{24\pi^2}\left\{\sqrt{k_i^2+(m_i^\star)^2}\Big[2k_i^3 - 3(m_i^\star)^2 k_i\Big] + 3(m_i^\star)^4 \ln\left[\frac{k_i+\sqrt{k_i^2+(m_i^\star)^2}}{m_i^\star}\right]\right\}, \end{aligned} \tag{F.14}$$

$$\begin{aligned} \epsilon =& \frac{1}{2}m_\sigma^2\langle\sigma\rangle^2 + \frac{1}{2}m_\omega^2\langle\omega_0\rangle^2 + \frac{1}{2}m_\rho^2\langle\rho_{30}\rangle^2 - \frac{1}{3}bm\big(g_\sigma\langle\sigma\rangle\big)^3 - \frac{1}{4}c\big(g_\sigma\langle\sigma\rangle\big)^4 \\ &+ \sum_{i=\mathrm{l,B}} \frac{1}{24\pi^2}\left\{\sqrt{k_i^2+(m_i^\star)^2}\Big[6k_i^3 + 3(m_i^\star)^2 k_i\Big] - 3(m_i^\star)^4 \ln\left[\frac{k_i+\sqrt{k_i^2+(m_i^\star)^2}}{m_i^\star}\right]\right\}. \end{aligned} \tag{F.15}$$

These are the equations required to compute the mass-radius relation for a given set of coupling constants $g_{\sigma\mathrm{B}}$ and $g_{\omega\mathrm{B}}$. A program for this purpose is given in Appendix G.3.

Appendix G

Code

G.1 Chapter 3 and 4

```python
import numpy as np
import matplotlib.pyplot as plt
import time
from numba import jit
from scipy import optimize as op
import math

##########################################################
##########################################################
###                                                    ###
###   Just press play to create all plots in thesis!   ###
###                                                    ###
##########################################################
##########################################################

t0 = time.clock() #Timing the program

#################################
#################################
###                           ###
###   Defining functions      ###
###                           ###
#################################
#################################


#################################
#################################
###                           ###
###   Non-relativistic case   ###
###                           ###
#################################
#################################

#########################################
#Pressure for constant density using TOV#
#########################################
@jit
def PconstTOV(r,M,R):
    mem1 = np.sqrt(1.-2*M*r**2/R**3)
    mem2 = np.sqrt(1.-2*M/R)
    return 3*M/(4*math.pi*R**3)*(mem1-mem2)/(3*mem2-mem1)

```

```
############################################
#Pressure for constant density using newton#
############################################
@jit
def PconstDensNewton(r,M,R):
    return 3*M**2/(8*math.pi*R**6)*(R**2-r**2)

##########################################################
#Plot the figures regarding constant density in chapter 3#
##########################################################
def plotConstPressure():
    N = 1000
    M = 1.
    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1.05)
    ax.set_xlabel('$r/R$',fontsize = 14)
    ax.set_ylabel('$P(r)/P_0$', rotation='horizontal', fontsize = 14)
    plt.ylim(-1.5,1.5)
    plt.xlim(0,1)
    R = 2.3*M
    r = np.linspace(0,R,N)
    P = PconstTOV(r,M,R)
    P0 = P[0]
    print(P0)
    plt.plot(r/R,P/P0,'g',label = '$R=2.30MG$')
    plt.plot(r/R,np.zeros(len(P)),'k--')
    R = 2.2*M
    r = np.linspace(0,R,N)
    P = PconstTOV(r,M,R)
    for i in range(1,len(P)):
        if(np.sign(P[i-1])!=np.sign(P[i])):
            index = i
    plt.plot(r[0:index]/R,P[0:index]/P0,'b',label = '$R=2.20MG$')
    plt.plot(r[index:len(P)]/R,P[index:len(P)]/P0,'b')
    R = 2.25*M
    r = np.linspace(0,R,N)
    P = PconstTOV(r,M,R)
    plt.plot(r/R,P/P0,'r',label = '$R=2.25MG$')


    plt.legend()

    plt.figure()
    R = 2.3*M
    r = np.linspace(0,R,N)
    P = PconstTOV(r,M,R)
    plt.xlim(0,1)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1.05)
    ax.text(0.2,0.4*10**(-1),'$R=2.3MG$',fontsize=14)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    ax.set_xlabel('$r/R$',fontsize = 14)
    ax.set_ylabel('$P(r)/P_0$', rotation='horizontal', fontsize = 14)
    plt.plot(r/R,P/P0,'r',label = 'TOV')
    P = PconstDensNewton(r,M,R)
    plt.plot(r/R,P/P0,'b',label = 'Newton')
    plt.legend()

    R = 10*M
    r = np.linspace(0,R,N)

    plt.figure()
    plt.xlim(0,1)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1.05)
    ax.text(0.2,0.4*10**(-5),'$R=10MG$',fontsize=14)
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    ax.set_xlabel('$r/R$',fontsize = 14)
    ax.set_ylabel('$P(r)/P_0$', rotation='horizontal', fontsize = 14)
    P = PconstTOV(r,M,R)
    plt.plot(r/R,P/P0,'r',label = 'TOV')
    P = PconstDensNewton(r,M,R)
    plt.plot(r/R,P/P0,'b',label = 'Newton')
    plt.legend()

    R = 100*M
    r = np.linspace(0,R,N)

    plt.figure()
    plt.xlim(0,1)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1.05)
    ax.text(0.2,0.4*10**(-9),'$R=100MG$',fontsize=14)
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    ax.set_xlabel('$r/R$',fontsize = 14)
    ax.set_ylabel('$P(r)/P_0$', rotation='horizontal', fontsize = 14)
    P = PconstTOV(r,M,R)
    plt.plot(r/R,P/P0,'r',label = 'TOV')
    P = PconstDensNewton(r,M,R)
    plt.plot(r/R,P/P0,'b',label = 'Newton')
    plt.legend()

    R = 1000*M
    r = np.linspace(0,R,N)

    plt.figure()
    plt.xlim(0,1)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1.05)
    ax.text(0.2,0.4*10**(-13),'$R=1000MG$',fontsize=14)
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    ax.set_xlabel('$r/R$',fontsize = 14)
    ax.set_ylabel('$P(r)/P_0$', rotation='horizontal', fontsize = 14)
    P = PconstTOV(r,M,R)
    plt.plot(r/R,P/P0,'r',label = 'TOV')
    P = PconstDensNewton(r,M,R)
    plt.plot(r/R,P/P0,'b',label = 'Newton')
    plt.legend()

############################################################################
#The derivative of the mass with respect to r in the non-relativistic limit#
############################################################################
@jit
def dMdrNonRel(r,P):
    if(P<0):
        return 0.
    return betaNonRel*r**2*P**(3/5)

############################################################################
#Derivative of the pressure with respect to r for the non-relativistic case#
#(TOV-equation)                                                            #
############################################################################
@jit
def dPdrTOVNonRel(r,P,M):
    if(P<0):
        return 0.
    return -alpha*P**(3/5)*r**(-2)*(1+P**(2/5)*K**(-1))*(
            (M+betaNonRel*r**3*P*K**(-1))*(1-2*R0*M*r**(-1))**(-1))

########################################################################
#Derivative of the mass with respect to r for the non-relativistic case#
#(TOV-equation)                                                        #
########################################################################
@jit
def dPdrNewtonNonRel(r,P,M):
    if(P<0):
        return 0.
    return -alpha*P**(3/5)*M*r**(-2)

#############################################################
#Creates the coefficients for the Runge Kutta routine in the#
#non-relativistic case using the TOV-equation               #
#############################################################
@jit
def kTOVNonRel(r,P,M):
    kP1 = dPdrTOVNonRel(r,P,M)
    kM1 = dMdrNonRel(r,P)
    kP2 = dPdrTOVNonRel(r+h/2,P+h/2*kP1,M+h/2*kM1)
    kM2 = dMdrNonRel(r+h/2,P+h/2*kP1)
    kP3 = dPdrTOVNonRel(r+h/2,P+h/2*kP2,M+h/2*kM2)
    kM3 = dMdrNonRel(r+h/2,P+h/2*kP2)
    kP4 = dPdrTOVNonRel(r+h,P+h*kP3,M+h*kM3)
    kM4 = dMdrNonRel(r+h,P+h*kP3)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4

#############################################################
#Creates the coefficients for the Runge Kutta routine in the#
#non-relativistic case using Newton's equation              #
#############################################################
@jit
def kNewNonRel(r,P,M):
    kP1 = dPdrNewtonNonRel(r,P,M)
    kM1 = dMdrNonRel(r,P)
    kP2 = dPdrNewtonNonRel(r+h/2,P+h/2*kP1,M+h/2*kM1)
    kM2 = dMdrNonRel(r+h/2,P+h/2*kP1)
    kP3 = dPdrNewtonNonRel(r+h/2,P+h/2*kP1,M+h/2*kM2)
    kM3 = dMdrNonRel(r+h/2,P+h/2*kP2)
    kP4 = dPdrNewtonNonRel(r+h,P+h*kP3,M+h*kM3)
    kM4 = dMdrNonRel(r+h,P+h*kP3)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4

###########################################################
#Function that calculates the mass and radi of a star with#
#given central pressure in the non-relativistic case      #
#Also returns a flag indicating true if                   #
#maximum iterations is reached                            #
###########################################################
@jit
def resultsNonRel(Pc,h,nMax):
    PTOV = Pc
    PNew = Pc
    MTOV = 0.
    MNew = 0.
    rTOV = 0.
    rNew = 0.
    flagNew = False
    flagTOV = False
    for i in range(nMax):
        MemoryMTOV = MTOV
        MemoryMNew = MNew
        if(flagTOV == False):
            rTOV=rTOV+h
            (kPTOV1,kPTOV2,kPTOV3,kPTOV4,kMTOV1,kMTOV2,kMTOV3,kMTOV4) = kTOVNonRel(rTOV,PTOV,MTOV)
            PTOV = PTOV + h/6*(kPTOV1+2*kPTOV2+2*kPTOV3+kPTOV4)
            MTOV = MTOV + h/6*(kMTOV1+2*kMTOV2+2*kMTOV3+kMTOV4)

        if(flagNew == False):
            rNew = rNew+h
            (kPNew1,kPNew2,kPNew3,kPNew4,kMNew1,kMNew2,kMNew3,kMNew4) = kNewNonRel(rNew,PNew,MNew)

            PNew = PNew + h/6*(kPNew1+2*kPNew2+2*kPNew3+kPNew4)
            MNew = MNew + h/6*(kMNew1+2*kMNew2+2*kMNew3+kMNew4)

        if(np.float((np.real(PTOV))<=0 or MTOV==MemoryMTOV) and flagTOV == False):
            flagTOV = True

        if(np.float((np.real(PNew))<=0 or MemoryMNew == MNew) and flagNew == False):
            flagNew = True

        if(flagNew == True and flagTOV == True):
            break
    if(flagNew == False):
        print("Maximum number of iterations reached")
        print("for Newton-equation with Pc = %.8f \n"%Pc)
    if(flagTOV == False):
        print("Maximum number of iterations reached")
        print("for TOV-equation with Pc = %.8f \n"%Pc)
    return rTOV,MTOV,rNew,MNew,flagTOV,flagNew



############################################################################
#Returns vectors containing the radii and mass for central pressures within#
#the range given in the non-relatiistic case                               #
############################################################################
@jit
def paramterisingNonRel(PcMin,PcMax,N,h,nMax):
    Pc = PcMin
    const = (PcMax/PcMin)**(np.float(1)/N)
    RTOV = np.zeros(N)
    RNew = np.zeros(N)
    MTOV = np.zeros(N)
    MNew = np.zeros(N)
    for i in range(N):
        RTOV[i],MTOV[i],RNew[i],MNew[i],flagTOV,flagNew = resultsNonRel(Pc,h,nMax)
        if(flagTOV == False or flagNew == False or Pc > PcMax or Pc > PcMax):
            RTOV = RTOV[0:i]
            MTOV = MTOV[0:i]
            RNew = RNew[0:i]
            MNew = MNew[0:i]
            N = i
            break
        Pc = Pc*const
    return RTOV,MTOV,RNew,MNew,N

###########################################################
#Writes all the non-relativistic mass-radii data to a file#
###########################################################
def writeResultsToFileNonRel(PcMin,PcMax,N,h,nMax,filename):
    RTOV,MTOV,RNew,MNew,N = paramterisingNonRel(PcMin,PcMax,N,h,nMax)
    f = open(filename,'w')
    f.write(str(N)+'\n')
    for i in range(N):
        a = str(RTOV[i])
        b = str(MTOV[i])
        c = str(RNew[i])
```

```python
        d = str(MNew[i])
        f.write(a+" "+b+" "+c+" "+d +"\n")
    f.close()
    return

########################################################################
#Reads the data from file and returns four vectors containing the      #
#mass-radius relation for both the TOV-equation and Newton's equation  #
#in the non-relativistic case                                          #
########################################################################
def readResultsFromFileNonRel(filename):
    f = open(filename,'r')
    N = int(f.readline())
    RTOV = np.zeros(N)
    MTOV = np.zeros(N)
    RNew = np.zeros(N)
    MNew = np.zeros(N)
    i = 0
    data = f.readlines()
    for line in data:
        numbers = line.split()
        RTOV[i] = numbers[0]
        MTOV[i] = numbers[1]
        RNew[i] = numbers[2]
        MNew[i] = numbers[3]
        i = i+1
    f.close()
    return RTOV,MTOV,RNew,MNew

###################################
###################################
###                             ###
###   Ultra-relativistic case   ###
###                             ###
###################################
###################################


##############################################################################
#The derivative of the mass with respect to r in the ultra-relativistic limit#
##############################################################################
@jit
def dMdrUltraRel(r,P):
    return betaUltraRel*r**2*P

############################################################################
#Derivative of the pressure with respect to r for the non-relativistic case#
#(TOV-equation)                                                            #
############################################################################
@jit
def dPdrTOVUltraRel(r,P,M):
    return -4*R0*P*r**(-2)*(M+betaUltraRel*P*r**3/3)*(1-2*R0*M*r**(-1))**(-1)

############################################################################
#Derivative of the pressure with respect to r for the non-relativistic case#
#Newton's equation                                                         #
############################################################################
@jit
def dPdrNewtonUltraRel(r,P,M):
    return -3*R0*P*M*r**(-2)

#############################################################
#Creates the coefficients for the Runge Kutta routine in the#
#ultra-relativistic case using the TOV-equation             #
#############################################################
@jit
```

```
def kTOVUltraRel(r,P,M,h):
    kP1 = dPdrTOVUltraRel(r,P,M)
    kM1 = dMdrUltraRel(r,P)
    kP2 = dPdrTOVUltraRel(r+h/2,P+h/2*kP1,M+h/2*kM1)
    kM2 = dMdrUltraRel(r+h/2,P+h/2*kP1)
    kP3 = dPdrTOVUltraRel(r+h/2,P+h/2*kP2,M+h/2*kM2)
    kM3 = dMdrUltraRel(r+h/2,P+h/2*kP2)
    kP4 = dPdrTOVUltraRel(r+h,P+h*kP3,M+h*kM3)
    kM4 = dMdrUltraRel(r+h,P+h*kP3)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4

#############################################################
#Creates the coefficients for the Runge Kutta routine in the#
#ultra-relativistic case using the Newton's equation        #
#############################################################
@jit
def kNewUltraRel(r,P,M,h):
    kP1 = dPdrNewtonUltraRel(r,P,M)
    kM1 = dMdrUltraRel(r,P)
    kP2 = dPdrNewtonUltraRel(r+h/2,P+h/2*kP1,M+h/2*kM1)
    kM2 = dMdrUltraRel(r+h/2,P+h/2*kP1)
    kP3 = dPdrNewtonUltraRel(r+h/2,P+h/2*kP2,M+h/2*kM2)
    kM3 = dMdrUltraRel(r+h/2,P+h/2*kP2)
    kP4 = dPdrNewtonUltraRel(r+h,P+h*kP3,M+h*kM3)
    kM4 = dMdrUltraRel(r+h,P+h*kP3)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4

############################################################
#Function that calculates the mass and pressure at N points#
#given a central pressure Pc as function of the radius r.  #
#Reurns 5 vectors which contains the radius and masses     #
#obtained using both Newton and TOV-equation               #
############################################################
@jit
def resultsUltraRel(R,R0,Pc,h):
    N = int(R/h)
    PTOV = np.zeros(N)
    PTOV[0] = Pc
    PNew = np.zeros(N)
    PNew[0] = Pc
    MTOV = np.zeros(N)
    MNew = np.zeros(N)

    r = np.linspace(h,R,N)
    for i in range(1,N):
        (kPTOV1,kPTOV2,kPTOV3,kPTOV4,
        kMTOV1,kMTOV2,kMTOV3,kMTOV4) = kTOVUltraRel(r[i-1],PTOV[i-1],MTOV[i-1],h)

        PTOV[i] = PTOV[i-1] + h/6*(kPTOV1+2*kPTOV2+2*kPTOV3+kPTOV4)
        MTOV[i] = MTOV[i-1] + h/6*(kMTOV1+2*kMTOV2+2*kMTOV3+kMTOV4)

        (kPNew1,kPNew2,kPNew3,kPNew4,
        kMNew1,kMNew2,kMNew3,kMNew4) = kNewUltraRel(r[i-1],PNew[i-1],MNew[i-1],h)

        PNew[i] = PNew[i-1] + h/6*(kPNew1+2*kPNew2+2*kPNew3+kPNew4)
        MNew[i] = MNew[i-1] + h/6*(kMNew1+2*kMNew2+2*kMNew3+kMNew4)
    return r,PTOV,MTOV,PNew,MNew

##############################################
#Writes all ultra-relativistic data to a file#
##############################################
def writeResultsToFileUltraRel(R,Pc,h,filename):
    r,PTOV,MTOV,PNew,MNew = resultsUltraRel(R,R0,Pc,h)
    f = open(filename,'w')
    f.write(str(len(r))+'\n')
    for i in range(len(r)):
        a = str(r[i])
        b = str(PTOV[i])
        c = str(MTOV[i])
        d = str(PNew[i])
        e = str(MNew[i])
        f.write(a+" "+b+" "+c+" "+d+" "+e+"\n")
    f.close()
    return

#######################################################################
#Reads all ultra-relativistic data from file and returns it as vectors#
#######################################################################
def readResultsFromFileUltraRel(filename):
    f = open(filename,'r')
    N = int(f.readline())
    r = np.zeros(N)
    PTOV = np.zeros(N)
    MTOV = np.zeros(N)
    PNew = np.zeros(N)
    MNew = np.zeros(N)
    i = 0
    data = f.readlines()
    for line in data:
        numbers = line.split()
        r[i] = numbers[0]
        PTOV[i] = numbers[1]
        MTOV[i] = numbers[2]
        PNew[i] = numbers[3]
        MNew[i] = numbers[4]
        i = i+1
    f.close()
    return r,PTOV,MTOV,PNew,MNew

###########################################################################
#Calculates the analytical solution to the pressure for ultra-relativistic#
#ideal neutron gas                                                        #
###########################################################################
@jit
def analyticPressure(r):
    const = (14*R0*betaUltraRel/3)**(-1)
    return const*r**(-2)

####################################################################
#Calculates the analytical mass solution for the ultra-relativistic#
#ideal neutron gas                                                 #
####################################################################
@jit
def analyticMass(r):
    const = (14*R0*betaUltraRel/3)**(-1)
    return betaUltraRel*const*r

################################
################################
###                          ###
###   Arbitrary relativity   ###
###                          ###
################################
################################

###########################################################
#The derivative of the mass M with respect to the radius r#
#for arbitrary relativity                                 #
###########################################################
@jit
def dMdr(r,epsilon):
    return beta*epsilon*r**2
```

```python

###############################################################
#The derivative of the pressure P with respect to the radius r#
#for arbitrary relativity                                     #
###############################################################
@jit
def dPdrTOV(r,P,M,epsilon):
    return-R0*r**(-2)*(epsilon+P)*(M+beta*P*r**3)*(1-2*R0*M*r**(-1))**(-1)

#################################################
#The pressure as function of the fermi energy pF#
#################################################
@jit
def pressure(pF):
    return gamma*(24.*math.pi**2)**(-1)*(np.sqrt(pF**2+1)*(2*pF**3-3*pF)+3*np.log(pF+np.sqrt(pF**2+1)))

###################################################################
#Function that creates the coefficients in the Runge Kutta routine#
###################################################################
@jit
def k(r,P,M,epsilon,h):
    kP1 = dPdrTOV(r,P,M,epsilon)
    kM1 = dMdr(r,epsilon)
    kP2 = dPdrTOV(r+h/2,P+h/2*kP1,M+h/2*kM1,epsilon)
    kM2 = dMdr(r+h/2,epsilon)
    kP3 = dPdrTOV(r+h/2,P+h/2*kP2,M+h/2*kM2,epsilon)
    kM3 = dMdr(r+h/2,epsilon)
    kP4 = dPdrTOV(r+h/2,P+h*kP3,M+h*kM3,epsilon)
    kM4 = dMdr(r+h,epsilon)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4

#######################################################
#The energy density as a function of fermi momentum pF#
#######################################################
@jit
def energyDensity(pF):
    return gamma*(24.*math.pi**2)**(-1)*(np.sqrt(pF**2+1)*(6*pF**3+3*pF)-3*np.log(pF+np.sqrt(pF**2+1)))

def f(P):
    return lambda pF: pressure(pF)-P

#############################################################
#Function that returns 2 vectors containing the mass-radius #
#relation and a flag indicating if the maximum number of    #
#iterations is reached                                      #
#############################################################
def results(Pc,h,nMax,pFMax):
    P = Pc
    M = 0.
    r = 0.
    pF = pFMax
    flag=True
    for i in range(nMax):
        Memoryr = r
        MemoryM = M
        r = r+h
        if(np.sign(f(P)(0))!=np.sign(f(P)(pFMax))):
            pF = op.brentq(f(P),0,pFMax)
        else:
            pF=0
        epsilon = energyDensity(pF)
        kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4 = k(r,P,M,epsilon,h)
        P = P+h/6*(kP1+2*kP2+2*kP3+kP4)
        M = M+h/6*(kM1+2*kM2+2*kM3+kM4)
        if(np.float(np.real(P))<=0 or M==MemoryM):
            r = Memoryr
            flag = False
            break
    if(flag == True):
            print("Maximum number of iterations reached")
            print("for TOV-equation with Pc = %.8f"%Pc)
            print("with arbitrary relativity \n")
    return r, np.float(np.real(M)),flag


###############################################################
#Does the same as parametrisingNonRel for arbitrary relativity#
###############################################################
@jit
def paramterising(PcMin,PcMax,N,h,nMax,pFMax):
    Pc = PcMin
    const = (np.float(PcMax)/PcMin)**(np.float(1)/N)
    R = np.zeros(N)
    M = np.zeros(N)
    for i in range(N):
        R[i],M[i],flag = results(Pc,h,nMax,pFMax)
        if(flag==True):
            print(R[i])
            R = R[0:i]
            M = M[0:i]
            N = i
            break
        Pc = Pc*const
    return R,M,N

#####################
#Writes data to file#
#####################
def writeResultsToFile(PcMin,PcMax,N,h,nMax,pFMax,filename):
    R,M,N = paramterising(PcMin,PcMax,N,h,nMax,pFMax)
    f = open(filename,'w')
    f.write(str(N)+'\n')
    for i in range(N):
        a = str(R[i])
        b = str(M[i])
        f.write(a+" "+b+"\n")
    f.close()
    return

######################
#Reads data from file#
######################
def readResultsFromFile(filename):
    f = open(filename,'r')
    N = int(f.readline())
    R= np.zeros(N)
    M = np.zeros(N)
    i = 0
    data = f.readlines()
    for line in data:
        numbers = line.split()
        R[i] = numbers[0]
        M[i] = numbers[1]
        i = i+1
    f.close()
    return R,M

####################################################################
#Reads data from file and creates plots of the mass radius relation#
#for arbitrary relativity and the non-relativistic case            #
```

```
########################################################################
def createPlots(filename,filenameNonRel,filenameUltraRel,PcMax,h,nMax,pFMax):
    plotConstPressure()
    plotdPde(pFMax,N)
    plt.figure()
    plt.ylim(0,1.5)
    RTOVnonRel,MTOVnonRel,RNewNonRel,MNewNonRel = readResultsFromFileNonRel(filenameNonRel)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$R$[km]', fontsize = 14)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$M/M_\odot$', rotation='horizontal',fontsize = 14)

    plt.plot(RTOVnonRel,MTOVnonRel,'r',label='TOV non-relativistic')
    a,b,c,d,e,f=resultsNonRel(0.0113,h,nMax)
    plt.scatter(a,b,marker='x',color='green')
    plt.plot(RNewNonRel,MNewNonRel,'b',label = 'Newton non-relativistic')
    R,M = readResultsFromFile(filename)
    for i in range(len(M)):
        if(M[i]==0.0):
            M[i] = None
    plt.plot(R,M,'y',label='Arbitrary relativity')
    MMax = max(M)
    RMax = R[np.argmax(M)]
    print(RMax,np.argmax(M))
    print("The biggest possible mass is (arbitrary rel) %.3f sun masses with radi %.3fkm"%(MMax,RMax))
    plt.legend()
    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.0)
    ax.set_xlabel('$r$[km]', fontsize = 14)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(r)$', rotation='horizontal',fontsize = 14)
    RultraRel,PultraRelTOV,MultraRelTOV,PultraRelNew,MultraRelNew = readResultsFromFileUltraRel(filenameUltraRel)
    PultraRelAnal = analyticPressure(RultraRel)
    MultraRelAnal = analyticMass(RultraRel)
    plt.ylim(0,max(PultraRelTOV)+max(PultraRelTOV)*0.1)
    plt.xlim(0,max(RultraRel))
    plt.plot(RultraRel,PultraRelTOV,'r',label = 'TOV')
    plt.plot(RultraRel,PultraRelNew,'b',label = 'Newton')
    plt.plot(RultraRel,PultraRelAnal,'g',label = 'Analytic')
    plt.legend()

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.08, -0.0)
    ax.set_xlabel('$r$ [km]', fontsize = 14)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    plt.ylim(0,8)
    plt.xlim(0,15)
    ax.set_ylabel('$M/M_\odot$', rotation='horizontal',fontsize = 14)
    plt.plot(RultraRel,MultraRelTOV,'r',label = 'TOV')
    plt.plot(RultraRel,MultraRelNew,'b',label = 'Newton')
    plt.plot(RultraRel,MultraRelAnal,'g',label='Analytic')
    plt.legend()
    return

@jit
def dPde(pFMax,N):
    pF = np.linspace(0,pFMax,N+2)
    dPde = np.ones(N+2)
    for i in range(1,N+1):
        dP = pressure(pF[i+1])-pressure(pF[i-1])
        de = energyDensity(pF[i+1])-energyDensity(pF[i-1])
        dPde[i] = dP/de
    return pF[1:N+1],dPde[1:N+1]


@jit
def plotdPde(pFMax,N):
    pF,Pde = dPde(pFMax,N)
    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$\\bar{p}_F$',fontsize = 14)
    ay = plt.gca()
    ay.yaxis.set_label_coords(-0.05, 1.0)
    ay.set_ylabel('d$\\bar{P}/$d$\\bar{\epsilon}$', rotation='horizontal', fontsize = 14)
    plt.plot(pF,Pde)

################################
################################
###                          ###
###   Program starts here    ###
###                          ###
################################
################################



alpha = 1.                  #alpha, beta and gamma is given
beta = 1.1426               #by calculations in thesis
betaNonRel = 0.7779
betaUltraRel = 1.
gamma = 1.

Pc = 0.1                    #Central pressure used in ultra relativistic star
R = 30                      #Radius we integrate up to for ultra relativistic star
R0 = 1.477                  #MG/c^2
K = 0.67704                 #Reffered to as K_NR "bar" in thesis
h = 0.0001                  #Step length in the Runge Cutta solver
nMax = int(100/h)           #Maximum number of iterations in solver
PcMin = 1.*10**(-6)         #Smallest central pressure we solve for
PcMax = 1.*10**(4)          #Largest central pressure we solve for
N = 300                     #Number of data points in plot
pFMax = 10.                 #Maximum fermimomentum used in root finding function

filenameNonRel = "massRadiRelationNonRel.txt"
filenameUltraRel = "massRadiRelationUltraRel.txt"
filename = "massRadiRelation.txt"

writeResultsToFileNonRel(PcMin,PcMax,N,h,nMax,filenameNonRel)
writeResultsToFileUltraRel(R,Pc,h,filenameUltraRel)
writeResultsToFile(PcMin,PcMax,N,h,nMax,pFMax,filename)
createPlots(filename,filenameNonRel,filenameUltraRel,PcMax,h,nMax,pFMax)
plt.show()

print("\nTime spent:")
print(time.clock()-t0)
```

G.2 Chapter 5

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import time
from numba import jit
from scipy import optimize as op
import math
from scipy import interpolate

##########################################################
##########################################################
###                                                    ###
###   Just press play to create all plots in thesis!   ###
###                                                    ###
##########################################################
##########################################################

#Variable used to time the program
t0 = time.clock()

###########################
###                     ###
###  Defining functons  ###
###                     ###
###########################

#################################################
#Takes in a number in fm and converts it to MeV #
#Takes in the exponent. For instance, to convert#
#from fm^-3 to MeV^3, set exponent to -3        #
#################################################
def fmToMeV(number,exponent):
    if(exponent>0):
        return (number**(1./exponent)/197.33)**exponent
    else:
        return (number**(-1./exponent)*197.33)**(-exponent)

#################################################
#Takes in a number in MeV and converts it to fm #
#Takes in the exponent. For instance, to convert#
#from MeV^3 to fm^-3, set exponent to 3         #
#################################################
def MeVtoFm(number,exponent):
    if(exponent>0):
        return (number**(1./exponent)/197.33)**exponent
    else:
        return (number**(-1./exponent)*197.33)**(-exponent)

################################################################
#Function that takes in a density and returns the corresponding#
#Fermi momentum given a degenracy factor f                     #
################################################################
@jit
def kFfromDensity(density,f):
    return (6*math.pi**2*density/f)**(1./3)

#########################################################
#Function that takes in a Fermi momentum and returns the#
#corresponding density given a degeneracy factor f      #
#########################################################
@jit
def density(kF,f):
    return f*kF**3./(6*math.pi**2)

#######################################################
#Returns te scalar density for a given Fermi momentum,#
#effective mass and degeneracy factor f               #
#######################################################
@jit
def scalarDensity(kF,gSigma,mStar,f):
    x = kF/mStar
    return f/4.*mStar**3*math.pi**(-2)*(np.sqrt(x**2+1.)*x-np.arcsinh(x))


#######################################################
#Returns the effective mass for a given Fermi momentum#
#and degeneracy factor f                              #
#######################################################
def effectiveMass(kF,gSigma,f):
    a = lambda mStar: 1.-gSigma**2*mSigma**(-2)*scalarDensity(kF,gSigma,mStar,f)-mStar
    if(np.sign(a(10**(-16)))==np.sign(a(1.))):
        return 10**(-16)
    return op.brentq(a,10**(-16),1.)

######################################################################
#Takes in a maximum value for the Fermi momenta kFmax, and the length#
#N1 requested. Returns one vector containing the Fermi momenta evenly#
#spaced between 0 and kFmax and two vectors containing the           #
#corresponding effective masses for degeneracy factor f=2 and f=4    #
######################################################################
@jit
def massVec(kFMax,gSigma,N1):
    kFvec = np.zeros(N1+1)
    kFvec = np.linspace(10**(-16),kFMax,N1)
    mStarVec2 = np.zeros(N1)
    mStarVec4 = np.zeros(N1)
    for i in range(N1):
        mStarVec2[i] = effectiveMass(kFvec[i],gSigma,2.)
        mStarVec4[i] = effectiveMass(kFvec[i],gSigma,4.)
    return kFvec,[mStarVec2,mStarVec4]

################################################################
#Takes in a Fermi momenta, effective mass and degeneracy factor#
#and returns the pressure                                      #
################################################################
@jit
def pressure(kF,gSigma,gOmega,mStar,f):
    x = kF/mStar
    a = 0.5*gSigma**2*mSigma**(-2)*scalarDensity(kF,gSigma,mStar,f)**2
    b = 0.5*gOmega**2*mOmega**(-2)*density(kF,f)**2
    c = f*(48*math.pi**2)**(-1)*mStar**4
    d = np.sqrt(x**2+1.)*(2*x**3-3*x)+3*np.arcsinh(x)
    return -a+b+c*d

#################################################################
#Takes in one vector countaining all the Fermi momenta and       #
#two containing the effective masses for each degenarcy factor, #
#as well as the length of the vectors. Returns the corresponding#
#pressures for each degenracy factor                            #
#################################################################
@jit
def pressureVec(kFvec,gSigma,gOmega,mStarVec,N1):
    Pvec2 = np.zeros(N1)
    Pvec4 = np.zeros(N1)
    mStarVec2 = mStarVec[0]
    mStarVec4 = mStarVec[1]
    for i in range(N1):
        Pvec2[i] = pressure(kFvec[i],gSigma,gOmega,mStarVec2[i],2.)
        Pvec4[i] = pressure(kFvec[i],gSigma,gOmega,mStarVec4[i],4.)
    return [Pvec2,Pvec4]

#############################################################
#Takes in a Fermi momenta and effective mass and returns the#
#energy density for a given degeneracy factor               #
#############################################################
@jit
def energyDensity(kF,gSigma,gOmega,mStar,f):
    x = kF/mStar
    a = 0.5*gSigma**2*mSigma**(-2)*scalarDensity(kF,gSigma,mStar,f)**2
    b = 0.5*gOmega**2*mOmega**(-2)*density(kF,f)**2
    c = f*(16*math.pi**2)**(-1)*mStar**4
    d = np.sqrt(x**2+1.)*(2*x**3+x)-np.arcsinh(x)
    return a+b+c*d

####################################################################
#Same as the function pressureVec, except that the vectors returned#
#are energy densities                                              #
####################################################################
@jit
def energyDensityVec(kFvec,gSigma,gOmega,mStarVec,N1):
    epsilonVec2 = np.zeros(N1)
    epsilonVec4 = np.zeros(N1)
    mStarVec2 = mStarVec[0]
    mStarVec4 = mStarVec[1]
    for i in range(N1):
        epsilonVec2[i] = energyDensity(kFvec[i],gSigma,gOmega,mStarVec2[i],2.)
        epsilonVec4[i] = energyDensity(kFvec[i],gSigma,gOmega,mStarVec4[i],4.)
    return [epsilonVec2,epsilonVec4]

#######################################################
#Same as the function pressure, except that it returns#
#the pressure for a free Fermi gas                    #
#######################################################
@jit
def fermiGasPressure(kF,mStar,f):
    a = f/(48*math.pi**2)
    b = np.sqrt(kF**2+mStar**2)
    c = 2*kF**3-3*mStar**2*kF
    return a*(b*c+3*mStar**4*np.log((kF+b)/mStar))

##########################################################
#Same as the function pressureVec, except that it returns#
#the pressure vecotrs for a free Fermi gas               #
##########################################################
@jit
def fermiGasPressureVec(kFvec,N1):
    fgPvec2 = fermiGasPressure(kFvec,np.ones(N1),2)
    fgPvec4 = fermiGasPressure(kFvec,np.ones(N1),4)
    return fgPvec2,fgPvec4

############################################################
#Same as the function energyDensity, except that it returns#
#the energy density for a free Fermi gas                   #
############################################################
def fermiGasEnergyDensity(kF,mStar,f):
    a = f/(48*math.pi**2)
    b = np.sqrt(kF**2+mStar**2)
    c = 6*kF**3+3*mStar**2*kF
    return a*(b*c-3*mStar**4*np.log((kF+b)/mStar))

###############################################################
#Same as the function energyDensityVec, except that it returns#
#the energy density vectors for a free Fermi gas              #
###############################################################
@jit
def fermiGasEnergyDensityVec(kFvec,N1):
    fgEpsilonVec2 = fermiGasEnergyDensity(kFvec,np.ones(N1),2)
    fgEpsilonVec4 = fermiGasEnergyDensity(kFvec,np.ones(N1),4)
    return fgEpsilonVec2,fgEpsilonVec4

#################################################################
#Function that takes in pressure vectors for the sigma-omega EoS#
#and the free Fermi gas EoS, and combines them at the point     #
```

```
#where they conside                                             #
#################################################################
@jit
def combineFermiSigmaOmega(Pvec,fgPvec,N1):
    print(N1)
    index = 0
    for i in range(N1-1):
        if(Pvec[i]>10**(-5)):
            if(np.sign(Pvec[i]-fgPvec[i])!=np.sign(Pvec[i+1]-fgPvec[i+1])):
                index = i+1
                break
    PvecCorr = np.zeros(N1)
    PvecCorr[0:index] = fgPvec[0:index]
    PvecCorr[index:N1] = Pvec[index:N1]
    return PvecCorr


##############################################################################
#Takes in a maximal Fermi momentum and the length of the vector requested    #
#Returns all vectors needed to calculate the mass-radius relations (and more)#
##############################################################################
@jit
def createVectors(kFMax,N1,gSigma,gOmega):
    kFvec,mStarVec = massVec(kFMax,gSigma,N1)
    Pvec = pressureVec(kFvec,gSigma,gOmega,mStarVec,N1)
    epsilonVec = energyDensityVec(kFvec,gSigma,gOmega,mStarVec,N1)
    fgPvec = fermiGasPressureVec(kFvec,N1)
    PvecCorr2 = combineFermiSigmaOmega(Pvec[0],fgPvec[0],N1)
    PvecCorr4 = combineFermiSigmaOmega(Pvec[1],fgPvec[1],N1)
    PvecCorr = [PvecCorr2,PvecCorr4]
    fgEpsilonVec = fermiGasEnergyDensityVec(kFvec,N1)

    gSigma = 10.7522
    gOmega = 15.8533
    kFvec,mStarVec = massVec(kFMax,gSigma,N1)
    PvecCoupl = pressureVec(kFvec,gSigma,gOmega,mStarVec,N1)
    epsilonVecCoupl = energyDensityVec(kFvec,gSigma,gOmega,mStarVec,N1)

    return kFvec,Pvec,epsilonVec,fgEpsilonVec,fgPvec,PvecCorr,epsilonVecCoupl,PvecCoupl

################################################################
#Takes in two vectors containing the pressure and enery density#
#values and creates plots of the EoS                          #
################################################################
def plotEoS(Pvec,epsilonVec):
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]
    epsilonVec2 = epsilonVec[0]
    epsilonVec4 = epsilonVec[1]

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$\\bar{\epsilon}$', fontsize = 15)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 15)
    ax.text(-0.00004,-0.4*10**(-7),'E',fontsize = 14)
    ax.text(0.00006,0.7*10**(-7),'D',fontsize = 14)
    ax.text(1.7*10**(-4),-0.4*10**(-7),'C',fontsize = 14)
    ax.text(0.00058,-1.45*10**(-6),'B',fontsize = 14)
    ax.text(0.00082,-0.4*10**(-7),'A',fontsize=14)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.xlim(-0.00005,0.001)
    plt.ylim(-0.0000015,0.0000007)
    plt.plot(epsilonVec2,Pvec2,'r')

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$\\bar{\epsilon}$', fontsize = 14)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 15)
    ax.text(-6*10**(-5),-5*10**(-7),'E',fontsize = 14)
    ax.text(0.00002,5*10**(-7),'D',fontsize = 14)
    ax.text(1.5*10**(-4),-5*10**(-7),'C',fontsize = 14)
    ax.text(0.001,-1.45*10**(-5),'B',fontsize = 14)
    ax.text(0.00143,-5*10**(-7),'A',fontsize=14)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.xlim(-7*10**(-5),0.0016)
    plt.ylim(-0.000016,0.00004)
    plt.plot(epsilonVec4,Pvec4,'b')

############################################################
#Takes in the "normal" sigma omega EoS and the one where we#
#have altered the couplings. Then plots them together      #
############################################################
def plotEoSDifferentCouplings(PvecCoupl,epsilonVecCoupl,Pvec,epsilonVec):
    PvecCoupl2 = PvecCoupl[0]
    PvecCoupl4 = PvecCoupl[1]
    epsilonVecCoupl2 = epsilonVecCoupl[0]
    epsilonVecCoupl4 = epsilonVecCoupl[1]
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]
    epsilonVec2 = epsilonVec[0]
    epsilonVec4 = epsilonVec[1]

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$\\bar{\epsilon}$', fontsize = 15)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 15)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.xlim(0,1.2*10**(-3))
    plt.ylim(-0.25*10**(-5),1.5*10**(-5))
    plt.plot(epsilonVec2,Pvec2,'r',label='original couplings')
    plt.plot(epsilonVecCoupl2,PvecCoupl2,'y',label = 'altered couplings')
    plt.legend()

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.set_xlabel('$\\bar{\epsilon}$', fontsize = 14)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 15)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    plt.xlim(0,1.7*10**(-3))
    plt.ylim(-1.3*10**(-5),8*10**(-5))
    plt.plot(epsilonVec4,Pvec4,'b',label='original couplings')
    plt.plot(epsilonVecCoupl4,PvecCoupl4,'g',label = 'altered couplings')
    plt.legend()

########################################################
#Plots pressure as a funciton of inverse energy density#
########################################################
def plotPV(kFvec,Pvec,epsilonVec):
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]
    epsilonVec2 = epsilonVec[0]
    epsilonVec4 = epsilonVec[1]

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('log($1/\\bar{\epsilon}$)', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$\\bar{P}(1/\\bar{\epsilon})$', rotation='horizontal',fontsize = 14)
    plt.xlim(2,10)
    plt.ylim(-0.000002,0.000002)
    ax.text(2400,-0.4*10**(-7),'C',fontsize = 14)
    ax.text(1300,-3.5*10**(-6),'B',fontsize = 14)
    ax.text(500,-0.75*10**(-6),'A',fontsize=14)
    plt.plot(np.log10(1./epsilonVec2),Pvec2,'r')
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    ax.text(2.900,-1.*10**(-7),'A',fontsize = 14)
    ax.text(3.1700,-15*10**(-7),'B',fontsize = 14)
    ax.text(3.55,-1.*10**(-7),'C',fontsize=14)
    ax.text(4,6*10**(-8),'D',fontsize=14)
    ax.text(9.8,2*10**(-8),'E',fontsize=14)

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('log($1/\\bar{\epsilon}$)', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 14)
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    plt.ticklabel_format(style='sci' ,axis='y', scilimits=(0,0))
    ax.text(2.600,-4*10**(-7),'A',fontsize = 14)
    ax.text(2.9500,-12.2*10**(-6),'B',fontsize = 14)
    ax.text(3.45,-4*10**(-7),'C',fontsize=14)
    ax.text(4.7,2*10**(-7),'D',fontsize=14)
    ax.text(9.8,2*10**(-7),'E',fontsize=14)
    plt.xlim(2,10)
    plt.ylim(-1.25*10**(-5),0.000005)
    plt.plot(np.log10(1./epsilonVec4),Pvec4,'b')

#######################################################
#Plots the combined free Fermi gas and sigma omega EoS#
#######################################################
def plotCorrectedPressure(kFvec,PvecCorr,Pvec):
    PvecCorr2 = PvecCorr[0]
    PvecCorr4 = PvecCorr[1]
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('$\\bar{k}_F$', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{k}_F)$', rotation='horizontal',fontsize = 14)
    plt.plot(kFvec,Pvec2,'y',label = '$\sigma-\omega$')
    plt.plot(kFvec,PvecCorr2,'r',label = 'corrected')
    plt.legend()
    plt.xlim(0,0.4)
    plt.ylim(-0.1*10**(-4),1.5*10**(-4))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('$\\bar{k}_F$', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{k}_F)$', rotation='horizontal',fontsize = 14)
    plt.plot(kFvec,Pvec4,'g',label = '$\sigma-\omega$')
    plt.plot(kFvec,PvecCorr4,'b',label = 'corrected')
    plt.legend()
    plt.xlim(0,0.4)
    plt.ylim(-0.2*10**(-4),1.5*10**(-4))
    plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))

##########################################################
#Takes in a pressure P and a vector with pressure values.#
#Returns the index of the vector closest to P            #
##########################################################
@jit
def findIndex(P,Pvec,N1):
    P2 = P[0]
    P4 = P[1]
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]
    index2 = np.zeros(3,int)
    index4 = np.zeros(3,int)
    a2 = 0
    a4 = 0
    tolerance2 = 100
    tolerance4 = 100
    for i in range(N1-1):

        if(abs(Pvec2[i]-Pvec2[i+1])<tolerance2):
            tolerance2 = abs(Pvec2[i]-Pvec2[i+1])

        if(abs(Pvec4[i]-Pvec4[i+1])<tolerance4):
            tolerance4 = abs(Pvec4[i]-Pvec4[i+1])

    for i in range(N1-1):
        if((np.sign(P2-Pvec2[i])!=np.sign(P2-Pvec2[i+1])) or abs(Pvec2[i]-P2)<tolerance2):
            index2[a2] = i+1
            a2+=1
        if((np.sign(P4-Pvec4[i])!=np.sign(P4-Pvec4[i+1])) or abs(Pvec4[i]-P4)<tolerance4):
            index4[a4] = i+1
            a4+=1

    flag2 = True
    flag4 = True

    if(a2<3):
        flag2 = False

    if(a4<3):
        flag4 = False
    return [index2,index4],[flag2,flag4]

###################################################################
#Returns the area under the curve of the pressure vs invese energy#
#density between index start and stop for the energy density      #
### ###############################################################
def area(epsilonVec,Pvec,start,stop):
    return integrate.simps(1./epsilonVec[start:stop],Pvec[start:stop])


################################
###                          ###
###   Mass-radii relations   ###
###                          ###
################################
```

```

###########################################################
#The derivative of the mass M with respect to the radius r#
#for arbitrary relativity                                 #
###########################################################
@jit
def dMdr(r,epsilon):
    return beta*epsilon*r**2

###############################################################
#The derivative of the pressure P with respect to the radius r#
#for arbitrary relativity                                     #
###############################################################
@jit
def dPdrTOV(r,P,M,epsilon):
    return-R0*r**(-2.)*(epsilon+P)*(M+beta*P*r**3)*(1.-2.*R0*M*r**(-1))**(-1)


###################################################################
#Function that creates the coefficients in the Runge Kutta routine#
###################################################################
@jit
def k(r,P,M,epsilon,h):
    kP1 = dPdrTOV(r,P,M,epsilon)
    kM1 = dMdr(r,epsilon)
    kP2 = dPdrTOV(r+h/2,P+h/2*kP1,M+h/2*kM1,epsilon)
    kM2 = dMdr(r+h/2,epsilon)
    kP3 = dPdrTOV(r+h/2,P+h/2*kP2,M+h/2*kM2,epsilon)
    kM3 = dMdr(r+h/2,epsilon)
    kP4 = dPdrTOV(r+h/2,P+h*kP3,M+h*kM3,epsilon)
    kM4 = dMdr(r+h,epsilon)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4



################################################################
#Function that returns the mass and radius of a star with given#
#central pressure Pc for both degeneracy factors. It also      #
#returns two flags to indicate if the routine converged        #
################################################################
@jit
def results(Pc,h,nMax,EoS2,EoS4):
    P2 = Pc
    M2 = 0.
    r2 = 0.
    flag2=True
    P4 = Pc
    M4 = 0.
    r4 = 0.
    flag4=True
    tolerance = 10**(-7)
    for i in range(nMax):
        if(flag2==True):
            Memoryr2 = r2
            MemoryM2 = M2
            r2 = r2+h
            epsilon2 = EoS2(P2)
            kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4 = k(r2,P2,M2,epsilon2,h)
            P2 = P2+h/6*(kP1+2*kP2+2*kP3+kP4)
            M2 = M2+h/6*(kM1+2*kM2+2*kM3+kM4)
            if(np.float(np.real(P2))<=0. or (((M2-MemoryM2)/(M2+MemoryM2))**2<tolerance**2 and M2!=0 and MemoryM2!=0)):
                r2 = Memoryr2
                flag2 = False

        if(flag4==True):
            Memoryr4 = r4
            MemoryM4 = M4
            r4 = r4+h
            epsilon4 = EoS4(P4)
            kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4 = k(r4,P4,M4,epsilon4,h)
            P4 = P4+h/6*(kP1+2*kP2+2*kP3+kP4)
            M4 = M4+h/6*(kM1+2*kM2+2*kM3+kM4)
            if(np.float(np.real(P4))<=0. or (((M4-MemoryM4)/(M4+MemoryM4))**2<tolerance**2 and M4!=0 and MemoryM4!=0)):
                r4 = Memoryr4
                flag4 = False

        if((flag2 == False) and (flag4 == False)):
            break


    if(flag2 == True):
            print("Maximum number of iterations reached")
            print("for TOV-equation with Pc = %.8f"%Pc)
            print("with arbitrary relativity and degenerassy 2\n")

    if(flag4 == True):
            print("Maximum number of iterations reached")
            print("for TOV-equation with Pc = %.8f"%Pc)
            print("with arbitrary relativity and degenerassy 4\n")
    return r2, np.float(np.real(M2)),r4,np.float(np.real(M4)),flag2,flag4


####################################################################
#Parametrises the mass-radus relation from central pressures between#
#PcMin and PcMax                                                    #
####################################################################
@jit
def paramterising(parameters,epsilonVec,Pvec):
    PcMin = parameters[0]
    PcMax = parameters[1]
    N2 = parameters[2]
    h = parameters[3]
    nMax = parameters[4]

    epsilonVec2 = epsilonVec[0]
    epsilonVec4 = epsilonVec[1]
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]

    Pc = PcMin
    N1 = len(Pvec2)
    if(PcMax>Pvec2[N1-1]):
        print("PcMax changed from ",PcMax," to ", Pvec2[N1-1])
        PcMax = Pvec2[N1-1]
    if(PcMax>Pvec4[N1-1]):
        print("PcMax changed from ",PcMax," to ", Pvec4[N1-1])
        PcMax = Pvec4[N1-1]
    const = (np.float(PcMax)/PcMin)**(1./N2)
    R2 = np.zeros(N2,np.double)
    M2 = np.zeros(N2,np.double)
    R4 = np.zeros(N2,np.double)
    M4 = np.zeros(N2,np.double)

    EoS2 = interpolate.interp1d(Pvec2,epsilonVec2)
    EoS4 = interpolate.interp1d(Pvec4,epsilonVec4)

    for i in range(N2):
        print(i)
        R2[i],M2[i],R4[i],M4[i],flag2,flag4 = results(Pc,h,nMax,EoS2,EoS4)
        if((flag2==True) or (flag4 == True)):
```

```python
            print(R2[i],R4[i])
            R2 = R2[0:i]
            M2 = M2[0:i]
            R4 = R4[0:i]
            M4 = M4[0:i]
            N = i
            break
        Pc = Pc*const
    return R2,M2,R4,M4,N2

######################################
#Writes mass-radius relations to file#
######################################
def writeResultsToFile(parameters,epsilonVec,Pvec,filename):

    R2,M2,R4,M4,N2 = paramterising(parameters,epsilonVec,Pvec)

    f2 = open(filename+'2','w')
    f2.write(str(N2)+'\n')
    for i in range(N2):
        a = str(R2[i])
        b = str(M2[i])
        f2.write(a+" "+b+"\n")
    f2.close()

    f4 = open(filename+'4','w')
    f4.write(str(N2)+'\n')
    for i in range(N2):
        a = str(R4[i])
        b = str(M4[i])
        f4.write(a+" "+b+"\n")
    f4.close()

    return

##########################################
#Writes all mass-radius relations to file#
##########################################
def writeAllResultsToFile(parameters,epsilonVec,Pvec,PvecCorr,PvecCoupl,filenameVec):
    writeResultsToFile(parameters,epsilonVec,Pvec,filenameVec[0])
    writeResultsToFile(parameters,epsilonVec,PvecCorr,filenameVec[1])
    writeResultsToFile(parameters,epsilonVec,PvecCoupl,filenameVec[2])




######################################
#Reads mass-radius relation from file#
######################################
def readResultsFromFile(filename):
    f2 = open(filename+'2','r')
    N2 = int(f2.readline())
    R2 = np.zeros(N,np.double)
    M2 = np.zeros(N,np.double)
    i = 0
    data = f2.readlines()
    for line in data:
        numbers = line.split()
        R2[i] = numbers[0]
        M2[i] = numbers[1]
        i = i+1
    f2.close()

    f4 = open(filename+'4','r')
    N2 = int(f4.readline())
    R4 = np.zeros(N,np.double)
    M4 = np.zeros(N,np.double)
    i = 0
    data = f4.readlines()
    for line in data:
        numbers = line.split()
        R4[i] = numbers[0]
        M4[i] = numbers[1]
        i = i+1
    f4.close()

    return [R2,R4],[M2,M4],N2

###################################################################
#Plots the mass-radius relation for a given mass and radius vector.#
#Colour is a vector with length 2 that decides the colour of the   #
#curves for f=2 and f=4.                                            #
###################################################################
def plotMassRadiusRelations(R,M,N,colour,Label):
    R2 = R[0]
    R4 = R[1]
    M2 = M[0]
    M4 = M[1]
    R2stable = R2[0:list(M2).index(max(M2))+1]
    R2unstable = R2[list(M2).index(max(M2)):N]
    M2stable = M2[0:list(M2).index(max(M2))+1]
    M2unstable = M2[list(M2).index(max(M2)):N]
    R4stable = R4[0:list(M4).index(max(M4))+1]
    R4unstable = R4[list(M4).index(max(M4)):N]
    M4stable = M4[0:list(M4).index(max(M4))+1]
    M4unstable = M4[list(M4).index(max(M4)):N]

    print('Maximum mass for '+colour[0]+' is',max(M2), 'solar masses with radius', R2[list(M2).index(max(M2))],'km')
    print('Maximum mass for '+colour[1]+' is',max(M4), 'solar masses with radius', R2[list(M4).index(max(M4))],'km')

    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('$R$[km]', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$M/M_\odot$', rotation='horizontal',fontsize = 14)
    plt.plot(R2stable,M2stable,colour[0],label=Label[0])
    plt.plot(R2unstable,M2unstable,colour[0]+'--')
    plt.plot(R4stable,M4stable,colour[1],label=Label[1])
    plt.plot(R4unstable,M4unstable,colour[1]+'--')
    plt.xlim(3,29)
    plt.legend()

#####################################################
#Function that plots all relevant mass-radius relations#
#####################################################
def plotAllMassRadiusRelationsFromFile(filenameVec,colourVec):
    R,M,N = readResultsFromFile(filenameVec[0])
    Rcorr,Mcorr,N = readResultsFromFile(filenameVec[1])
    Rcoupl,Mcoupl,N = readResultsFromFile(filenameVec[2])

    Label = ['f=2','f=4']
    plotMassRadiusRelations(R,M,N,colourVec[0],Label)
    plotMassRadiusRelations(Rcorr,Mcorr,N,colourVec[1],Label)
    plotMassRadiusRelations(Rcoupl,Mcoupl,N,colourVec[2],Label)

    R2 = R[0]
    R4 = R[1]
    M2 = M[0]
    M4 = M[1]
    Rcorr2 = Rcorr[0]
    Rcorr4 = Rcorr[1]
    Mcorr2 = Mcorr[0]
    Mcorr4 = Mcorr[1]

    Rcoupl2 = Rcoupl[0]
    Rcoupl4 = Rcoupl[1]
    Mcoupl2 = Mcoupl[0]
    Mcoupl4 = Mcoupl[1]

    R2stable = R2[0:list(M2).index(max(M2))+1]
    R2unstable = R2[list(M2).index(max(M2)):N]
    M2stable = M2[0:list(M2).index(max(M2))+1]
    M2unstable = M2[list(M2).index(max(M2)):N]

    Rcorr2stable = Rcorr2[0:list(Mcorr2).index(max(Mcorr2))+1]
    Rcorr2unstable = Rcorr2[list(Mcorr2).index(max(Mcorr2)):N]
    Mcorr2stable = Mcorr2[0:list(Mcorr2).index(max(Mcorr2))+1]
    Mcorr2unstable = Mcorr2[list(Mcorr2).index(max(Mcorr2)):N]

    Rcoupl2stable = Rcoupl2[0:list(Mcoupl2).index(max(Mcoupl2))+1]
    Rcoupl2unstable = Rcoupl2[list(Mcoupl2).index(max(Mcoupl2)):N]
    Mcoupl2stable = Mcoupl2[0:list(Mcoupl2).index(max(Mcoupl2))+1]
    Mcoupl2unstable = Mcoupl2[list(Mcoupl2).index(max(Mcoupl2)):N]

    R4stable = R4[0:list(M4).index(max(M4))+1]
    R4unstable = R4[list(M4).index(max(M4)):N]
    M4stable = M4[0:list(M4).index(max(M4))+1]
    M4unstable = M4[list(M4).index(max(M4)):N]

    Rcorr4stable = Rcorr4[0:list(Mcorr4).index(max(Mcorr4))+1]
    Rcorr4unstable = Rcorr4[list(Mcorr4).index(max(Mcorr4)):N]
    Mcorr4stable = Mcorr4[0:list(Mcorr4).index(max(Mcorr4))+1]
    Mcorr4unstable = Mcorr4[list(Mcorr4).index(max(Mcorr4)):N]

    Rcoupl4stable = Rcoupl4[0:list(Mcoupl4).index(max(Mcoupl4))+1]
    Rcoupl4unstable = Rcoupl4[list(Mcoupl4).index(max(Mcoupl4)):N]
    Mcoupl4stable = Mcoupl4[0:list(Mcoupl4).index(max(Mcoupl4))+1]
    Mcoupl4unstable = Mcoupl4[list(Mcoupl4).index(max(Mcoupl4)):N]

    colour = colourVec[0]
    colourCorr = colourVec[1]
    colourCoupl = colourVec[2]

    plt.figure()
    plt.plot(R2stable,M2stable,colour[0],label = '$\sigma-\omega$, bounded, f=2')
    plt.plot(R2unstable,M2unstable,colour[0]+'--')
    plt.plot(R4stable,M4stable,colour[1],label = '$\sigma-\omega$, bounded, f=4')
    plt.plot(R4unstable,M4unstable,colour[1]+'--')
    plt.plot(Rcoupl2stable,Mcoupl2stable,colourCoupl[0],label = 'New Couplings $\sigma-\omega$, f=2')
    plt.plot(Rcoupl2unstable,Mcoupl2unstable,colourCoupl[0]+'--')
    plt.plot(Rcoupl4stable,Mcoupl4stable,colourCoupl[1],label = 'New Couplings $\sigma-\omega$, f=4')
    plt.plot(Rcoupl4unstable,Mcoupl4unstable,colourCoupl[1]+'--')
    plt.plot(Rcorr2stable,Mcorr2stable,colourCorr[0],label = 'Fermi-$\sigma-\omega$, f=2')
    plt.plot(Rcorr2unstable,Mcorr2unstable,colourCorr[0]+'--')
    plt.plot(Rcorr4stable,Mcorr4stable,colourCorr[1],label = 'Fermi-$\sigma-\omega$, f=4')
    plt.plot(Rcorr4unstable,Mcorr4unstable,colourCorr[1]+'--')
    plt.xlim(3,29)


    plt.legend()

    ax = plt.gca()
    ax.xaxis.set_label_coords(1.07, -0.02)
    ax.set_xlabel('$R$[km]', fontsize = 14)
    ax.yaxis.set_label_coords(-0.06, 1.03)
    ax.set_ylabel('$M/M_\odot$', rotation='horizontal',fontsize = 14)


##########################################################################
#Calculates the coupling constants within an interval divided in N pieces#
##########################################################################
@jit
def couplingConstants(gSigmaRange,gOmegaRange,N):
    satDens = fmToMeV(0.153,-3)/939**3
    bindingEnergy = -16.3/939
    gSigma = gSigmaRange[0]
    gOmega = gOmegaRange[0]
    f=4
    hSigma = (gSigmaRange[1]-gSigmaRange[0])/(N-1)
    hOmega = (gOmegaRange[1]-gSigmaRange[0])/(N-1)
    kFsat = kFfromDensity(satDens,f)
    bestFit = 100.
    bestGsigma = 0.
    bestGomega = 0.
    h = 10**(-10)
    rhoPluss = density(kFsat+h,f)
    rhoMinus = density(kFsat-h,f)
    drho = rhoPluss-rhoMinus

    for i in range(N):
        for j in range(N):
            mStar = effectiveMass(kFsat,gSigma,f)
            epsilon = energyDensity(kFsat,gSigma,gOmega,mStar,f)
            mStarPluss = effectiveMass(kFsat+h,gSigma,f)
            epsilonPluss = energyDensity(kFsat+h,gSigma,gOmega,mStarPluss,f)
            mStarMinus = effectiveMass(kFsat-h,gSigma,f)
            epsilonMinus = energyDensity(kFsat-h,gSigma,gOmega,mStarMinus,f)

            testBindingEnergy = epsilon/satDens-1.

            derivativeEpsilon = (epsilonPluss-epsilonMinus)/(drho)

            k = 1.-testBindingEnergy/bindingEnergy
            l = 1.-derivativeEpsilon/epsilon*satDens
            test = np.sqrt(k**2+l**2)

            if(test<bestFit):
                bestGsigma = gSigma
                bestGomega = gOmega
                bestFit = test

            gOmega += hOmega

        gOmega = gOmegaRange[0]
        gSigma += hSigma

    return bestGsigma,bestGomega

##############################################################
#N decides how many pieces we divide the interval [0,200] in.#
#Returns the obtained coupling constants                     #
##############################################################
@jit
def couplings(N):
    gSigmaRange = [0,200]
    gOmegaRange = [0,200]
    gSigma,gOmega = couplingConstants(gSigmaRange,gOmegaRange,N)
    print(gSigma,gOmega)
    gSigmaRange[0] = gSigma-10.
    gSigmaRange[1] = gSigma+10.
    gOmegaRange[0] = gOmega-10.
    gOmegaRange[1] = gOmega+10.
    gSigma,gOmega = couplingConstants(gSigmaRange,gOmegaRange,N)
    print(gSigma,gOmega)
    gSigmaRange[0] = gSigma-1.
    gSigmaRange[1] = gSigma+1.
    gOmegaRange[0] = gOmega-1.
    gOmegaRange[1] = gOmega+1.
    gSigma,gOmega = couplingConstants(gSigmaRange,gOmegaRange,N)
    print(gSigma,gOmega)
    gSigmaRange[0] = gSigma-0.1
    gSigmaRange[1] = gSigma+0.1
    gOmegaRange[0] = gOmega-0.1
    gOmegaRange[1] = gOmega+0.1
    gSigma,gOmega = couplingConstants(gSigmaRange,gOmegaRange,N)
    return gSigma,gOmega





##################################################################################
#Plots the binding energy. choose nc = 'n' for normal view, and nc = 'z' for zoom#
##################################################################################
def plotBindingEnergy(kFvec,epsilonVec,nc):
    plt.figure()
    kFvecMeV = 939*kFvec
    epsilonVecMeV2 = epsilonVec[0]*939**4
    epsilonVecMeV4 = epsilonVec[1]*939**4
    rho2 = density(kFvecMeV,2)
    rho4 = density(kFvecMeV,4)
    m = 939*np.ones(len(kFvec))
    B2 = epsilonVecMeV2/rho2-m
    B4 = epsilonVecMeV4/rho4-m
    B2[0] = 0
    B4[0] = 0
    zero = np.zeros(len(kFvec))
    rho2 = MeVtoFm(density(kFvecMeV,2),3)
    rho4 = MeVtoFm(density(kFvecMeV,4),3)
    plt.plot(rho4,zero,'g--')
    plt.plot(rho2,B2,'r',label = 'f=2')
    plt.plot(rho4,B4,'b',label = 'f=4')

    if(nc=='n'):
        plt.xlim(0,0.5)
        plt.ylim(-20,100)
    elif(nc=='z'):
        plt.xlim(0,0.3)
        plt.ylim(-1,4)

    ax = plt.gca()
    ax.xaxis.set_label_coords(0.9, -0.07)
    ax.set_xlabel('$\\rho$ [fm$^{-3}$]', fontsize = 14)
    ax.yaxis.set_label_coords(-0, 1.04)
    ax.set_ylabel('$B(\\rho)$ [MeV]', rotation='horizontal',fontsize = 14)
    plt.legend()



#####################################################
#Function that plots all the figures given in thesis#
#####################################################
```

```
def plotEverything(kFvec,Pvec,epsilonVec,fgEpsilonVec,fgPvec,PvecCorr,epsilonVecCoupl,PvecCoupl,filenameVec):
    colour = ['r','b']
    colorCorr = ['y','g']
    colourCoupl = ['m','k']
    colourVec = [colour,colorCorr,colourCoupl]
    plotEoS(Pvec,epsilonVec)
    plotEoSDifferentCouplings(PvecCoupl,epsilonVecCoupl,Pvec,epsilonVec)
    plotPV(kFvec,Pvec,epsilonVec)
    plotCorrectedPressure(kFvec,PvecCorr,Pvec)
    plotAllMassRadiusRelationsFromFile(filenameVec,colourVec)
    plotBindingEnergy(kFvec,epsilonVec,'n')
    plotBindingEnergy(kFvec,epsilonVec,'z')



######################################################################
#Function that checks if the Maxwell construction is possible for    #
#a given EoS. Returns a string containg a message that can be printed#
######################################################################
def isMaxwellPossible(epsilonVec,Pvec,N1):
    index,flags = findIndex([0,0],Pvec,N1)
    text = ''
    if(flags[0] == False): #Checks if the function is strictly increasing#
        text+='Maxwell construction not nercecary for f=2. Function is strictly increasing!\n'
    if(flags[1] == 0):
        text+='Maxwell construction not nercecary for f=4. Function is strictly increasing!\n'
    index2 = index[0]
    index4 = index[1]
    epsilonVec2 = epsilonVec[0]
    epsilonVec4 = epsilonVec[1]
    Pvec2 = Pvec[0]
    Pvec4 = Pvec[1]

    if(flags[0] == True):
        firstArea2 = area(epsilonVec2,Pvec2,index2[0],index2[1])
        secondArea2 = area(epsilonVec2,Pvec2,index2[1],index2[2])
        if(abs(firstArea2)<abs(secondArea2)):
            text+='Maxwell construction not possible for f=2\n'
        else:
            text+='Maxwell construction possible for f=2\n'
    if(flags[1] == True):
        firstArea4 = area(epsilonVec4,Pvec4,index4[0],index4[1])
        secondArea4 = area(epsilonVec4,Pvec4,index4[1],index4[2])
        if(abs(firstArea4)<abs(secondArea4)):
            text+='Maxwell construction not possible for f=4\n'
        else:
            text+='Maxwell construction possible for f=4\n'
    return text



########################
###                  ###
###   Progran start  ###
###                  ###
########################


#Defining constants and parameters

mSigma = 0.5857294994675186368     #Mass of sigma divided by nucleon mass
mOmega = 0.8338658146964856230     #Mass of omega divided by nucleon mass

```

```
R0 = 1.47               #Half the suns Swarzchild radius
beta = 1.1426           #Parameter found in thesis
kFMax = 2               #Maximum nucleon Fermi momentum used to calculate EoS
N = 1000                #Number of divisions of the interall we search for the coupling constants
N1 = 50000              #Number of points used to calculate EoS
N2 = 300                #Number of points used to calculate mass-radius relation
PcMin = 10**(-10)       #Minimum central pressure used in mass-radius relation
PcMax = 10**(4)         #Maximum central pressure used in mass-radius relation
h = 0.0005              #Step length in km used to calculate the mass-radius relation
nMax = int(100./h)      #Maximum number of iterations before we stop the mass calculations

filename = 'SigmaOmegaMassRadiRelation1'
filenameCorr = 'SigmaOmegaMassRadiRelationCorr1'
filenameCoupl = 'SigmaOmegaMassRadiRelationCoupl1'

#Create a vector containing all filenames
filenameVec = [filename,filenameCorr,filenameCoupl]

#Create vector containing important parameters
parameters = [PcMin,PcMax,N2,h,nMax,kFMax]

#Finds the coupling constants
gSigma,gOmega = couplings(N)

#The progrm that finds the couplings uses a bit of time. If desired, one
#can just remove the comment below and use the obtained values
#gSigma = 10.94
#gOmega = 13.59

#Creates all vectors
kFvec,Pvec,epsilonVec,fgEpsilonVec,fgPvec,PvecCorr,epsilonVecCoupl,PvecCoupl= createVectors(kFMax,N1,gSigma,gOmega)

#Writes all mass-radius relations to file
writeAllResultsToFile(parameters,epsilonVec,Pvec,PvecCorr,PvecCoupl,filenameVec)

#Print if the maxwell construction is possible
print(isMaxwellPossible(epsilonVec,Pvec,N1))

#Produce all plots given in thesis
plotEverything(kFvec,Pvec,epsilonVec,fgEpsilonVec,fgPvec,PvecCorr,epsilonVecCoupl, PvecCoupl,filenameVec)

#Show plots
plt.show()

#Print the time spent
print("\nTime spent:")
print(time.clock()-t0)
```

G.3 Chapter 6, 7 and 8

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt
import time
from numba import jit
from scipy import optimize as op
import math
from scipy import interpolate

##########################################################
##########################################################
```

```
###                                                   ###
###   Just press play to create all plots in thesis!   ###
###                                                   ###
#########################################################
#########################################################

#variable used to time the program
t0 = time.clock()

##############################
###                        ###
###   Defining functions   ###
###                        ###
##############################

#######################################################
#Function that takes inn the hyperon fermi momenta and#
#sets the ones that are negative to zero              #
#######################################################
def isNegative(kFhyperonVec):
    klambda,ksminus,ks0,ksplus,kximinus,kxi0 = kFhyperonVec[:]
    if(ksminus<0):
        ksminus = 0
    if(klambda<0):
        klambda = 0
    if(ks0<0):
        ks0 = 0
    if(ksplus<0):
        ksplus = 0
    if(kximinus<0):
        kximinus = 0
    if(kxi0<0):
        kxi0 = 0
    return [klambda,ksminus,ks0,ksplus,kximinus,kxi0]

###############################################################
#Takes in a number in fm^exponent the exponent and returns the#
#value in MeV^-exponent                                       #
###############################################################
def fmToMeV(number,exponent):
    if(exponent>0):
        return (number**(1./exponent)/197.33)**exponent
    else:
        return (number**(-1./exponent)*197.33)**(-exponent)

#############################
#Inverse function of fmToMeV#
#############################
def MeVtoFm(number,exponent):
    if(exponent>0):
        return (number**(1./exponent)/197.33)**exponent
    else:
        return (number**(-1./exponent)*197.33)**(-exponent)

################################################
#Takes in Fermi momenta and returns the density#
################################################
def density(kF):
    return kF**3/(3*math.pi**2)

###########################################################
#Takes in a particle density and returns the Fermi momenta#
###########################################################
def momentaFromDensity(dens):
    return (3*math.pi**2*dens)**(1./3)

```

```
##############################################################
#Takes in the Fermi momenta and effective mass for a particle#
#and returns its contribution to the scalar density          #
##############################################################
def scalarDensityFunc(kF,mStar):
    if(kF == 0 or mStar == 0):
        return 0.
    else:
        mem1 = mStar/(2*math.pi**2)
        mem2 = np.sqrt(kF**2+mStar**2)*kF
        mem3 = mStar**2*np.arcsinh(kF/mStar)
        return mem1*(mem2-mem3)

##################################
#Returns the total scalar density#
##################################
@jit
def scalarDensity(omega,rho,dens,ke,kMu,kn,kp,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp):
    mStarLambda = mLambda-gSigma*xSigma[0]*sigmaField(mStar,gSigma)
    mStarS = mS-gSigma*xSigma[1]*sigmaField(mStar,gSigma)
    mStarXi = mXi-gSigma*xSigma[2]*sigmaField(mStar,gSigma)
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    mem1 = scalarDensityFunc(kn,mStar)+scalarDensityFunc(kp,mStar)
    mem2 = xSigma[0]*scalarDensityFunc(kFhyperonVec[0],mStarLambda)
    mem3 = xSigma[1]*(scalarDensityFunc(kFhyperonVec[1],mStarS)+scalarDensityFunc(kFhyperonVec[2],mStarS)+scalarDensityFunc(kFhyperonVec[3],mStarS))
    mem4 = xSigma[2]*(scalarDensityFunc(kFhyperonVec[4],mStarXi)+scalarDensityFunc(kFhyperonVec[5],mStarXi))
    return mem1+mem2+mem3+mem4

##########################################
#Calculates the Fermi momenta for the proton#
##########################################
@jit
def kpFunc(ke,kMu,kFhyperonVec):
    if(ke<0):
        ke=0
    mem1 = kFhyperonVec[1]**3 - kFhyperonVec[3]**3+ kFhyperonVec[4]**3
    if(ke**3+kMu**3+mem1<0):
        return 0
    return (ke**3+kMu**3+mem1)**(1./3)

############################################
#Gives the expectation value for the sigma field#
############################################
@jit
def sigmaField(mStar,gSigma):
    return (m-mStar)/gSigma

#########################################################
#Gives a function for the expectation of the omega field to be#
#used by root solver function.                          #
#########################################################
@jit
def omegaFieldFunc(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xOmega,hyp):
    kMu = kMuFunc(ke)
    if(ke<0):
        ke=0
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kFhyperonVec = isNegative(kFhyperonVec)
    kp = kpFunc(ke,kMu,kFhyperonVec)
    mem = 0.
    mem+=density(kn)+density(kp)
    mem+=xOmega[0]*density(kFhyperonVec[0])
    mem+=xOmega[1]*(density(kFhyperonVec[1])+density(kFhyperonVec[2])+density(kFhyperonVec[3]))
    mem+=xOmega[2]*(density(kFhyperonVec[4])+density(kFhyperonVec[5]))
    return (omega-mem*gOmega*mOmega**(-2.))

##########################################
#Returns the expectation of the rho field#
##########################################
@jit
def rhoField(kn,kp,ksminus,ksplus,kximinus,kxi0,gRho,xRho):
    return 0.5*gRho*mRho**(-2.)*(density(kn)-density(kp)+2*xRho[1]*density(ksminus)-2*xRho[1]*density(ksplus)+xRho[2]*density(kximinus)-xRho[2]*density(kxi0))

##########################################################
#Gives function for the expectation of the rho field that#
#can be used by root solver                              #
##########################################################
@jit
def rhoFieldFunc(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp):
    if(ke<0):
        ke=0
    kMu = kMuFunc(ke)
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kFhyperonVec = isNegative(kFhyperonVec)
    kp = kpFunc(ke,kMu,kFhyperonVec)
    mem1 = density(kn)-density(kp)
    mem2 = 2*xRho[1]*density(kFhyperonVec[1])-2*xRho[1]*density(kFhyperonVec[3])
    mem3 = xRho[2]*density(kFhyperonVec[4])-xRho[2]*density(kFhyperonVec[5])
    return (rho-0.5*gRho*mRho**(-2.)*(mem1+mem2+mem3))

############################################
#Returns the Fermi momenta for the electron#
############################################
@jit
def keFermi(dens,kFvec):
    kF3 = dens*3*math.pi**2
    mem1 = kFvec[1]**3+kFvec[2]**3
    mem2 = kFvec[4]**3+2*kFvec[5]**3
    mem3 = kFvec[6]**3+2*kFvec[8]**3
    mem4 = kFvec[9]**3
    return (kF3-mem1-mem2-mem3-mem4)**(1./3)

####################################################
#Gives function for the electron Fermi momenta that#
#can be used by root solver                        #
####################################################
@jit
def keFunc(dens,omega,rho,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp):
    if(ke<0):
        ke=0
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kFhyperonVec = isNegative(kFhyperonVec)
    kMu = kMuFunc(ke)
    kF3 = dens*3*math.pi**2
    mem1 = kMu**3+kn**3
    mem2 = kFhyperonVec[0]**3
    mem3 = 2*kFhyperonVec[1]**3+kFhyperonVec[2]**3
    mem4 = 2*kFhyperonVec[4]**3+kFhyperonVec[5]**3
    if(kF3-mem1-mem2-mem3-mem4<0):
        return ke
    return (ke-(kF3-mem1-mem2-mem3-mem4)**(1./3))

########################################
```

```python
#Returns the Fermi momenta for the muon#
########################################
@jit
def kMuFunc(ke):
    if(ke**2+me**2-mMu**2>0):
        return np.sqrt(ke**2+me**2-mMu**2)
    else:
        return 0.

###################################################
#Gives function for the neutron Fermi momenta that#
#can be used by root solver                       #
###################################################
@jit
def knFunc(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xRho,hyp):
    if(ke<0):
        ke=0
    kMu = kMuFunc(ke)
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kFhyperonVec = isNegative(kFhyperonVec)
    kp = kpFunc(ke,kMu,kFhyperonVec)
    mem = -gRho*rho+np.sqrt(ke**2+me**2)+np.sqrt(kp**2+mStar**2)
    if(mem**2-mStar**2<=0):
        return kn
    return (kn-(mem**2-mStar**2)**(1./2))

#############################################################
#Gives function for the effective mass for the nucleons that#
#can be used by root solver                                 #
#############################################################
@jit
def mStarFunc(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,b,c,xSigma,xOmega,xRho,hyp):
    if(ke<0):
        ke=0
    kMu = kMuFunc(ke)
    mem1 = (m-mStar)
    mem2 = (gSigma**2*mSigma**(-2))
    kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kFhyperonVec = isNegative(kFhyperonVec)
    kp = kpFunc(ke,kMu,kFhyperonVec)
    mem3 = scalarDensity(omega,rho,dens,ke,kMu,kn,kp,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    mem4 = b*m*mem1**2+c*mem1**3
    if(mem3+mem4==0):
        return mem1
    return (mem1-(mem2*(mem3+mem4)))

################################################
#Returns the chemical potential for the neutron#
################################################
@jit
def munFunc(omega,rho,kn,mStar,gOmega,gRho,xOmega,xRho):
    return np.sqrt(kn**2+mStar**2)+gOmega*omega+0.5*gRho*rho

###############################################################################
#Sets the functions knFunc, mStarFunc, rhoFieldFunc, keFunc and omegaFieldFunc#
#together to one vector function to be used in root solver function           #
###############################################################################
def knMstarFunc(keOrDens,gSigma,gOmega,gRho,xOmega,xRho,b,c,nr,hyp):
    if(hyp==True and nr=='n'):
        return lambda var:[knFunc(var[0],var[1],keOrDens,var[2],var[3],var[4],gOmega,gSigma,gRho,xRho,hyp),
                            mStarFunc(var[0],var[1],keOrDens,var[2],var[3],var[4],gSigma,gOmega,gRho,b,c,xSigma,xOmega,xRho,hyp),
                             rhoFieldFunc(var[0],var[1],keOrDens,var[2],var[3],var[4],gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp),
                             keFunc(keOrDens,var[0],var[1],var[2],var[3],var[4],gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp),
                             omegaFieldFunc(var[0],var[1],keOrDens,var[2],var[3],var[4],gSigma,gOmega,gRho,xOmega,hyp)]

    elif(nr=='n'):
        return lambda var:[knFunc(var[0],var[1],0,keOrDens,var[2],var[3],gOmega,gSigma,gRho,xRho,hyp),
                             mStarFunc(var[0],var[1],0,keOrDens,var[2],var[3],gSigma,gOmega,gRho,b,c,xSigma,xOmega,xRho,hyp),
                             rhoFieldFunc(var[0],var[1],0,keOrDens,var[2],var[3],gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp),
                             omegaFieldFunc(var[0],var[1],0,keOrDens,var[2],var[3],gSigma,gOmega,gRho,xOmega,hyp)]
    elif(nr=='r'):
        kMu = kMuFunc(keOrDens)
        kp = kpFunc(keOrDens,kMu,[0,0,0,0,0,0])
        return lambda var:[knFunc(var[0],var[1],0,keOrDens,var[2],var[3],gOmega,gSigma,gRho,xRho,hyp),
                             renormalizedMstarFunc(keOrDens,var[2],kp,var[3],gSigma,gOmega,gRho,b,c),
                             rhoFieldFunc(var[0],var[1],0,keOrDens,var[2],var[3],gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp),
                             omegaFieldFunc(var[0],var[1],0,keOrDens,var[2],var[3],gSigma,gOmega,gRho,xOmega,hyp)]

#################################################
#Returns the Fermi momenta for a type of hyperon#
#################################################
@jit
def fermiHyperonFunc(omega,rho,ke,kn,q,I,mStar,mStarB,gOmega,gRho,xSigma,xOmega,xRho,index):
    mun = munFunc(omega,rho,kn,mStar,gOmega,gRho,xOmega,xRho)
    mem = (mun-q*np.sqrt(ke**2+me**2)-gOmega*xOmega[index]*omega-gRho*xRho[index]*I*rho)**2-mStarB**2
    if(mem>0):
        return np.sqrt(mem)
    else:
        return 0

########################################################
#Returns the Fermi momenta for the hyperons in a vector#
########################################################
def fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp):
    if(hyp==False):
        return [0,0,0,0,0,0]
    mStarLambda = mLambda-gSigma*xSigma[0]*sigmaField(mStar,gSigma)
    mStarS = mS-gSigma*xSigma[1]*sigmaField(mStar,gSigma)
    mStarXi = mXi-gSigma*xSigma[2]*sigmaField(mStar,gSigma)
    klambda = fermiHyperonFunc(omega,rho,ke,kn,0,0,mStar,mStarLambda,gOmega,gRho,xSigma,xOmega,xRho,0)
    ksminus = fermiHyperonFunc(omega,rho,ke,kn,-1,1,mStar,mStarS,gOmega,gRho,xSigma,xOmega,xRho,1)
    ks0 = fermiHyperonFunc(omega,rho,ke,kn,0,0,mStar,mStarS,gOmega,gRho,xSigma,xOmega,xRho,1)
    ksplus = fermiHyperonFunc(omega,rho,ke,kn,1,-1,mStar,mStarS,gOmega,gRho,xSigma,xOmega,xRho,1)
    kximinus = fermiHyperonFunc(omega,rho,ke,kn,-1,0.5,mStar,mStarXi,gOmega,gRho,xSigma,xOmega,xRho,2)
    kxi0 = fermiHyperonFunc(omega,rho,ke,kn,0,-0.5,mStar,mStarXi,gOmega,gRho,xSigma,xOmega,xRho,2)
    return [klambda,ksminus,ks0,ksplus,kximinus,kxi0]

#########################
```

```
#Returns the omega field#
#########################
@jit
def omegaField(kFvec,gOmega,xOmega):
    omega = 0
    omega+=density(kFvec[2])+density(kFvec[3])
    omega+=xOmega[0]*density(kFvec[4])
    omega+=xOmega[1]*(density(kFvec[5])+density(kFvec[6])+density(kFvec[7]))
    omega+=xOmega[2]*(density(kFvec[8])+density(kFvec[9]))
    return gOmega*mOmega**(-2.)*omega

################################################
#Returns all Fermi momenta and effective masses#
################################################
@jit
def fermiMomentaAndEffectiveMass(keOrDens,guesses,gSigma,gOmega,gRho,xSigma,xOmega,xRho,b,c,nr,hyp):
    if(hyp==False):
        ke = keOrDens
        xSigma = [0,0,0]
        xRho = [0,0,0]
        xOmega = [0,0,0]
        solution = op.root(knMstarFunc(ke,gSigma,gOmega,gRho,xOmega,xRho,b,c,nr,hyp),guesses)
        omega,rho,kn,mStar = solution.x
        kFhyperonVec = [0,0,0,0,0,0]


    else:
        dens = keOrDens
        solution = op.root(knMstarFunc(dens,gSigma,gOmega,gRho,xOmega,xRho,b,c,nr,hyp),guesses,options={'factor':0.00003})
        omega,rho,ke,kn,mStar = solution.x
        kFhyperonVec = fermiHyperon(omega,rho,dens,ke,kn,mStar,gSigma,gOmega,gRho,xSigma,xOmega,xRho,hyp)
    kMu = kMuFunc(ke)
    klambda,ksminus,ks0,ksplus,kximinus,kxi0 = kFhyperonVec[:]
    kp = kpFunc(ke,kMu,kFhyperonVec)
    mStarLambda = mLambda-gSigma*xSigma[0]*sigmaField(mStar,gSigma)
    mStarS = mS-gSigma*xSigma[1]*sigmaField(mStar,gSigma)
    mStarXi = mXi-gSigma*xSigma[2]*sigmaField(mStar,gSigma)

    return omega,rho,ke,kMu,kn,kp,klambda,ksminus,ks0,ksplus,kximinus,kxi0,mStar,mStarLambda,mStarS,mStarXi,solution.success

################################################
#Returns the Fermi momenta for a free Fermi gas#
################################################
@jit
def freeFermiGasEnergyDensity(kF,mStar):
    if(kF==0):
        return 0.
    else:
        mem1 = (24*math.pi**2)**(-1)
        mem2 = np.sqrt(kF**2+mStar**2)
        mem3 = 6*kF**3+3*mStar**2*kF
        if(mStar==0):
            mem4 = 0.
        else:
            mem4 = 3*mStar**4*np.arcsinh(kF/mStar)
        return mem1*(mem2*mem3-mem4)

###########################################
#Returns the pressure for a Free fermi gas#
###########################################
@jit
```

```
def freeFermiGasPressure(kF,mStar):
    if(kF==0):
        return 0.
    else:
        mem1 = (24*math.pi**2)**(-1)
        mem2 = np.sqrt(kF**2+mStar**2)
        mem3 = 2*kF**3-3*mStar**2*kF
        if(mStar==0):
            mem4 = 0.
        else:
            mem4 = 3*mStar**4*np.arcsinh(kF/mStar)
        return mem1*(mem2*mem3+mem4)

######################
#Returns the pressure#
######################
@jit
def pressure(kFvec,mStar,mStarLambda,mStarS,mStarXi,gSigma,gOmega,gRho,b,c,xRho,nr,hyp):
    if(kFvec[0]==0 and kFvec[3]==0):
        return 0.
    elif(nr=='r'):
        return renormalizedPressure(kFvec,mStar,gSigma,gOmega,gRho,b,c)
    else:
        ke,kMu,kn,kp,klambda,ksminus,ks0,ksplus,kximinus,kxi0 = kFvec[:]
        sigma = sigmaField(mStar,gSigma)
        omega = omegaField(kFvec,gOmega,xOmega)
        rho = rhoField(kn,kp,ksminus,ksplus,kximinus,kxi0,gRho,xRho)
        mem1 = -0.5*mSigma**2*sigma**2+1./3*b*m*(gSigma*sigma)**3+1./4*c*(gSigma*sigma)**4
        mem2 = 0.5*mOmega**2*omega**2+0.5*mRho**2*rho**2
        mem3 = freeFermiGasPressure(ke,me)
        mem4 = freeFermiGasPressure(kMu,mMu)
        mem5 = freeFermiGasPressure(kn,mStar)
        mem6 = freeFermiGasPressure(kp,mStar)
        mem7 = freeFermiGasPressure(klambda,mStarLambda)
        mem8 = freeFermiGasPressure(ksminus,mStarS)
        mem9 = freeFermiGasPressure(ks0,mStarS)
        mem10 = freeFermiGasPressure(ksplus,mStarS)
        mem11 = freeFermiGasPressure(kximinus,mStarXi)
        mem12 = freeFermiGasPressure(kxi0,mStarXi)
        return mem1+mem2+mem3+mem4+mem5+mem6+mem7+mem8+mem9+mem10+mem11+mem12

############################
#Returns the energy density#
############################
@jit
def energyDensity(kFvec,mStar,mStarLambda,mStarS,mStarXi,gSigma,gOmega,gRho,b,c,xRho,nr,hyp):
    if(kFvec[0]==0 and kFvec[3]==0):
        return 0.
    elif(nr=='r'):
        return renormalizedEnergyDensity(kFvec,mStar,gSigma,gOmega,gRho,b,c)
    else:
        ke,kMu,kn,kp,klambda,ksminus,ks0,ksplus,kximinus,kxi0 = kFvec[:]
        sigma = sigmaField(mStar,gSigma)
        omega = omegaField(kFvec,gOmega,xOmega)
        rho = rhoField(kn,kp,ksminus,ksplus,kximinus,kxi0,gRho,xRho)
        mem1 = 0.5*mSigma**2*sigma**2-1./3*b*m*(gSigma*sigma)**3-1./4*c*(gSigma*sigma)**4
        mem2 = 0.5*mOmega**2*omega**2+0.5*mRho**2*rho**2
        mem3 = freeFermiGasEnergyDensity(ke,me)
        mem4 = freeFermiGasEnergyDensity(kMu,mMu)
        mem5 = freeFermiGasEnergyDensity(kn,mStar)
        mem6 = freeFermiGasEnergyDensity(kp,mStar)
        mem7 = freeFermiGasEnergyDensity(klambda,mStarLambda)
        mem8 = freeFermiGasEnergyDensity(ksminus,mStarS)
        mem9 = freeFermiGasEnergyDensity(ks0,mStarS)
        mem10 = freeFermiGasEnergyDensity(ksplus,mStarS)
        mem11 = freeFermiGasEnergyDensity(kximinus,mStarXi)
        mem12 = freeFermiGasEnergyDensity(kxi0,mStarXi)
        return mem1+mem2+mem3+mem4+mem5+mem6+mem7+mem8+mem9+mem10+mem11+mem12

###########################
###                     ###
###   Renromalization   ###
###                     ###
###########################

####################
#f'' from chapter 7#
####################
@jit
def fDoubleDer(mStar,gSigma,b,c):
    return 2*b*m*gSigma**3*sigmaField(mStar,gSigma)+3*c*gSigma**4*sigmaField(mStar,gSigma)**2

#####################
#f''' from chapter 7#
#####################
@jit
def fTrippelDer(mStar,gSigma,b,c):
    return 2*b*gSigma**3+6*c*gSigma**4*sigmaField(mStar,gSigma)

##################
#U from chapter 7#
##################
@jit
def U(mStar,gSigma):
    if(sigmaField(mStar,gSigma)<=0.):
        return 0.
    mem1 = -(2*math.pi)**(-2)
    if(mStar==0.):
        mem2 = 0.
    else:
        mem2 = mStar**4*np.log(mStar/m)
    mem3 = m**3*(m-mStar)-7./2*m**2*(m-mStar)**2
    mem4 = 13./3*m*(m-mStar)**3-25./12*(m-mStar)**4
    return mem1*(mem2+mem3+mem4)

####################################
#The derivative of U from chapter 7#
####################################
@jit
def derU(mStar):
    mem1 = (math.pi)**(-2)
    if(mStar <= 0.):
        mem2 = 0.
    else:
        mem2 = mStar**3*np.log(mStar/m)
    mem3 = m**2*(m-mStar)
    mem4 = -5./2*m*(m-mStar)**2
    mem5 = 11./6*(m-mStar)**3
    return mem1*(mem2+mem3+mem4+mem5)



##################
#V from chapter 7#
##################
@jit
def V(mStar,gSigma,b,c):
    mem1 = fDoubleDer(mStar,gSigma,b,c)
    if(mSigma**2-mem1<=0):
        mem2 = 0.
    else:
        mSigmaStar = np.sqrt(mSigma**2-mem1)
        mem2 = mSigmaStar**4*np.log(mSigmaStar/mSigma)
    mem3 = 2*b*m*gSigma**3*sigmaField(mStar,gSigma)
    mem4 = 3*c*gSigma**4*sigmaField(mStar,gSigma)**2
    return (mem2-0.5*(-mSigma**2*mem1+3./2*mem1**2-(mem3**3+3*mem3**2*mem4)/(3*mSigma**2)-mem3**4/(12*mSigma**4)))/(32*math.pi**2)

##################################
#Derivative of V from chapter 7#
##################################
@jit
def derV(mStar,gSigma,b,c):
    mem1 = fDoubleDer(mStar,gSigma,b,c)
    if(mSigma**2-mem1<0):
        mSigmaStar = 0.
    else:
        mSigmaStar = np.sqrt(mSigma**2-mem1)
    mem2 = fTrippelDer(mStar,gSigma,b,c)
    mem3 = 2*b*m*gSigma**3
    mem4 = 3*c*gSigma**4
    mem5 = sigmaField(mStar,gSigma)
    if(mSigmaStar <= 0.):
        mem6 = 0.
    else:
        mem6 = (4*mSigmaStar**3*np.log(mSigmaStar/mSigma)+mSigmaStar**3)*mem2/(2*gSigma*mSigmaStar)
    return -(mem6+0.5/gSigma*(-mSigma**2*mem2+3*mem2*mem1-(mem3**3*mem5**2+4*mem3**2*mem4*mem5**3)/mSigma**2-(mem3**4*mem5**3)/(3*mSigma**4)))/(32*math.pi**2)

####################################################################
#Same as mStarFunc except that renormalization is accounted for#
####################################################################
@jit
def renormalizedMstarFunc(ke,kn,kp,mStar,gSigma,gOmega,gRho,b,c):
    kMu = kMuFunc(ke)
    mem1 = m-mStar
    mem2 = gSigma**2*mSigma**(-2)
    mem3 = b*m*mem1**2
    mem4 = c*mem1**3
    omega = omegaField([0,0,kn,kp,0,0,0,0,0,0],gOmega,xOmega)
    rho = rhoField(kn,kp,0,0,0,0,gRho,xRho)
    mem5 = scalarDensity(omega,rho,0,ke,kMu,kn,kp,mStar,gSigma,gOmega,gRho,[0,0,0],[0,0,0],[0,0,0],False)
    return 1.-mem1/(mem2*(mem3+mem4+mem5-derU(mStar)-derV(mStar,gSigma,b,c)))

########################################################################
#Same as energyDensity except that renormalization is accounted for#
########################################################################
@jit
def renormalizedEnergyDensity(kFvec,mStar,gSigma,gOmega,gRho,b,c):
    mem1 = energyDensity(kFvec,mStar,0,0,0,gSigma,gOmega,gRho,b,c,[0,0,0],'n',False)
    mem2 = V(mStar,gSigma,b,c)
    mem4 = U(mStar,gSigma)
    return mem1+mem2+mem4

##################################################################
#Same as pressure except that renormalization is accounted for#
##################################################################
@jit
def renormalizedPressure(kFvec,mStar,gSigma,gOmega,gRho,b,c):
    mem1 = pressure(kFvec,mStar,0,0,0,gSigma,gOmega,gRho,b,c,[0,0,0],'n',False)
    mem2 = V(mStar,gSigma,b,c)
    mem4 = U(mStar,gSigma)
    return mem1-mem2-mem4




#############################################################
###                                                       ###
###   Functions used to calculate the coupling constants  ###
###                                                       ###
#############################################################

################################################################
#Same as mStarFunc except that it only holds for nuclear matter#
################################################################
@jit
def nuclearMatterMassFunc(kF,mStar,gSigma,gOmega,gRho,b,c,nr,hyp):
    mem1 = (m-mStar)
    mem2 = gSigma**2*mSigma**(-2)
    omega = omegaField([0,0,kF,kF,0,0,0,0,0,0],gOmega,xOmega)
    rho = rhoField(kF,kF,0,0,0,0,gRho,[0,0,0])
    mem3 = scalarDensity(omega,rho,0,0,0,kF,kF,mStar,gSigma,gOmega,gRho,[0,0,0],[0,0,0],[0,0,0],hyp)
    mem4 = b*m*mem1**2+c*mem1**3
    if(nr=='r'):
        return -mem1+mem2*(mem3+mem4-derU(mStar)-derV(mStar,gSigma,b,c))
    return -mem1+mem2*(mem3+mem4)

#####################################################
#Finds the effective nucleon mass for nuclear matter#
#####################################################
def nuclearEffectiveMass(kF,gSigma,gOmega,gRho,b,c,nr,hyp):
    if(kF==0):
        return 1.
    else:
        func = lambda mStar: nuclearMatterMassFunc(kF,mStar,gSigma,gOmega,gRho,b,c,nr,hyp)
        if(np.sign(func(0))==np.sign(func(1))):
            return 0.
        else:
            dsad = op.brentq(func,0,1)
            return dsad

############################################################################
#Returns the effective mass, energy density and pressure for nuclear matter#
#for a given Fermi momenta                                                 #
############################################################################
@jit
def nuclearMatterEoS(kF,gSigma,gOmega,gRho,b,c,nr,hyp):
    mStar = nuclearEffectiveMass(kF,gSigma,gOmega,gRho,b,c,nr,hyp)
    epsilon = energyDensity([0,0,kF,kF,0,0,0,0,0,0],mStar,0,0,0,gSigma,gOmega,gRho,b,c,[0,0,0],nr,hyp)
    P = pressure([0,0,kF,kF,0,0,0,0,0,0],mStar,0,0,0,gSigma,gOmega,gRho,b,c,[0,0,0],nr,hyp)
    return mStar,epsilon,P

###############################################################################
#Finds the rho coupling given a Fermi momenta and effective mass at saturation#
#as well as the symmetry energy coefficient as                                #
###############################################################################
@jit
def findgRho(kFsat,mStar,aS):
    mem1 = aS-kFsat**2/(6*np.sqrt(kFsat**2+mStar**2))
    mem2 = 12*math.pi**2*kFsat**(-3)
    gRho = mRho*np.sqrt(mem1*mem2)
    return gRho

```

```
#############################################################
#Function used to calculate the effective mass at saturation#
#############################################################
@jit
def mStarFuncCouplings(kFsat,gSigma,gOmega,gRho,b,c,nr):
    return nuclearMatterEoS(kFsat,gSigma,gOmega,gRho,b,c,nr,False)[0]

##########################################
#Returns the binding energy at saturation#
##########################################
@jit
def Bfunc(kFsat,satDens,gSigma,gOmega,gRho,b,c,nr):
    epsilon = nuclearMatterEoS(kFsat,gSigma,gOmega,gRho,b,c,nr,False)[1]
    return epsilon/satDens-m

###################################################
#Returns the criteria that dB/drho=0 at saturation#
###################################################
@jit
def densFunc(kFsat,satDens,dkF,dRho,gSigma,gOmega,gRho,b,c,nr):
    epsilon = nuclearMatterEoS(kFsat,gSigma,gOmega,gRho,b,c,nr,False)[1]
    epsilonPlus = nuclearMatterEoS(kFsat+dkF,gSigma,gOmega,gRho,b,c,nr,False)[1]
    epsilonMinus = nuclearMatterEoS(kFsat-dkF,gSigma,gOmega,gRho,b,c,nr,False)[1]
    return (satDens)**(-1)*(epsilonPlus-epsilonMinus)**(-1.)*dRho*epsilon

#################################
#Returns the compression modolus#
#################################
@jit
def Kfunc(kFsat,satDens,dkF,dRho,gSigma,gOmega,gRho,b,c,nr):
    epsilon = nuclearMatterEoS(kFsat,gSigma,gOmega,gRho,b,c,nr,False)[1]
    epsilonPlus = nuclearMatterEoS(kFsat+dkF,gSigma,gOmega,gRho,b,c,nr,False)[1]
    epsilonMinus = nuclearMatterEoS(kFsat-dkF,gSigma,gOmega,gRho,b,c,nr,False)[1]
    epsilonDensPlus = epsilonPlus/(2*density(kFsat+dkF))
    epsilonDens = epsilon/satDens
    epsilonDensMinus = epsilonMinus/(2*density(kFsat-dkF))
    return kFsat**2*(epsilonDensPlus+epsilonDensMinus-2*epsilonDens)*dkF**(-2)

#######################################################################
#Function that returns the values of computed binding energy and        #
#saturation density for a given set of couplings and the empirical value#
#######################################################################
def bindingEnergyAndSaturationDensityCheck(kFvec,satDens,satB,nuclearEpsilonVec,N):
    k = kFvec[0]
    dens = 2*density(k[1:N])
    nuclearEpsilonVec = nuclearEpsilonVec[1:N]
    N=N-1
    epsilon = interpolate.interp1d(dens,nuclearEpsilonVec)
    func = lambda den: epsilon(den)/den - m
    satDensIndex = list(func(dens)).index(min(func(dens)))
    B = min(func(dens))
    satDens = MeVtoFm(satDens*939**3,3)
    return satDensIndex,[MeVtoFm(dens[satDensIndex]*939**3,3),satDens],[B*939,satB*939]

#############################################################
#Function that returns the values of computed effective mass#
#for a given set of couplings and the empirical value       #
#############################################################
def mStarCheck(kFvec,satMstar,nuclearMstarVec,satDensIndex,N):
    return [np.float(nuclearMstarVec[satDensIndex]), satMstar]

##################################################################
#Function that returns the values of computed compression modolus#
#for a given set of couplings and the empirical value            #
##################################################################
def compressionModuloCheck(kFvec,satK,nuclearEpsilonVec,satDensIndex):
```

```python
    k = kFvec[0]
    deltaEpsilon = nuclearEpsilonVec[satDensIndex+1]-nuclearEpsilonVec[satDensIndex-1]
    deltaEpsilon2 = nuclearEpsilonVec[satDensIndex+1]+nuclearEpsilonVec[satDensIndex-1]
    deltaK = k[satDensIndex+1]-k[satDensIndex-1]
    derEpsilon = deltaEpsilon/deltaK
    derEpsilon2 = (deltaEpsilon2-2*nuclearEpsilonVec[satDensIndex])*(0.5*deltaK)**(-2)
    satDens = 2*density(k[satDensIndex])
    satkF = k[satDensIndex]
    satE = nuclearEpsilonVec[satDensIndex]
    mem1 = derEpsilon2
    if(satkF == 0.):
        K = 0.
    else:
        mem2 = 6*derEpsilon/satkF
        mem3 = 6*satE*satkF**(-2)
        mem4 = 18*satE*satkF**(-2)
        K = satkF**2*(mem1-mem2-mem3+mem4)*satDens**(-1)
    return [K*939,satK*939]

###########################################################################
#Function that returns the values of computed symmetry energy coefficient#
#for a given set of couplings and the empirical value                    #
###########################################################################
def symmetryCoefficientCheck(kFvec,satAs,satDensIndex,mStar,gRho):
    k = kFvec[0]
    satkF = k[satDensIndex]
    if(satkF==0.):
        S = 0.
    else:
        mem1 = gRho**2*mRho**(-2)*satkF**3/(12*math.pi**2)
        mem2 = satkF**2/(6*(satkF**2+mStar**2)**(0.5))
        S = (mem1+mem2)*939
    return [S,satAs*939]

############################################################
#Returns the results from all the four check-functions above#
############################################################
def runAllChecks(kFvec,satDens,satB,satK,satMstar,satAs,nuclearMstarVec,nuclearEpsilonVec,gRho):
    N = len(nuclearEpsilonVec)
    satDensIndex,B,D = bindingEnergyAndSaturationDensityCheck(kFvec,satDens,satB,nuclearEpsilonVec,N)
    mStar = mStarCheck(kFvec,satMstar,nuclearMstarVec,satDensIndex,N)
    K = compressionModuloCheck(kFvec,satK,nuclearEpsilonVec,satDensIndex)
    symmetryCoefficientCheck(kFvec,satAs,satDensIndex,mStar[0],gRho)
    return B,D,mStar,K

##################################
#Calculates the coupling constants#
##################################
def couplingConstants(best,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,N,tolerance,nr,hyp):
    kFsat = momentaFromDensity(satDens)/2**(1./3)
    dkF = 10**(-5)
    if(N!=1):
        deltaSigma = (gSigmaRange[1]-gSigmaRange[0])/(N-1)
        deltaOmega = (gOmegaRange[1]-gOmegaRange[0])/(N-1)
        deltab = (bRange[1]-bRange[0])/(N-1)
        deltac = (cRange[1]-cRange[0])/(N-1)
    else:
        deltaSigma = (gSigmaRange[1]-gSigmaRange[0])/2
        deltaOmega = (gOmegaRange[1]-gOmegaRange[0])/2
        deltab = (bRange[1]-bRange[0])/2
        deltac = (cRange[1]-cRange[0])/2
    gSigmaGuess = gSigmaRange[0]-deltaSigma
    gOmegaGuess = gOmegaRange[0]-deltaOmega
    bGuess = bRange[0]-deltab
    cGuess = cRange[0]-deltac
    dRho = 2*(density(kFsat+dkF)-density(kFsat-dkF))
    gRho = findgRho(kFsat,satMstar,satAs)
    func = lambda arg:[(satMstar-mStarFuncCouplings(kFsat,arg[0],arg[1],gRho,arg[2],arg[3],nr)),
                        (satB-Bfunc(kFsat,satDens,arg[0],arg[1],gRho,arg[2],arg[3],nr)),
                        (densFunc(kFsat,satDens,dkF,dRho,arg[0],arg[1],gRho,arg[2],arg[3],nr)),
                        (satK-Kfunc(kFsat,satDens,dkF,dRho,arg[0],arg[1],gRho,arg[2],arg[3],nr))]
    bestgSigma = gSigmaGuess+0.5*(gSigmaRange[1]-gSigmaRange[0])
    bestgOmega = gOmegaGuess+0.5*(gOmegaRange[1]-gOmegaRange[0])
    bestb = bGuess+0.5*(bRange[1]-bRange[0])
    bestc = cGuess+0.5*(cRange[1]-cRange[0])
    solver = 'hybr'
    for i in range(N):
        gSigmaGuess+=deltaSigma
        for j in range(N):
            gOmegaGuess+=deltaOmega
            for l in range(N):
                bGuess+=deltab
                for n in range(N):
                    print((N**4),(n+l*N+j*N*N+i*N*N*N+1),best)
                    cGuess+=deltac
                    solution = op.root(func,[gSigmaGuess,gOmegaGuess,bGuess,cGuess],method=solver)
                    gSigma,gOmega,b,c = solution.x
                    if (solution.success == True and (b<0) and (c<0)):
                        N2 = 100
                        kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear = createVectors(1.,0.03,[gSigma,gSigma],[gOmega,gOmega],[gRho,gRho],[0,0,0],[0,0,0],[0,0,0],[b,b],[c,c],N2,nr,hyp)
                        d,e,f,g = runAllChecks(kFvec,satDens,satB,satK,satMstar,satAs,nuclear[0],nuclear[1],gRho)
                        d = 1.-d[0]/d[1]
                        e = 1.-e[0]/e[1]
                        f = 1.-f[0]/f[1]
                        g = 1.-g[0]/g[1]
                        test = np.sqrt(d**2+e**2+f**2+g**2)
                        if(test<best):
                            N2 = 5000
                            kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear = createVectors(1.,0.03,[gSigma,gSigma],[gOmega,gOmega],[gRho,gRho],[0,0,0],[0,0,0],[0,0,0],[b,b],[c,c],N2,nr,hyp)
                            d,e,f,g = runAllChecks(kFvec,satDens,satB,satK,satMstar,satAs,nuclear[0],nuclear[1],gRho)
                            d = 1.-d[0]/d[1]
                            e = 1.-e[0]/e[1]
                            f = 1.-f[0]/f[1]
                            g = 1.-g[0]/g[1]
                            test = np.sqrt(d**2+e**2+f**2+g**2)
                            if(test<best):
                                best = test
                                print(best)
                                if(best<tolerance):
                                    print("Coupling sucess for",nr)
                                    return gSigma,gOmega,gRho,b,c,best
                            bestgSigma = gSigma
                            bestgOmega = gOmega
                            bestb = b
                            bestc = c
                cGuess = cRange[0]-deltac
            bGuess = bRange[0]-deltab
        gOmegaGuess = gOmegaRange[0]-deltaOmega
    print("Failed to find satisfactory couplings for",nr)
    print(bestgSigma,bestgOmega,gRho,bestb,bestc,best)
    return bestgSigma,bestgOmega,gRho,bestb,bestc,best


#####################################################
#Creates coupling constants, then write them to file#
#####################################################
def writeCouplingConstantsToFile(satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,N,iterations,multiplier,tolerance,nr,hyp,filename):
    best = 1000
    for i in range(iterations-1):
        gSigmatest,gOmegatest,gRhotest,btest,ctest,test = couplingConstants(best,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,N,tolerance,nr,hyp)
        if(test<best):
            best=test
            gSigma = gSigmatest
            gOmega = gOmegatest
            gRho = gRhotest
            b = btest
            c = ctest
            gSigmaRange[1] = gSigma+(gSigmaRange[1]-gSigmaRange[0])/2*multiplier
            gSigmaRange[0] = gSigma-(gSigmaRange[1]-gSigmaRange[0])/2*multiplier
            gOmegaRange[1] = gOmega+(gOmegaRange[1]-gOmegaRange[0])/2*multiplier
            gOmegaRange[0] = gOmega-(gOmegaRange[1]-gOmegaRange[0])/2*multiplier
            bRange[1] = b+(bRange[1]-bRange[0])/2*multiplier
            bRange[0] = b-(bRange[1]-bRange[0])/2*multiplier
            cRange[1] = c+(cRange[1]-cRange[0])/2*multiplier
            cRange[0] = c-(cRange[1]-cRange[0])/2*multiplier
        else:
            break
        if(best<tolerance):
            gSigma = gSigmatest
            gOmega = gOmegatest
            gRho = gRhotest
            b = btest
            c = ctest
            if(nr=='r'):
                filename = filename+'Renormalized'
            f = open(filename,'w')
            f.write(str(gSigma)+" "+str(gOmega)+" "+str(gRho)+" "+str(b)+" "+str(c)+"\n")
            f.close()
            return
        print('------------------',i,'--------------------')

    print('Falure couplings')
    best=test
    gSigma = gSigmatest
    gOmega = gOmegatest
    gRho = gRhotest
    b = btest
    c = ctest
    if(nr=='r'):
        filename = filename+'Renormalized'
    f = open(filename,'w')
    f.write(str(gSigma)+" "+str(gOmega)+" "+str(gRho)+" "+str(b)+" "+str(c)+"\n")
    f.close()
    return
    return


######################################################################
#Takes a filename and returns the coupling constants in that file#
######################################################################
def readCouplingConstantsFromFile(filename,mSigma):
    numbers = np.zeros(5,np.float)
    if(mSigma==550./939):
        f = open(filename,'r')
        line = f.readline()
        numbers[:] = line.split()
        f.close()
    f = open(filename+'Renormalized','r')
    line = f.readline()
    numbersR = np.zeros(5,np.float)
    numbersR[:] = line.split()
    f.close()
    return ([numbers[0],numbersR[0]],[numbers[1],numbersR[1]],[numbers[2],numbersR[2]],
    [numbers[3],numbersR[3]],[numbers[4],numbersR[4]])


#############################################################
#Creates all vectors (Fermi momenta, effective masses etc) #
#############################################################
@jit
def createVectors(keMax,densMax,gSigma,gOmega,gRho,xSigma,xOmega,xRho,b,c,N,nr,hyp):
    if(nr=='n'):
        gSigma = gSigma[0]
        gOmega = gOmega[0]
        gRho = gRho[0]
        b = b[0]
        c = c[0]
    else:
        gSigma = gSigma[1]
        gOmega = gOmega[1]
        gRho = gRho[1]
        b = b[1]
        c = c[1]
    if(hyp==True):
        densVec = np.linspace(0,densMax,N+1)
        keVec = np.zeros(N+1)
    else:
        keVec = np.linspace(0,keMax,N+1)
        densVec = np.zeros(N+1)

    kMuVec = np.zeros(N+1)
    knVec = np.zeros(N+1)
    kpVec = np.zeros(N+1)
    klambdaVec = np.zeros(N+1)
    ksminusVec = np.zeros(N+1)
    ks0Vec = np.zeros(N+1)
    ksplusVec = np.zeros(N+1)
    kximinusVec = np.zeros(N+1)
    kxi0Vec = np.zeros(N+1)
    omegaVec = np.zeros(N+1)
    rhoVec = np.zeros(N+1)
    mStarVec = np.ones(N+1)
    mStarLambdaVec = np.ones(N+1)*mLambda
    mStarSvec = np.ones(N+1)*mS
    mStarXiVec = np.ones(N+1)*mXi
    Pvec = np.zeros(N+1)
    epsilonVec = np.zeros(N+1)
    nuclearEpsilon = np.zeros(N+1)
    nuclearP = np.zeros(N+1)
    nuclearMstar = np.ones(N+1)
    counter = 0

    knVec[0] = (densVec[0]*3*math.pi**2)**(1./3)
    for i in range(1,N+1):
        if(hyp==True):
            guesses = [omegaVec[i-1],rhoVec[i-1],keVec[i-1],knVec[i-1],mStarVec[i-1]]
            omegaVec[i],rhoVec[i],keVec[i],kMuVec[i],knVec[i],kpVec[i],klambdaVec[i],
    ksminusVec[i],ks0Vec[i],ksplusVec[i],kximinusVec[i],kxi0Vec[i],mStarVec[i],
            mStarLambdaVec[i],mStarSvec[i],mStarXiVec[i],sol = fermiMomentaAndEffectiveMass(densVec[i],guesses,gSigma,gOmega,gRho,xSigma,xOmega,xRho,b,c,nr,hyp)
            kFvec = [keVec[i],kMuVec[i],knVec[i],kpVec[i],klambdaVec[i],ksminusVec[i],ks0Vec[i],ksplusVec[i],kximinusVec[i],kxi0Vec[i]]
        else:
            guesses = [omegaVec[i-1],rhoVec[i-1],knVec[i-1],mStarVec[i-1]]
            omegaVec[i],rhoVec[i],keVec[i],kMuVec[i],knVec[i],kpVec[i],klambdaVec[i],ksminusVec[i],ks0Vec[i],ksplusVec[i],kximinusVec[i],kxi0Vec[i],mStarVec[i],mStarLambdaVec[i],mStarSvec[i],mStarXiVec[i],sol = fermiMomentaAndEffectiveMass(keVec[i],guesses,gSigma,gOmega,gRho,xSigma,xOmega,xRho,b,c,nr,hyp)
            kFvec = [keVec[i],kMuVec[i],knVec[i],kpVec[i],klambdaVec[i],ksminusVec[i],ks0Vec[i],ksplusVec[i],kximinusVec[i],kxi0Vec[i]]
            dens = density(keVec[i]) + density(kMuVec[i]) + density(knVec[i])
            dens = dens + density(kpVec[i])+density(klambdaVec[i]) + density(ksminusVec[i])
            dens = dens + density(ks0Vec[i])+density(ksplusVec[i]) + density(kximinusVec[i]) + density(kxi0Vec[i])
            densVec[i] = dens
        if(sol==True):
                counter += 1

        nucM,nuce,nucP = nuclearMatterEoS(keVec[i],gSigma,gOmega,gRho,b,c,nr,hyp)
        nuclearMstar[i] = nucM
        nuclearEpsilon[i] = nuce
        nuclearP[i] = nucP
        Pvec[i] = pressure(kFvec,mStarVec[i],mStarLambdaVec[i],mStarSvec[i],mStarXiVec[i],gSigma,gOmega,gRho,b,c,xRho,nr,hyp)
        epsilonVec[i] = energyDensity(kFvec,mStarVec[i],mStarLambdaVec[i],mStarSvec[i],mStarXiVec[i],gSigma,gOmega,gRho,b,c,xRho,nr,hyp)
    print('FermiMomentaAndEffectiveMass converged',counter,'out of',N,'times')
    kFvec = [keVec,kMuVec,knVec,kpVec,klambdaVec,ksminusVec,ks0Vec,ksplusVec,kximinusVec,kxi0Vec]
    mStarVec = [mStarVec,mStarLambdaVec,mStarSvec,mStarXiVec]
    nuclearVec = [nuclearMstar,nuclearEpsilon,nuclearP]
    return kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclearVec





################################
###                          ###
###   Mass-radii relations   ###
###                          ###
################################

###########################################################
#The derivative of the mass M with respect to the radius r#
#for arbitrary relativity                                 #
###########################################################
@jit
def dMdr(r,epsilon):
    return beta*epsilon*r**2

###############################################################
#The derivative of the pressure P with respect to the radius r#
#for arbitrary relativity                                     #
###############################################################
@jit
def dPdrTOV(r,P,M,epsilon):
    return-R0*r**(-2.)*(epsilon+P)*(M+beta*P*r**3)*(1.-2.*R0*M*r**(-1))**(-1)


###################################################################
#Function that creates the coefficients in the Runge Kutta routine#
###################################################################
```

```
@jit
def k(r,P,M,epsilon,h):
    kP1 = dPdrTOV(r,P,M,epsilon)
    kM1 = dMdr(r,epsilon)
    kP2 = dPdrTOV(r+h/2,P+h/2*kP1,M+h/2*kM1,epsilon)
    kM2 = dMdr(r+h/2,epsilon)
    kP3 = dPdrTOV(r+h/2,P+h/2*kP2,M+h/2*kM2,epsilon)
    kM3 = dMdr(r+h/2,epsilon)
    kP4 = dPdrTOV(r+h/2,P+h*kP3,M+h*kM3,epsilon)
    kM4 = dMdr(r+h,epsilon)
    return kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4


#############################################################
#Function that returns 2 vectors containing the mass-radius #
#relation and a flag indicating if the maximum number of    #
#iterations is reached                                      #
#############################################################
@jit
def results(Pc,h,nMax,EoS,PvecMin):
    P = Pc
    M = 0.
    r = 0.
    flag=True
    for i in range(nMax):
        if(flag==True):
            Memoryr = r
            r = r+h
            MemoryM = M
            if(P<PvecMin):
                epsilon=0.
                print(P)
            else:
                epsilon = EoS(P)
            kP1,kP2,kP3,kP4,kM1,kM2,kM3,kM4 = k(r,P,M,epsilon,h)
            P = P+h/6*(kP1+2*kP2+2*kP3+kP4)
            M = M+h/6*(kM1+2*kM2+2*kM3+kM4)
            if(np.float(np.real(P))<=0. or M==MemoryM):
                r = Memoryr
                flag = False
                break

    if(flag == True):
            print("Maximum number of iterations reached")
            print("for TOV-equation with Pc = %.8f"%Pc)

    return r, np.float(np.real(M)),flag


###############################################################
#Does the same as parametrisingNonRel for arbitrary relativity#
###############################################################
@jit
def paramterising(parameters,epsilonVec,Pvec):
    PcMmax = 0.
    PcMin = parameters[0]
    PcMax = parameters[1]
    N1 = int(parameters[2])
    h = parameters[3]
    nMax = int(parameters[4])
    N2 = len(Pvec)
    if(PcMax>Pvec[N2-1]):
        print("PcMax changed from ",PcMax," to ", Pvec[N2-1])
        PcMax = Pvec[N2-1]
    PvecMin = min(Pvec)
    Pc = PcMin
    const = (np.float(PcMax)/PcMin)**(1./N1)
    R = np.zeros(N1,np.double)
    M = np.zeros(N1,np.double)

    EoS = interpolate.interp1d(Pvec,epsilonVec)
    mMax = 0.
    for i in range(N1):
        print(i)
        R[i],M[i],flag = results(Pc,h,nMax,EoS,PvecMin)
        if(M[i]>mMax):
            PcMmax = Pc
        if(flag==True):
            print(R[i])
            R = R[0:i]
            M = M[0:i]
            N1 = i
            break
        Pc = Pc*const
    print("Pressure in the center of maximum mass:",PcMmax) #Print the pressure in the star with largest mass
    return R,M,N1

######################################
#Writes mass-radius relation to file#
######################################
def writeResultsToFile(parameters,epsilonVec,Pvec,filename,nr,hyp):
    if(nr=='r'):
        filename = filename+'Renormalized'
    elif(hyp==True):
        filename = filename+'Hyperon'
    R2,M2,N = paramterising(parameters,epsilonVec,Pvec)

    f = open(filename,'w')
    f.write(str(N)+'\n')
    for i in range(N):
        a = str(R2[i])
        b = str(M2[i])
        f.write(a+" "+b+"\n")
    f.close()
    return

#########################################
#Writes all data used in theses to file#
#########################################
def writeAllThingsToFile(keMax,densMax,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,filenameCouplings,filenameEoS,filenameMassRadii,mSigma,N,N2):
    if(mSigma==550./939):
        N3=2
    else:
        N3=1
    for i in range(N3):
        if(i==1):
            nr = 'n'
        else:
            nr = 'r'
        nr='n'
        writeCouplingConstantsToFile(satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,N,10,1./N,0.05,nr,False,filenameCouplings)
    gSigma,gOmega,gRho,b,c= readCouplingConstantsFromFile(filenameCouplings,mSigma)
    if(N3==2):
        N3=3
    for i in range(N3):
        if(i==2):
            nr = 'n'
            hyp = False
        elif(i==0):
            nr = 'r'
            hyp = False
        else:
            nr = 'n'
            hyp = True
        writeEoStoFile(keMax,densMax,gSigma,gOmega,gRho,b,c,N2,filenameEoS,nr,hyp)
        kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenameEoS,nr,hyp)
        writeResultsToFile(parameters,epsilonVec,Pvec,filenameMassRadii,nr,hyp)


########################################
#Reads mass-radius relations from file#
########################################
def readResultsFromFile(filename,nr,hyp):
    if(nr=='r'):
        filename = filename+'Renormalized'
    elif(hyp==True):
        filename = filename+'Hyperon'
    print(filename)
    f = open(filename,'r')
    N = int(f.readline())
    R = np.zeros(N,np.double)
    M = np.zeros(N,np.double)
    i = 0
    data = f.readlines()
    for line in data:
        numbers = line.split()
        R[i] = numbers[0]
        M[i] = numbers[1]
        i = i+1
    f.close()
    return R,M,N

########################
#Writes the EoS to file#
########################
def writeEoStoFile(keMax,densMax,gSigma,gOmega,gRho,b,c,N,filename,nr,hyp):
    if(nr=='r'):
        filename=filename+'Renormalized'
    elif(hyp==True):
        filename=filename+'Hyperon'
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear = createVectors(keMax,densMax,gSigma,gOmega,gRho,xSigma,xOmega,xRho,b,c,N,nr,hyp)
    ke = kFvec[0]
    kMu = kFvec[1]
    kn = kFvec[2]
    kp = kFvec[3]
    klambda = kFvec[4]
    ksminus = kFvec[5]
    ks0 = kFvec[6]
    ksplus = kFvec[7]
    kximinus = kFvec[8]
    kxi0 = kFvec[9]
    nucMstar = nuclear[0]
    nucEpsilon = nuclear[1]
    nucP = nuclear[2]
    mStar = mStarVec[0]
    mStarLambda = mStarVec[1]
    mStarS = mStarVec[2]
    mStarXi = mStarVec[3]
    f = open(filename,'w')
    f.write(str(N)+'\n')
    for i in range(N):
        a = str(ke[i])
        b = str(kMu[i])
        c = str(kn[i])
        d = str(kp[i])
        e = str(klambda[i])
        g = str(ksminus[i])
        h = str(ks0[i])
        j = str(ksplus[i])
        k = str(kximinus[i])
        l = str(kxi0[i])
        m = str(densVec[i])
        n = str(mStar[i])
        o = str(mStarLambda[i])
        p = str(mStarS[i])
        q = str(mStarXi[i])
        r = str(epsilonVec[i])
        s = str(Pvec[i])
        t = str(nucMstar[i])
        u = str(nucEpsilon[i])
        v = str(nucP[i])

        f.write(a+" "+b+" "+c+" "+d+" "+e+" "+g+" "+h+" "+j+" "+k+" "+l+" "+m+" ")
        f.write(n+" "+o+" "+p+" "+q+" "+r+" "+s+" "+t+" "+u+" "+v+"\n")
    f.close()
    return

#########################
#Reads the EoS from file#
#########################
def readEoSfromFile(filename,nr,hyp):
    if(nr=='r'):
        filename=filename+'Renormalized'
    elif(hyp==True):
        filename = filename+'Hyperon'
    f = open(filename,'r')
    N = int(f.readline())
    ke = np.zeros(N)
    kMu = np.zeros(N)
    kn = np.zeros(N)
    kp = np.zeros(N)
    klambda = np.zeros(N)
    ksminus = np.zeros(N)
    ks0 = np.zeros(N)
    ksplus = np.zeros(N)
    kximinus = np.zeros(N)
    kxi0 = np.zeros(N)
    densVec = np.zeros(N)
    mStarVec = np.zeros(N)
    mStarLambdaVec = np.zeros(N)
    mStarSvec = np.zeros(N)
    mStarXiVec = np.zeros(N)
    epsilonVec = np.zeros(N)
    Pvec = np.zeros(N)
    nucMstar = np.zeros(N)
    nucEpsilon = np.zeros(N)
    nucP = np.zeros(N)
    i = 0
    data = f.readlines()
    for line in data:
        numbers = line.split()
        ke[i] = numbers[0]
        kMu[i] = numbers[1]
        kn[i] = numbers[2]
        kp[i] = numbers[3]
        klambda[i] = numbers[4]
        ksminus[i] = numbers[5]
        ks0[i] = numbers[6]
        ksplus[i] = numbers[7]
        kximinus[i] = numbers[8]
        kxi0[i] = numbers[9]
        densVec[i] = numbers[10]
        mStarVec[i] = numbers[11]
        mStarLambdaVec[i] = numbers[12]
        mStarSvec[i] = numbers[13]
        mStarXiVec[i] = numbers[14]
        epsilonVec[i] = numbers[15]
        Pvec[i] = numbers[16]
        nucMstar[i] = numbers[17]
        nucEpsilon[i] = numbers[18]
        nucP[i] = numbers[19]
        i = i+1
    return [ke,kMu,kn,kp,klambda,ksminus,ks0,ksplus,kximinus,kxi0],densVec,[mStarVec,mStarLambdaVec,mStarSvec,mStarXiVec],epsilonVec,Pvec,[nucMstar,nucEpsilon,nucP],N

#################################
#Plots all mass radius relations#
#################################
def plotMassRadiusRelation(filenamesMassRadii):
    plt.figure()
    ax = plt.gca()
    ax.xaxis.set_label_coords(1.05, -0.02)
    ax.yaxis.set_label_coords(-0.05, 1)
    ax.set_xlabel('$R$[km]',fontsize = 13)
    ax.set_ylabel('$M/M_\odot$', rotation='horizontal', fontsize = 13)
    colours = ['b','y','r','g','m']
    for i in range(3):
        mSigma = 500 + 50*i
        if(i==0):
            R,M,N = readResultsFromFile(filenamesMassRadii[1],'n',False)
            maxIndex = list(M).index(max(list(M)))
            print("npmue-matter: Maximum mass is",M[maxIndex],"solar masses with radius", R[maxIndex], "km\n")
            plt.plot(R[0:maxIndex+1],M[0:maxIndex+1],colours[0],label = 'MFA')
            plt.plot(R[maxIndex:N],M[maxIndex:N],colours[0]+'--')
            R,M,N = readResultsFromFile(filenamesMassRadii[1],'n',True)
            maxIndex = list(M).index(max(list(M)))
            print("hyperon-matter: Maximum mass is",M[maxIndex],"solar masses with radius", R[maxIndex], "km\n")
            plt.plot(R[0:maxIndex+1],M[0:maxIndex+1],colours[4],label = 'MFA with Hyperons')
            plt.plot(R[maxIndex:N],M[maxIndex:N],colours[4]+'--')

        R,M,N = readResultsFromFile(filenamesMassRadii[i],'r',False)
        maxIndex = list(M).index(max(list(M)))
        print("Renormalized npmue-matter, mSigma =",mSigma,": Maximum mass is",M[maxIndex],"solar masses with radius", R[maxIndex], "km\n")

        plt.plot(R[0:maxIndex+1],M[0:maxIndex+1],colours[i+1],label = 'RHA with $m_\sigma=$'+str(mSigma))
        plt.plot(R[maxIndex:N],M[maxIndex:N],colours[i+1]+'--')
    plt.legend()

######################################
#Plots the equation of state from file#
######################################
def plotEoS(filenameEoS):
    plt.figure()
    plt.xlim(0,0.02)
    plt.ylim(0,0.01)
    ax = plt.gca()
    ax.xaxis.set_label_coords(1., -0.07)
    ax.set_xlabel('$\\bar{\epsilon}$', fontsize = 15)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\bar{P}(\\bar{\epsilon})$', rotation='horizontal',fontsize = 15)
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'n',False)
    plt.plot(epsilonVec,Pvec,'b',label='MFA')
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'r',False)
    plt.plot(epsilonVec,Pvec,'r',label='RHA')
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'n',True)
    plt.plot(epsilonVec,Pvec,'m',label='MFA with hyperons')
    plt.legend()


#####################################################################
#Plots the population density from using the file containing the EoS#
#####################################################################
def plotPopulationDensity(filenameEoS):
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'n',False)
    plt.figure()
    ax = plt.gca()
    plt.xlim(0,1.6)
    plt.ylim(0.001,1.01)
    ax.text(0.84,0.66,'n',fontsize=14)
    ax.text(0.60,0.19,'p',fontsize=14)
    ax.text(0.57,0.105,'e',fontsize=14)
    ax.text(0.12,0.0066,'$\mu$',fontsize=14)

    ax.xaxis.set_label_coords(1., -0.05)
    ax.set_xlabel('$\\rho$[fm$^{-3}$]', fontsize = 15)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\rho_i/\\rho$', rotation='horizontal',fontsize = 15)

    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],density(kFvec[0])[1:N2+1]/densVec[1:N2+1],'k',label='e')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[1])[1:N2+1]/densVec[1:N2+1]),'r',label='$\mu$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[2])[1:N2+1]/densVec[1:N2+1]),'b',label='n')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[3])[1:N2+1]/densVec[1:N2+1]),'g',label='p')
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'n',True)


    plt.figure()
    ax = plt.gca()
    plt.xlim(0,1.6)
    plt.ylim(0.001,1.01)
    ax.text(0.616,0.55,'n',fontsize=14)
    ax.text(0.596,0.25,'p',fontsize=14)
    ax.text(0.34,0.10,'e',fontsize=14)
    ax.text(0.10,0.0066,'$\mu$',fontsize=14)
    ax.text(0.24,0.0013,'$\Sigma^-$',fontsize=14)
    ax.text(0.36,0.007,'$\Lambda$',fontsize=14)
    ax.text(0.586,0.0013,'$\Xi^-$',fontsize=14)
    ax.text(0.83,0.007,'$\Sigma^0$',fontsize=14)
    ax.text(0.92,0.0013,'$\Sigma^+$',fontsize=14)
    ax.text(1.14,0.007,'$\Xi^0$',fontsize=14)

    ax.xaxis.set_label_coords(1., -0.05)
    ax.set_xlabel('$\\rho$[fm$^{-3}$]', fontsize = 15)
    ax.yaxis.set_label_coords(-0.05, 1.03)
    ax.set_ylabel('$\\rho_i/\\rho$', rotation='horizontal',fontsize = 15)

    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],density(kFvec[0])[1:N2+1]/densVec[1:
```

```
N2+1],'k',label='e')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[1])[1:N2+1]/densVec[1:N2+1]),'r',label='$\mu$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[2])[1:N2+1]/densVec[1:N2+1]),'b',label='n')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[3])[1:N2+1]/densVec[1:N2+1]),'g',label='p')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[4])[1:N2+1]/densVec[1:N2+1]),'k—',label='$\Lambda$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[5])[1:N2+1]/densVec[1:N2+1]),'r—',label='$\Sigma^-$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[6])[1:N2+1]/densVec[1:N2+1]),'b—',label='$\Sigma^0$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[7])[1:N2+1]/densVec[1:N2+1]),'g—',label='$\Sigma^+$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[8])[1:N2+1]/densVec[1:N2+1]),'y—',label='$\Xi^-$')
    plt.semilogy(MeVtoFm(densVec*939**3,3)[1:N2+1],(density(kFvec[9])[1:N2+1]/densVec[1:N2+1]),'m—',label='$\Xi^0$')

###########################
#Plots the binding energy#
###########################
def plotBindingEnergy(kFvec,epsilonVec,nuclearEpsilonVec):
    dens = density(kFvec[2])+density(kFvec[3])
    nuclearDens = 2*density(kFvec[0])
    N = len(epsilonVec)
    B = epsilonVec[1:N]/dens[1:N]-m
    nuclearB = nuclearEpsilonVec[1:N]/nuclearDens[1:N]-m
    plt.figure()
    plt.xlim(0,1.5)
    plt.ylim(-50,300)
    plt.plot(MeVtoFm(dens[1:N]*939**3,3),B*939,'b')
    plt.plot(MeVtoFm(nuclearDens[1:N]*939**3,3),nuclearB*939,'r')
    return

#####################################################################
#Takes in the filenames of all relevant files and plots everything#
#####################################################################
def plotEverything(filenamesEoS,filenamesMassRadii):
    kFvec,densVec,mStarVec,epsilonVec,Pvec,nuclear,N2 = readEoSfromFile(filenamesEoS[1],'nr',False)
    plotBindingEnergy(kFvec,epsilonVec,nuclear[1])
    plotEoS(filenamesEoS[1])
    plotPopulationDensity(filenamesEoS[1])
    plotMassRadiusRelation(filenamesMassRadii)


##########################
###                    ###
###   Program start    ###
###                    ###
##########################

#Some spesifications:

#The variable 'hyp' is bool. If hyp==True then calculations include hyperons
#The variable 'nr' is string. If nr=='n' we use MFA, while nr='r' uses RHA

############
#Parameters#
############

#particle masses
m = 1.
me = 0.51099/939
```

```python
mMu = 105.7/939.
mSigma = 550./939
mOmega = 783./939
mRho = 775.5/939
mLambda = 1115./939
mS = 1192./939
mXi = 1318./939

#Other parameters
R0 = 1.47
beta = 1.1426

#Nuclear matter properties
landauMass = 0.83
satDens = fmToMeV(0.153,-3)/939**3
satMstar =  np.sqrt(landauMass**2-((3/2*math.pi**2*satDens)**(1./3))**2)
satB = -16.3/939
satK = 300./939
satAs = 32.5/939

#Variables used when computing massradius relations
PcMin = 4*10**(-6)      #Minimum central pressure
PcMax = 1.2*10**(0)     #Maximum central pressure
N1 = 300                #Number of data points in mass-radius plot
h = 0.001               #Step size in km
nMax = 100/h            #Maximum number of iterations

#Collects the mass-radius parameters in one vector
parameters = [PcMin,PcMax,N1,h,nMax]

N = 8#Number of intervalls used to compute the coupling constants

N2 = 50000          #Number of data points used for the EoS
keMax = 0.5         #Maximum electron Fermi momentum
densMax = 0.035     #Maximum baryon density

#Coupling ratios: [xgLambda,xgSigma,xgXi]
xSigma = [1,1,1]
xOmega = [1,1,1]
xRho = [1,1,1]

#Filenames
filenamesCouplings = ['couplingconstants500','couplingconstants550','couplingconstants600']
filenamesEoS = ['equationOfState500','equationOfState550','equationOfState600']
filenamesMassRadii = ['massRadiusRelation500','massRadiusRelation550','massRadiusRelation600']

#The ranges we look for the coupling constants
gSigmaRange =  [6,7]
gOmegaRange =  [8,9]
bRange = [-0.01,0]
cRange = [-0.01,0]
mSigma = 500./939
writeAllThingsToFile(keMax,densMax,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,filenamesCouplings[0],filenamesEoS[0],filenamesMassRadii[0],mSigma,N,N2)
print(1,'out of',3,'complete!')
gSigmaRange =  [7,9]
gOmegaRange =  [8,9]
bRange = [-0.01,0]
cRange = [-0.01,0]
mSigma = 550./939
writeAllThingsToFile(keMax,densMax,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,filenamesCouplings[1],filenamesEoS[1],filenamesMassRadii[1],mSigma,N,N2)
```

```
print(2,'out of',3,'complete! Nearly there...')
gSigmaRange =  [9,10]
gOmegaRange =  [8,9]
bRange = [-0.01,0]
cRange = [-0.01,0]
mSigma = 600./939
writeAllThingsToFile(keMax,densMax,satDens,satMstar,satB,satK,satAs,gSigmaRange,gOmegaRange,bRange,cRange,filenamesCouplings[2],filenamesEoS[2],filenamesMassRadii[2],mSigma,N,N2)
print('Done! Just plotting now...')

plotEverything(filenamesEoS,filenamesMassRadii)


plt.show()



print("\nTime spent:")
print(time.clock()-t0)
```

Bibliography

[1] D. G. Yakovlev et al. “Lev Landau and the concept of neutron stars”. In: *Phys. Usp.* 56 (2013). [Usp. Fiz. Nauk183,307(2013)], pp. 289–295. DOI: 10.3367/UFNe.0183.201303f.0307. arXiv: 1210.0682 [physics.hist-ph].

[2] I. Vidana. “Hyperons: the strange ingredients of the nuclear equation of state”. In: (2018). arXiv: 1803.00504 [nucl-th]. URL: http://inspirehep.net/record/1658062/files/1803.00504.pdf.

[3] I. Bombaci. “The Hyperon Puzzle in Neutron Stars”. In: *JPS Conf. Proc.* 17 (2017), p. 101002. DOI: 10.7566/JPSCP.17.101002. arXiv: 1601.05339 [nucl-th].

[4] M G. Alford. “Quark matter in neutron stars”. In: *Nucl. Phys.* A830 (2009), pp. 385C–392C. DOI: 10.1016/j.nuclphysa.2009.09.034. arXiv: 0907.0200 [nucl-th].

[5] N. Rea. “Magnetars: neutron stars with huge magnetic storms”. In: *IAU Symp.* 291 (2013), pp. 11–18. DOI: 10.1017/S1743921312023058. arXiv: 1211.2086 [astro-ph.HE].

[6] N. K. Glendenning. *Compact Stars.* Springer, Apr. 2000. ISBN: 0-387-98977-3.

[7] J. Antoniadis et al. “A Massive Pulsar in a Compact Relativistic Binary”. In: *Science* 340.6131 (2013). ISSN: 0036-8075. DOI: 10.1126/science.1233232. eprint: http://science.sciencemag.org/content/340/6131/1233232.full.pdf. URL: http://science.sciencemag.org/content/340/6131/1233232.

[8] L. Lindblom. “Determining the nuclear equation of state from neutron-star masses and radii”. In: *The Astrophysical Journal* 398 (Oct. 1992), pp. 569–573. DOI: 10.1086/171882.

[9] M. Ferraris, M. Francaviglia, and C. Reina. “Variational formulation of general relativity from 1915 to 1925 “Palatini’s method” discovered by Einstein in 1925”. In: *General Relativity and Gravitation* 14.3 (Mar. 1982), pp. 243–254. ISSN: 1572-9532. DOI: 10.1007/BF00756060. URL: https://doi.org/10.1007/BF00756060.

[10] M. Kachelrieß. *Lecture Notes for FY3452 Gravitation and Cosmology.* URL: http://web.phys.ntnu.no/~mika/skript_grav.pdf. Online; accessed April 2018. 2015.

[11] J. R. Magnus. *Matrix differntial calculus with applications in statistics and economics.* John Wiley & Sons, 1995. ISBN: 0-471-98632-1.

[12] A.-H. Pedersen. *Linearized general relativity and its quantization.* eng. 2017. URL: http://hdl.handle.net/11250/2445854.

[13] R. F. Polishchuk. “Post-Einstein cosmology: Torsion, strings, dual symmetry. I. The necessity of using the Einstein-Cartan theory instead of the Einstein theory”. In: *Bulletin of the Lebedev Physics Institute* 40.1 (Jan. 2013), pp. 12–16. ISSN: 1934-838X. DOI: 10.3103/S106833561301003X. URL: https://doi.org/10.3103/S106833561301003X.

[14] R. Aldrovandi and J. G. Pereira. “Teleparallelism: A New Way to Think the Gravitational Interaction”. In: *Ciencia Hoje* 55 (2015), p. 32. arXiv: `1506.03654 [physics.pop-ph]`.

[15] L. Combi and G. E. Romero. “Is Teleparallel Gravity really equivalent to General Relativity?” In: *Annalen Phys.* 530.1 (2018), p. 1700175. DOI: `10.1002/andp.201700175`. arXiv: `1708.04569 [gr-qc]`.

[16] J. Myrheim. *Classical Theory of Fields.* URL: `https://dvikan.no/ntnu-studentserver/kompendier/JanMyrheim-Kompendium.Klassisk.FeltTeori.pdf`. Online; accessed August 2017.

[17] S. W. Hawking and G. F. R. Ellis. *The Large Scale Structure of Space-Time.* Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1973. ISBN: 9780521200165. URL: `http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=502384&site=ehost-live`.

[18] P. G. Frè. *Gravity, a Geometrical Course : Volume 1: Development of the Theory and Basic Physical Applications.* eng. 2012. ISBN: 1-283-90990-1.

[19] J. R. Oppenheimer and G. M. Volkoff. “On massive neutron cores”. In: *Physical Review* 55.4 (1939), pp. 374–381. ISSN: 0031899X.

[20] R. C. Tolman. “Static Solutions of Einstein’s Field Equations for Spheres of Fluid”. In: *Phys. Rev.* 55 (4 Feb. 1939), pp. 364–373. DOI: `10.1103/PhysRev.55.364`. URL: `https://link.aps.org/doi/10.1103/PhysRev.55.364`.

[21] R. R. Silbar and S. Reddy. “Neutron stars for undergraduates.(Author Abstract)”. English. In: *American Journal of Physics* 72.7 (July 2004). ISSN: 0002-9505.

[22] L. Kyllingstad. *Pion condensation in effective field theories.* eng. 2007. URL: `http://web.phys.ntnu.no/~mika/kyllingstad.pdf`.

[23] L. Álvarez-Gaumé and M. Á. Vázquez-Mozo. “Why Do We Need Quantum Field Theory After All?” In: *An Invitation to Quantum Field Theory.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–9. ISBN: 978-3-642-23728-7. DOI: `10.1007/978-3-642-23728-7_1`. URL: `https://doi.org/10.1007/978-3-642-23728-7_1`.

[24] M. E. Peskin. *An introduction to quantum field theory.* eng. Reading, Mass: Addison-Wesley, 1995. ISBN: 0201503972.

[25] J. I. Kapusta and C. Gale. *Finite-temperature field theory : principles and applications.* eng. 2nd ed. Cambridge monographs on mathematical physics. Cambridge: Cambridge University Press, 2006. ISBN: 9780521820820.

[26] J. O. Andersen. *Introduction to statistical mechanics.* eng. Trondheim: Akademika forl, 2012. ISBN: 9788232101054.

[27] A. Kumar. “Matsubara Frequency Sums”. In: (Feb. 2010).

[28] S. L. Shapiro and S. A. Teukolsky. *Black Holes, White Dwarfs, and Neutron Stars. The physics of compact objects.* Vol. 1. The address: Wiley, 1983. ISBN: 0-471-87316-0.

[29] M G. Alford, Steven P. Harris, and Pratik S. Sachdeva. “On the stability of strange dwarf hybrid stars”. eng. In: *The Astrophysical Journal* 847.2 (Oct. 2017). ISSN: 0004-637X.

[30] J. M. Bardeen, K. S. Thorne, and D. W. Meltzer. “A Catalogue of Methods for Studying the Normal Modes of Radial Pulsation of General-Relativistic Stellar Models”. In: *Astrophysical Journal* 145 (Apr. 1966). An optional note, pp. 505–513.

[31] M. H. Johnson and E. Teller. “Classical Field Theory of Nuclear Forces”. In: *Phys. Rev.* 98 (3 May 1955), pp. 783–787. DOI: `10.1103/PhysRev.98.783`. URL: `https://link.aps.org/doi/10.1103/PhysRev.98.783`.

[32] H.-P. Duerr. “Relativistic Effects in Nuclear Forces”. In: *Phys. Rev.* 103 (1956), pp. 469–480. DOI: `10.1103/PhysRev.103.469`.

[33] S. A. Chin and J. D. Walecka. “An equation of state for nuclear and higher-density matter based on relativistic mean-field theory”. eng. In: *Physics Letters B* 52.1 (1974), pp. 24–28. ISSN: 0370-2693.

[34] N. K. Glendenning. “Vacuum polarization effects in the non-linear σ, ω-model”. eng. In: *Physics Letters B* 208.3 (1988), pp. 335–338. ISSN: 0370-2693.

[35] J. C. Maxwell. “On the dynamical evidence of the molecular constitution of bodies”. In: *Nature* 11.279 (1875), pp. 357–359. ISSN: 00280836.

[36] G. G. Emch. *The logic of thermostatistical physics*. eng. Berlin, 2002.

[37] D. T. Khoa, G. R. Satchler, and W. Von Oertzen. “Nuclear incompressibility and density dependent NN interactions in the folding model for nucleus-nucleus potentials”. eng. In: *Physical Review C* 56.2 (Aug. 1997), pp. 954–969. ISSN: 0556-2813.

[38] J. P. Blaizot. “Nuclear compressibilities”. eng. In: *Physics Reports* 64.4 (1980), pp. 171–248. ISSN: 0370-1573.

[39] D. H. Youngblood, H. L. Clark, and Y. Lui. “Incompressibility of Nuclear Matter from the Giant Monopole Resonance”. English. In: *Physical Review Letters* 82.4 (1999), pp. 691–694. ISSN: 0031-9007.

[40] R. Nayak, V. S. Uma Maheswari, and L. Satpathy. “Saturation properties and incompressibility of nuclear matter: A consistent determination from nuclear masses”. In: *Phys. Rev.* C52 (1995), pp. 711–717. DOI: 10.1103/PhysRevC.52.711. arXiv: nucl-th/9506032 [nucl-th].

[41] R. Machleidt, K. Holinde, and Ch. Elster. “The bonn meson-exchange model for the nucleon—nucleon interaction”. In: *Physics Reports* 149.1 (1987), pp. 1–89. ISSN: 0370-1573. DOI: https://doi.org/10.1016/S0370-1573(87)80002-9. URL: http://www.sciencedirect.com/science/article/pii/S0370157387800029.

[42] J. Boguta and A. R. Bodmer. “Relativistic calculation of nuclear matter and the nuclear surface”. eng. In: *Nuclear Physics, Section A* 292.3 (1977), pp. 413–428. ISSN: 0375-9474.

[43] M. Baldo and G. F. Burgio. “The nuclear symmetry energy”. In: *Prog. Part. Nucl. Phys.* 91 (2016), pp. 203–258. DOI: 10.1016/j.ppnp.2016.06.006. arXiv: 1606.08838 [nucl-th].

[44] N. K. Glendenning. “Vacuum polarization effects on nuclear matter and neutron stars”. In: *Nuclear Physics A* 493.3 (1989), pp. 521–548. ISSN: 0375-9474. DOI: https://doi.org/10.1016/0375-9474(89)90101-2. URL: http://www.sciencedirect.com/science/article/pii/0375947489901012.

[45] T. Padmanabhan. “Why Does Gravity Ignore the Vacuum Energy?” In: *Int. J. Mod. Phys.* D15 (2006), pp. 2029–2058. DOI: 10.1142/S0218271806009455. arXiv: gr-qc/0609012 [gr-qc].

[46] S. Wen. *n-dimension Spherical coordinates and the volumes of the n-ball in* $\mathbb{R}^n$. URL: http://www.ams.sunysb.edu/~wshih/mathnotes/n-D_Spherical_coordinates.pdf. Online; accessed April 2018.

[47] X.-F. Zhao. “Can the massive neutron star PSR J0348+0432 be a hyperon star?” In: *Acta Phys. Polon.* B48 (2017), p. 171. DOI: 10.5506/APhysPolB.48.171. arXiv: 1712.08870 [nucl-th].

[48] Jürgen Schaffner and Igor N. Mishustin. “Hyperon-rich matter in neutron stars”. In: *Phys. Rev. C* 53 (3 Mar. 1996), pp. 1416–1429. DOI: 10.1103/PhysRevC.53.1416. URL: https://link.aps.org/doi/10.1103/PhysRevC.53.1416.

[49] N. K. Glendenning. “Neutron Stars Are Giant Hypernuclei?” In: *Astrophys. J.* 293 (1985), pp. 470–493. DOI: 10.1086/163253.

[50] X. Mu et al. “Effects of the σ^* and ϕ Mesons on the Properties of Massive Protoneutron Stars”. In: *Astrophys. J.* 846.2 (2017), p. 140. DOI: 10.3847/1538-4357/aa880c.

[51] K. A. Maslov, E. E. Kolomeitsev, and D. N. Voskresensky. “Solution of the Hyperon Puzzle within a Relativistic Mean-Field Model”. In: *Phys. Lett.* B748 (2015), pp. 369–375. DOI: 10.1016/j.physletb.2015.07.032. arXiv: 1504.02915 [astro-ph.HE].

[52] P. Wang, A. W. Thomas, and A. G. Williams. “Phase transition from hadronic matter to quark matter”. In: *Phys. Rev.* C75 (2007), p. 045202. DOI: 10.1103/PhysRevC.75.045202. arXiv: nucl-th/0610084 [nucl-th].

[53] Z. Fodor and S. D. Katz. “Critical point of QCD at finite T and mu, lattice results for physical quark masses”. In: *JHEP* 04 (2004), p. 050. DOI: 10.1088/1126-6708/2004/04/050. arXiv: hep-lat/0402006 [hep-lat].

[54] K. Konno, T. Obata, and Y. Kojima. “Flattening modulus of a neutron star by rotation and magnetic field”. In: *Astron. Astrophys.* 356 (2000), pp. 234–237. arXiv: astro-ph/0001397 [astro-ph].

[55] S. Balberg, I. Lichtenstadt, and G. B. Cook. “Roles of hyperons in neutron stars”. In: *Astrophys. J. Suppl.* 121 (1999), p. 515. DOI: 10.1086/313196. arXiv: astro-ph/9810361 [astro-ph].