UNIVERSITY OF OSLO
Department of Physics

A Numerical framework for simumulating the cooling of a Neutron star

Master Thesis 2007
Jon Thonstad
Fysisk Institut

5th June 2007

Preface

0.1 Acknowledgements

Who would have believed it? I finally finished my master!

I can't really say I did it all without help, so thanks to all who helped me during the last couple of years. Especially Morten Hjorth-Jensen for putting up with all my delays, and being my supervisor. Though he could have been better at kicking my butt into gear. Sutharsan for being an awesome partner and a great help. Lillian for actually kicking my butt into gear. All my other friends. My parents. And last but not least the coffee machine in the institute lobby. Couldn't have done it without you!

0.2 Abstract

In this work I will study various URCA processes relevant for neutrino emission from a neutron star, with the main focus on the direct and modified URCA processes.

$$n \to p + l + \bar{\nu}_l$$

and

$$n + n \to n + p + l + \bar{\nu}_l.$$

I will build the necessary tools and formalism to do numerical Quantum Field Theory calculations of the luminosity of a neutron star, taking different matter-effects into consideration. I will look at different equations of state ranging from nucleon-matter to hyperion-matter. The work on these equations of state has already been done by Henning Heiselberg and Morten Hjorth-Jensen. So I will use their data for the necessary parameters, such as effective masses, proton fraction, electron and muon chemical potentials.

We also see that the tools built for this task can easily be extended to study at more exotic processes such as URCA processes with hyperons and Δ Isobars

Contents

Chapter 1

Introduction

1.1 The neutron star

A neutron star is one of the possible outcomes after the gravitational collapse of the core of a massive star. The collapse triggers a supernova explosion, and it is within this explosion that the neutron star is formed. This is one of the more extreme event in the universe, and so the characteristics of a neutron star is equally extreme. A typical neutron star has a mass between 1.35 to about 2.1 solar masses ($M_\odot$), with a corresponding radius between 10 and 20 km. That means that the neutron star is 30000 to 70000 times smaller than the sun but with greater mass. This makes the central density of a neutron star n_c as high as 5 to 10 times that of nuclear equilibrium density $n_0 \simeq 0.16 fm^{-3}$, so the core of a neutron star is one of the densest forms of matter known. These characteristics is not found anywhere else in the universe. Because of these internal conditions, the neutron star offers an unique opportunity to study properties of nuclear matter in conditions not possible here on earth, such as hyperon-dominated matter, deconfined quark matter, super fluidity and superconductivity with critical temperatures near 10^{10} kelvin, opaqueness to neutrino, and magnetic fields in excess of 10^{13} Gauss.

It is unfortunately impossible to obtain detailed experimental information about the neutron star, so we have to use astrophysical observables to divulge any information about its internal workings, the main observables being the mass of the star, its radius and its surface temperature.

The structure of a neutron star and its macroscopic physics is described by the all important equation of state, from now on EoS. By using astrophysical observables we can get information about about the EoS and thereby learn more about nuclear physics.

1.2 Neutrino emission - URCA processes

The connection between the EoS and astrophysical observables of a neutron star has been studied in different articles since the concept of the neutron star was proposed in 1933. Chiu and Salpeter proposed that the main source of energy loss from a neutron star was by neutrino emission by URCA processes [4]. This showed to be correct. Other non-nucleon processes proved to be unimportant at typical neutron star densities and temperatures [5].

Standard neutron decay Eq. (1.1) is strongly suppressed in neutron stars at equilibrium by a large energy-momentum mismatch. It is therefore necessary to modify the process by using a bystander particle that lends its momentum to the process. This process is called the modified URCA process Eq. (1.2).

$$n \rightarrow p + l + \bar{\nu}_l \tag{1.1}$$

$$(n, p) + n \rightarrow (n, p) + p + l + \bar{\nu}_l \tag{1.2}$$

l is a lepton (electron, muon, tau) and (n, p) is either a neutron or a proton. Other processes of importance is neutrino bremsstrahlung from nucleon-nucleon scattering

$$n \rightarrow n + \nu_l + \bar{\nu}_l \tag{1.3}$$

and

$$n + n \rightarrow p + n + \nu_l + \bar{\nu}_l. \tag{1.4}$$

Calculations of the neutrino emissivity by URCA processes have been done by several authors, but never in a self-consistent matter. Most articles do not get all their data from the equations of state in a self-consistent matter. The main problem being the nucleon-nucleon interactions where authors have done different approximations to get results. The most resent and most cited is "Neutrino Emissivities of Neutron Stars" by Friman and Maxwell [6]. They approximate the nucleon-nucleon interaction with a long-range one-pion-exchange part and a short-range part parametrized with nuclear Fermi liquid (Landau) parameters. Other work in cooling of neutron stars use computational scheme of this article as a basis for their approximations, and study of different kinds of EoS.

1.3 My work

In this thesis I have developed numerical tools and formalism to do many-body Quantum Field Theory(QFT) simulations of the different processes in the neutron star taking matter-effects into considerations. This is done by making a program for calculating general Feynman amplitudes as functions of external particle momenta and spin. Using this program I can parameterize the nucleon-nucleon interaction using the Fermi invariant γ-matrices.

$$V = \bar{u}(\mathbf{p}_1{}')\bar{u}(\mathbf{p}_2{}') \left(\sum_{j=1}^{5} v_j(\mathbf{k}) F_j \right) u(\mathbf{p}_1)u(\mathbf{p}_2). \tag{1.5}$$

Here v_j is the parameterization values and F_j $(j = 1, ..., 5)$ denote the Fermi invariants defined as:

$$S = 1^{(1)}1^{(2)}, \quad V = \gamma_\mu^{(1)}\gamma_\mu^{(2)}, \quad T = \frac{1}{2}\sigma_{\mu\nu}^{(1)}\sigma_{\mu\nu}^{(2)}, \tag{1.6}$$

$$A = i\gamma_5^{(1)}\gamma_\mu^{(1)}i\gamma_5^{(2)}\gamma_\mu^{(2)}, \quad P = \gamma_5^{(1)}\gamma_5^{(2)}, \tag{1.7}$$

The parameterization values $v_j(\mathbf{k})$ is usually the muon propogator $\frac{i}{-\mathbf{k^2}-\mathbf{m}_\alpha}$, including the appropriate coupling constants. This parameterization can further be used to calculate the neutrino emissivity of a neutron star in a self-consistent matter. The main work of this thesis is to build a program with these capabilities and test it. Making the parameterization itself will be left for further work. The testing is done by calculating the luminosity of the direct and modified URCA-processes using a simple one-pion-exchange approximation. The results can therefore be compared to that of Friman and Maxwell and others who have done similar calculations.

The data for the different EoS in this thesis will be taken from the article "Phases of dense matter in neutron stars" [10]. I will focus on the modified URCA process while my colleague Sutharsan Arumugam will focus on neutrino bremsstrahlung from nucleon-nucleon scattering.

1.4 Structure of this thesis

The first part of this thesis will serve as an introduction to neutron star physics. Here I will build up the necessary formalism and phenomenology needed to understand the rest of this thesis. In chapter 2 I will give an introduction to neutron star physics. A brief history of neutron star observations and the development of neutron star physics, the composition of the neutron star, and will give an introductions to the different EoS. Chapter 3 explains calculational details of the cooling of the neutron star, with the focus of the direct and modified URCA processes. These chapters can be considered an introduction to neutron physics and do not contain any new exciting calculations in them, so these chapters borrow heavily from other sources, mainly the article "Phases of Dense Matter in Neutron Stars" by

Henning Heiselberg and Morten Hjorth-Jensen [10], "The Physics of Neutron Stars" by J.M. Lattimer and M. Prakash [13], and Chapters 9-11 of the book "Black Holes, white Dwarfs, and Neutron Stars" by S.L Shapiro and S. A. Teukolsky [17].

The following part of the thesis will contain a description of the software my colleague Sutharsan Arumugam and I have developed to do QFT calculations. Chapter 4 will contain a introduction to the software itself. Chapter 5 will contain the numerical solutions to the URCA processes, with the main focus on comparing the numerical results with the analytical approximations.

The last part of this thesis will contain the conclusion.

Chapter 2

The neutron star

The goal of the following chapter is to give a brief introduction to the neutron star. It contains a short history of the discovery and observations of different neutron stars, and how neutron star physics has evolved since its birth. It then goes on to explain the physics of a neutron star. Starting with the structure and going on to explain the equations of state. The equations that govern the macroscopic behaviour of the star, such as temperature, pressure and the different phases of the matter.

This chapter lends a bit from M. Hjorth-Jensen and H. Heiselbergs article, "Phases of dense matters in Neutron stars".

2.1 History

2.1.1 1932 - There might be neutron stars

The fist mention of a neutron star came from Landau in 1932 just after Chadwick [3] had discovered the neutron. He never made an article on the subject. The first article was made by Walter Blaade and Fritz Zwicky in nature in 1933. A neutron star would be a star with high density and small radius, that would be much more gravitationally bound than ordinary stars. They also made the prediction that a neutron star would be formed in supernova explosions [2]. The explosion caused when a star runs out of fuel for it fusion process and collapses under the force of its own gravity. The first theoretical calculation of a neutron star model was done by Oppenheimer and Volkoff in 1939 [15].They assumed the neutron stars to be composed of an ideal gas of neutrons at high density. The main motivation for studying neutron stars at the time was to check if it was a source of stellar energy. As thermonuclear fusion become understood and physicists discovered that neutron stars did not give a significant contribution to the stellar energy, they lost interest in the idea. Another thing that made physicist less interested was the fact that a neutron star is virtually undetectable by optical telescopes. It is cold and dense so it gives off very little thermal radiations. So physicists focused on other stellar objects for the next 30 years. There were made articles about neutron stars in that time. Papers were made by Harrison, Wakanao, and Wheeler (1958), Cameron (1959) Ambartsumyan and Saakyan (1960), and Hamada and Salpeter (1961) all of these discussing the equation of state for neutron stars. The discovery of cosmic, nonsolar X-ray sources by Giacconi [8] in 1962, sparked the interest for neutrino stars again. Neutrino stars might be the source of X-ray's so lots of physicists started to work on the cooling of neutron stars.

2.1.2 1968 - Discovery

The first “Quasi stellar object” (QSO, or quasar) was detected in 1963. Quasars had large redshifts in the spectral lines, that might come from the gravitational redshift of a neutron star. But it was soon discovered that the redshifts exceeded the max gravitational redshift form a stable neutron star. The first observation of a neutron star came with the discovery of a radio pulsar in 1968 by Bell and Hewish [11]. The pulsar was located in the remnant of the Crab supernova observed by the Chinese in 1054 A.D. and confirmed the link to supernovae.

2.2 Types of neutron stars

Radio-quiet neutron stars are almost impossible to detect. They are dense, relatively cold objects with an average mass of about $1.4 M_\odot$ and a radius of about $10 km$. So far very few radio-quiet neutron star been directly observed. Luckily there are other more exotic kinds of neutron stars that have other properties that make them easier to detect.

2.2.1 Radio Pulsars

Pulsars are rapidly rotating neutron stars with periods in the range off $0.33s \leq P \leq 4.0s$. They emit electromagnetic radiation that varies in intensity with a regular period, believed to correspond with the rotational period of the star. This corresponds to a "lighthouse effect". They are believed to be powered by rotational energy loss and are rapidly spinning down with period derivatives of order $\dot{P} = 10^{-12} - 10^{-16}$. Their high magnetic field B leads to dipole magnetic braking radiation proportional to the magnetic field squared. Estimates for the magnetic field is of the order $B \sim 10^{11} - 10^{13} G$. : A distinct subclass of radio pulsars are *millisecond pulsars* with periods between 1.56 ms$\leq P \leq$ 100 ms. The period derivatives are very small corresponding to very small magnetic fields $B \sim 10^8 - 10^{10}$G. They are believed to be recycled pulsars, i.e. old pulsars with low magnetic fields that have been spun up by accretion preserving their low magnetic field and therefore only slowly spinning down. About 20 - almost half of the millisecond pulsars - are found in binaries where the companion is either a white dwarf or a neutron star. Six double neutron stars are known so far including the Hulse-Taylor PSR 1913+16. The first binary pulsar was found by Hulse and Taylor in 1973 and by measuring the general relativistic corrections to Newtonian gravity one could determine all parameters in the binary system as both masses, orbital periods and period derivatives, orbital distances and inclination. Parameters are overdetermined and thus provides a test of general relativity. Inward spiralling or orbital decay is an additional test of general relativity to an unprecedented accuracy. The binary neutron stars all have masses in the narrow interval $1.3 - 1.5 M_\odot$, which may either be due to the creation process or that heavier neutron stars are unstable.

2.2.2 X-ray pulsars and Bursters

With X-ray detectors on board satellites since 1971 almost two hundred *X-ray pulsars and bursters* have been found of which the orbital period has been determined for about sixty. The X-ray pulsars and bursters are believed to be accreting neutrons stars from high ($M \gtrsim 10 M_\odot$) and low mass ($M \lesssim 1.2 M_\odot$) companions respectively. The X-ray pulses are most probably due to strong accretion on the magnetic poles emitting X-ray (as northern lights) with orbital frequency. The X-ray bursts are due to slow accretion spreading all over the neutron star surface before igniting in a thermonuclear flash. The resulting (irregular) bursts have periods depending on accretion rates rather than orbital periods. Recently, bursters and pulsars have been linked by observations of X-ray pulsations in bursts from several low mass X-ray bursters [21]. The pulsations are with spin frequency 300-400Hz and increase by a few Hz only during the burst. The small increase is expected from cooling after a thermonuclear explosion, which leads to smaller size and moment of inertia and, conserving angular momentum, to larger frequency.

2.2.3 Other more exotic kinds of neutron stars

The radiation from X-ray bursters is not blackbody and therefore only upper limits on temperatures can be extracted from observed luminosities in most cases. Masses are less accurately measured than for binary pulsars. A subclass of six anomalous X-ray pulsars are slowly rotating but rapidly spinning down indicating that they are young with enormous magnetic fields, $B \sim 10^{15}$G, and thus named *Magnetars* [18]. Recently, quasi-periodic oscillations (QPO) have been found in 12 low mass X-ray binaries. The QPO's set strict limits on masses and radii of neutrons stars, but if the periodic oscillations arrive from the innermost stable orbit [20], it implies definite neutron star masses up to $M \simeq 2.3 M_\odot$.

2.3 Physics of a Neutron Star

The properties of a neutron star is given by the equation of state(EoS). It is our model of the neutron star. We start out with a so-called realistic EoS to determine the overall structure of the star. This is simply a EoS that takes into account that the star is stable and has a given mass and radius and therefore a given central density n_c. This puts certain limits on a proper EoS. To make a good model we need to analyze the neutron star at different densities around the central density and map the relevant degrees of freedom into a more advanced EoS.

2.3.1 The realistic EoS

The global aspects of neutron stars, such as the mass-radius (M-R) relation, are determined by the equations of hydrostatic equilibrium. For a spherical object in general relativity (GR), these are the so-called TOV (Tolman-Oppenheimer-Volkov) equations:

$$\frac{dP}{dr} = -\frac{G(m(r) + 4\pi r^3 P/c^2)(\rho + P/c^2)}{r(r - 2Gm(r)/c^2)} \tag{2.1}$$

$$\frac{dm(r)}{dt} = 4\pi\rho r^2 \tag{2.2}$$

where P and ρ are the pressure and mass-energy density, respectively, and $m(r)$ is the gravitational mass enclosed within a radius r and G is the gravitational constant. To get a realistic EoS these equations must be solved. The TOV puts two important bounds on the maximum mass of the neutron star, $R \leq 2GM/c^2$ which is excluded because of GR and $R \lesssim 3GM/c^2$ which is excluded by causality. The lower bound is given by the rotation of the neutron star and stems from the neutron star's origin in a supernova. Through the use of a realistic EoS we can get a picture of the structure of a neutron star.

2.3.2 Structure

A neutron star has a radius of about 10-20km that can be divided into five major regions, the inner and outer cores, the crust, the envelope and the atmosphere (Fig. 2.1). The atmosphere and envelope contain a negligible amount of mass, but the atmosphere plays an important role in shaping the emergent photon spectrum. The envelope crucially influences the transport and release of thermal energy from the stars surface. The surface region, with typical densities $\rho < 10^6\text{g/cm}^3$, is a region in which temperatures and magnetic fields may affect the EoS. The crust extends about 1 to 2km below the surface and have a density $10^6\text{g/cm}^3 < \rho < 4 \cdot 10^{11}\text{g/cm}^3$. It is a solid region where a Coulomb lattice of heavy nuclei coexist in β-equilibrium with a relativistic degenerate electron gas. Within the crust, at densities above the neutron drip density $4 \cdot 10^{11}\text{g/cm}^3$ where the neutron chemical potential (the energy required to remove a neutron from the filled sea of degenerate fermions) is zero, neutrons leak out of nuclei. At the highest densities in the crust, more of the matter resides in the neutron fluid than in nuclei. At the core crust-interface at $n \approx n_0/3$, $4 \cdot 10^{11}\text{g/cm}^3 < \rho < 2 \cdot 10^{14}\text{g/cm}^3$, nuclei are so closely packed that they are almost touching. The inner core of the neutron star at densities $2 \cdot 10^{14}\text{g/cm}^3 < \rho < 2 \cdot 10^{15}\text{g/cm}^3$, contains mainly superfluid neutrons with a smaller concentration of superconducting protons and normal electrons . At higher densities, typically 2-3 times nuclear matter saturation density, interesting phase transition from a phase with just nucleonic degrees of freedom to quark matter may take place. Furthermore, one may have a mixed phase of quark an nuclear matter, kaon or pion condensate, hyperonic matter etc.

2.3.3 Equations of State

To get new insights into neutron star physics, or nuclear physics in general we need to connect the astrophysical observables with the EoS a consistent matter. By making advanced EoS that take the different phenomena in the neutron star into account and comparing these EoS with the observed luminosities we can get new insights into with EoS corresponds the best with observations.

In this thesis we are going to use equations of state developed by Morten Hjorth-Jensen and Henning Heiselberg. The composition of matter for two of these EoS is shown in Fig. 2.2. It is important to note, by changing the EoS we get a very different composition of matter within the neutron star.

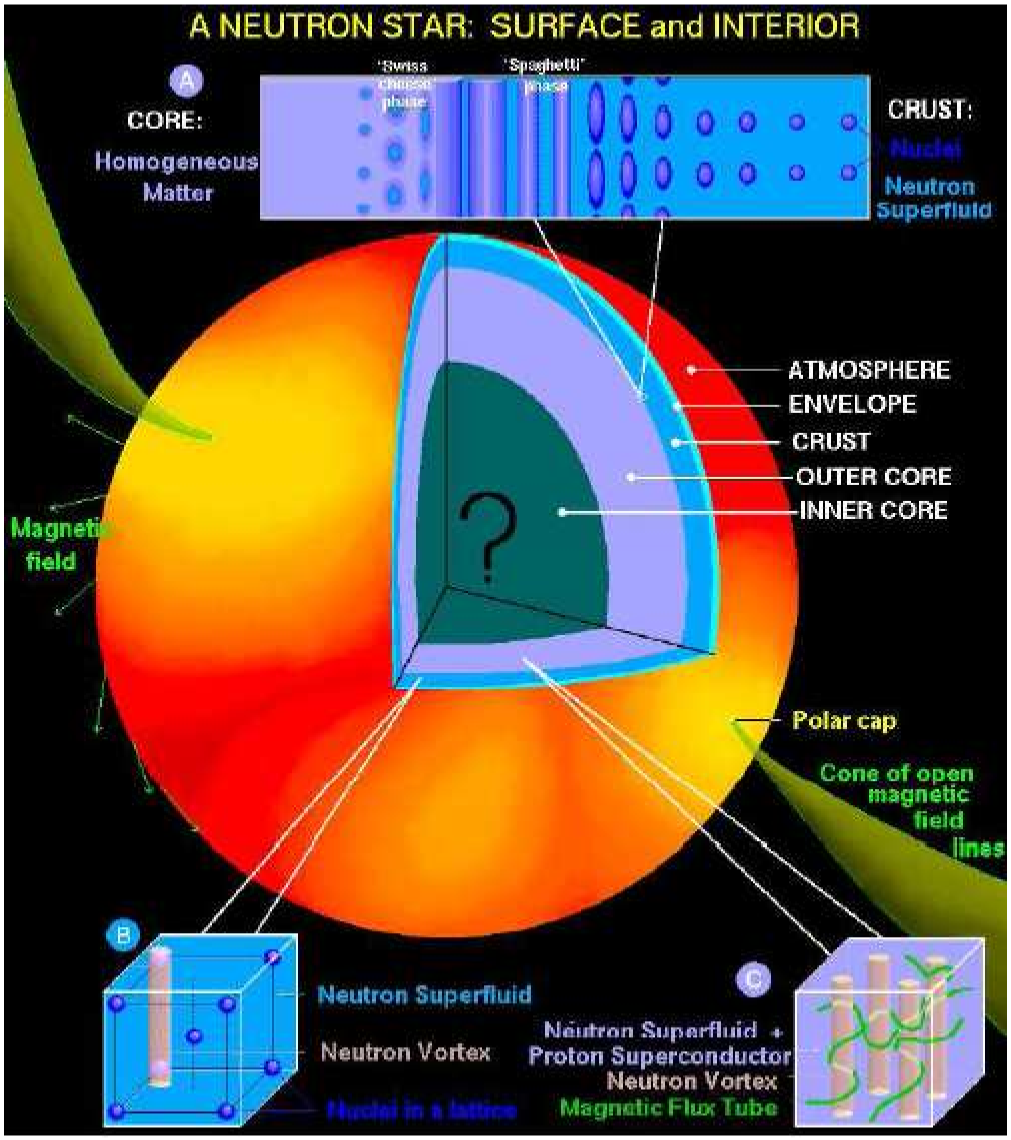

Figure 2.1: This figure shows the major regions of a neutron star. The top bar illustrates the different phases that nuclear matter in the neutron star goes through, from homogeneous in the core to nucleons with independent degrees of freedom in the crust.

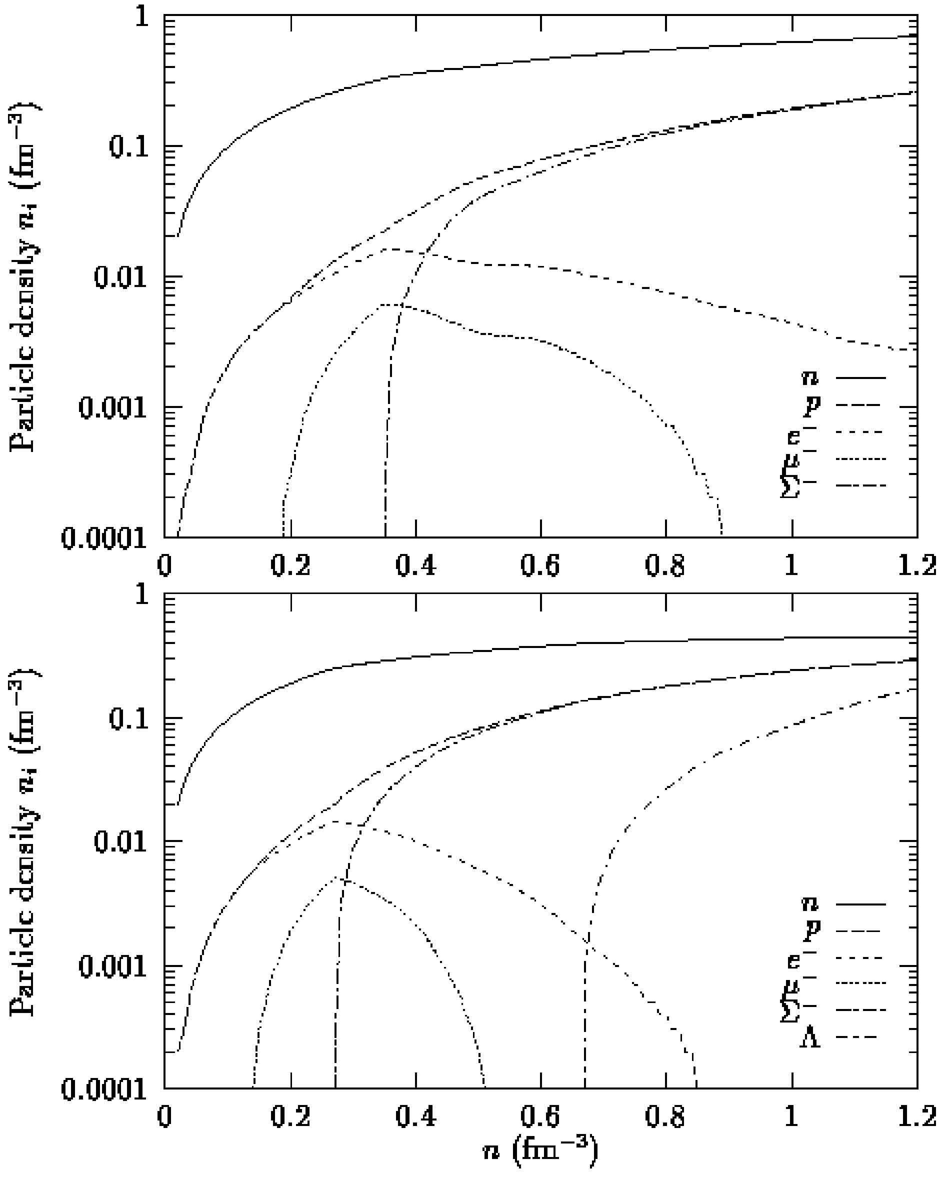

Figure 2.2: Particle densities in β=stable neutron star matter as functions of the total baryonic density n. The upper panel represents represents the results obtained using Brueckner-Hartree-Fock. In the lower pannel the nucleonic part of the self energy of the neuclons has been replaced with the EoS

Chapter 3

Cooling of a Neutron Star - URCA Processes

This chapter serves as brief summary of the physics of the cooling of neutron stars. The surface temperature, or indirectly the cooling of a neutron star is one of the few astrophysical observables we can measure, so establishing a relation between the equations of state and the surface temperature of a neutron star is of paramount importance. The main source of energy emission from a neutron star is by emission of neutrinos through the URCA processes. Most of this chapter will therefore be devoted to the different kinds of URCA processes. In the first part of this chapter the formalism to do calculations of the luminosity of the neutron star will be build up. After the formalism is established we will use it to calculate the given process's contribution to the neutrino emission. Each process will be given its own section and each section will be divided in three parts. The Feynman amplitude, the phase space factor and the results. The first process will be standard β-decay in vacuum. Then we will go to do the direct URCA process, and in the end, outline the calculation of the modified URCA process. The calculation of each of these processes will be focused on making an expression suitable for numeric simulation. These simulations will be the topic of chapter 5 and will mainly serve as tests for the programming tools developed in this thesis.

3.1 URCA Processes

As discussed in chapter 2, neutron stars are dense objects with a relatively low surface temperature. The main source of energy loss for a neutron star is through neutrino emission. The neutrino is the only particle that manages to escape the neutron star without much loss of energy, see section 3.2.3. The simplest possible process involving emission and absorption of neutrinos is the URCA processes, first proposed by Gamow and Schoenberg [7] as an important process in stellar collapse and supernova explosions. The URCA processes is the dominant

$$n \rightarrow p + l + \bar{\nu}_l, \tag{3.1}$$

from now on called the direct URCA process and the inverse process,

$$p + l \rightarrow n + \nu_l. \tag{3.2}$$

Here l is a lepton (electron, muon, tau).

Temperatures in neutron stars are much less than the Fermi temperatures of the constituents, typically of order 100MeV in energy units, which corresponds to a temperature of order 10^{12}K. Therefor matter is degenerate. This also has the effect of putting the neutron star in β-equilibrium,

$$\mu_n = \mu_p + \mu_e \tag{3.3}$$

If there is muons presents we have to add μ_m to the right hand side of the equation. This corresponds physically to the requirement that it cost no energy to convert a neutron into a proton plus an electron, or vice versa. Since the chemical potential is the energy of a particle at the Fermi surface, there is, at zero temperature, no phase space for neutrinos and anti-neutrinos in the final states of the direct URCA process and its inverse. However, at nonzero temperature, fermions are excited above their respective Fermi surfaces by energies of order k_BT. Particles participating in the direct URCA processes must therefore have energies that lie within $\sim k_BT$ of their respective Fermi energies, and therefore have momentum close to their Fermi momenta. Since the neutrino have momentum $\sim k_BT$ the following equality has to be fulfilled for the direct URCA processes to occur.

$$p_{F_e} + p_{F_p} \geq p_{F_n} \tag{3.4}$$

Since the density of particles of species i is given by

$$n_i = p_{F_i}^3 / 3\pi^2\hbar^3 \tag{3.5}$$

inside a neutron star, and the proton faction is assumed to be of order a few percent. The equality Eq. (3.4) can't be fulfilled. The result is that the direct URCA processes is suppressed inside a neutron star. This prompted Chiu and Salpeter to propose the modified URCA processes in 1964 [4].

$$n + n \rightarrow n + p + l + \bar{\nu}_l \tag{3.6}$$

In this process a bystander particle give the needed momentum to the neutron to make up for the energy-momentum mismatch. The problem with this process is that it infers a nucleon-nucleon interaction. Nucleon-nucleon interactions at low energies is not well understood as it is governed by the strong force. As given by Quantum Cromo Dynamics(QCD) and is not renormalizable at low energy levels. This makes the calculations of the scattering amplitude of this process a very complicated matter. Friman and Maxwell's approach to this problem was to make an effective interaction consisting of a long-range one-pion-exchange part and one short-range part parameterizing the many-body correlations using Fermi-liquid parameters [6]. This method of calculating the interaction has later been used in different articles to study different EoS.

3.2 Preliminaries to the calculations

In this section we will introduce the formalism needed to calculate the transition rate of a given process and thereby the luminosity.

3.2.1 Transition-rates

The main equation that governs transition-rates is Fermi's golden rule Eq. (3.7). Here stated without proof , but the derivation can be found in any good book on quantum field theory[14, 16].

$$\text{Transition rate} = 2\pi \times \langle|\mathscr{M}|^2\rangle \times (\text{phasespace}). \tag{3.7}$$

Or in more detail,

$$d\Gamma = 2\pi \times |\mathscr{M}|^2 \times S \times (2\pi)^4 \delta^4(\sum p_i - \sum p_f) \times \prod_i \left(\frac{d^3\mathbf{p}_i}{(2\pi)^3 2E_i}\right) \prod_f \left(\frac{d^3\mathbf{p}_f}{(2\pi)^3 2E_f}\right). \tag{3.8}$$

where $\mathscr{M}$ is the Feynman-amplitude. S is a product of statistical factors $1/j!$ for each group of identical particles. $p_i = (E_i, \mathbf{p}_i)$, $E_i = \sqrt{m_i^2 + |\mathbf{p}_i|^2}$. The delta function $(2\pi)^4\delta^4(\sum p_i - \sum p_f)$ makes sure that energy and momentum is conserved.

The transition-rate Γ consists of two main components. The squared Feynman-amplitude $|\mathscr{M}|^2$ and the phase space factor. The Feynman amplitude contains all the dynamical information about the system. It is calculated by evaluating the relevant Feynman diagrams, by summing and averaging over them using the Feynman rules. These rules can also be found in any book about QFT (See the references above). The phase space factor contains all the kinematical information. It depends on the masses, energies and momenta of participating particles. The integration is done over the available phase space of the given particle. Its this factor that all the relevant information from the EoS is inserted.

3.2.2 Emissivity and the Luminosity

The emissivity of a given particle is calculated almost the same way as the transition rate. To evaluate the emissivity you integrate the transition rate over all initial state of the given particle and multiply with the particle energy.

Once you have the emissivity as a function of density and temperature. The luminosity of the star can be calculated by integrating the emissivity over the volume of the star.

3.2.3 Neutrino Transparency

Once we know the neutrino luminosities, we can calculate the cooling timescales. First we need to figure out if the neutrinos escape from the neutron star without interacting with something on the way out and thereby losing it's energy. We need to calculate the neutrino mean free path λ inside a neutron star. Luckily we will see that for a neutron star at $T \lesssim 10^9 K$, $\lambda \gg R$. So the energy loss from escaping is negligible.

Inside a neutron star the following processes are forbidden by conservation of energy and momentum:

$$\nu_e + n \rightarrow p + e^-, \qquad \bar{\nu}_e + p \rightarrow n + e^+ \tag{3.9}$$

$$\nu_e + p + n \rightarrow n + n + e^+ \tag{3.10}$$

The most important interaction for neutrino energy loss would then be inelastic scattering off electrons (for ν_e and $\bar{\nu}_e$) and off muons (for ν_μ and $\bar{\nu}_\mu$). In a degenerate gas, the cross section for ν_e - e^- scattering, assuming only charged current interactions, is for ν_e [19]:

$$\sigma_e \simeq \chi \sigma_0 \left(\frac{E_\nu}{m_e c^2} \right)^2 \frac{E_\nu}{E_F(e)} \tag{3.11}$$

With

$$\sigma_0 \equiv \frac{4}{\pi}\left(\frac{\hbar}{m_e c^2}\right)^{-4}\left(\frac{G_F}{m_e c^2}\right)^2 \quad = \quad 1.76 \times 10^{-44}\text{cm}^2 \tag{3.12}$$

and where

$$\begin{aligned} \chi &\approx 0.1 && \text{(V - A theory: only charged-current interaction),} \\ &\approx 0.06 && \text{(WSG theory: charged- and neutral-current interactions)} \end{aligned}$$

The results are similar for $\bar{\nu}_e$, ν_μ and $\bar{\nu}_\mu$. The mean free path for the neutrino when we neglect natural current interactions is:

$$\lambda_e = (\sigma_e n_e)^{-1}$$

Using results from D. Griffiths Introduction to Elementary Particles [9].

$$\lambda_e \approx \left(9 \times 10^7 km\right)\left(\frac{\rho_{nuc}}{\rho}\right)^{4/3}\left(\frac{100\text{keV}}{E_\nu}\right)^3$$

Since $\lambda_e \ll R_{max} \approx 10km$, energy loss from escaping is negligible. Now we need to take into account the interactions involving neutral currents. Because of $n - \nu$ elastic scattering reactions, the effective mean free path is significantly reduced. We can say that the process is elastic since to a good approximation the neutron energy $E_\nu \ll m_n$ unchanged after the process. The processes we have to consider is:

$$n + \nu_e \rightarrow n + \nu_e, n + \nu_\mu \rightarrow n + \nu_\mu \tag{3.13}$$

Even though elastic scattering does not reduce the energy of the neutrino, it prevents the neutrino from escaping the neutron star directly after emission. The neutrino scatters many times off neutrons in the interior, until it finally encounters an electron and scatters inelastically, or random-walks its way to surface and escapes with its energy unchanged. The cross section for elastic scattering off neutrons is [19]:

$$\sigma_n \quad = \quad \frac{1}{4}\sigma_0\left(\frac{E_\nu}{m_e}\right)^2 \tag{3.14}$$

and the mean free path is

$$\lambda_n = (\sigma_n n_n)^{-1} \simeq 300\text{km}\frac{\rho_{nuc}}{\rho}\left(\frac{100\text{keV}}{E_\nu}\right)^2 \tag{3.15}$$

The effective mean free path for the electron neutrino then becomes:

$$\lambda_{eff} \sim (\lambda_n \lambda_e)^{1/2} \tag{3.16}$$

$$\sim 2 \times 10^5 \text{km} \left(\frac{\rho_{nuc}}{\rho}\right)^{7/6} \left(\frac{100\text{keV}}{E_\nu}\right)^{5/2} \tag{3.17}$$

Where we have used the WSG value for σ_e. We see that even though the interactions involving the natural currents reduce the mean free path considerably, $\lambda_{eff} \gg R_{max}$. So we can conclude that the neutrinos escape the neutron star without considerable loss of energy.

3.2.4 Interactions

The main problem of doing QFT is managing the different interactions in the process. The basic interactions between elementary particles is fairly straight-forward, but interactions between composite particles such as nucleons is a different matter. In this work I assume basic understanding of Feynman-calculus, but a review of the different interactions inside a neutron star will be useful. There are only two significant interactions in the URCA processes. Namely weak interactions and nucleon-nucleon interactions, which is a combination of the different forces (EM, weak, strong) combined. This review will be rather superficial, its main purpose is to present the tools to do calculations with the given interactions. With the golden rule Eq. (3.7) and knowledge about the interactions between the particles inside the neutron star it is possible to compute the different transition-rates inside the neutron star. As an introduction we will start with *beta*-decay in vacuum. This is an example of a purely weak process.

Weak Interactions

The weak force is mediated by the 3 “intermediate vector bosons” W^+,W^- and Z^0. Unlike photons and gluons they are massive,

$$M_W = 82 \pm 2\text{GeV}, M_Z = 92 \pm 2\text{GeV}, \tag{3.18}$$

The propagator of the weak force is the propagator for massive spin 1 particles,

$$\frac{-i(g_{\mu\nu}) - q_\mu q_\nu / M^2}{q^2 - M^2} \tag{3.19}$$

Where M is M_W or M_Z. The momentum transfer $q^2 \ll M$ inside a neutron star so propagator can without loss of precision be approximated to

$$\frac{-i g_{\mu\nu}}{M^2} \tag{3.20}$$

The weak vertex factor for elementary particles is,

$$\frac{-ig_W}{2\sqrt{2}}\gamma^\mu(1-\gamma^5) \tag{3.21}$$

where g_w is the weak coupling constant. Related to the electromagnetic coupling constant through the Weinberg' weak mixing angle, $g_W = g_e/\sin\theta_W$. The factor $1/2\sqrt{2}$ is purely conventional. The γ^μ part is a standard vector coupling, and the $\gamma^\mu\gamma^5$ part is a pseudo-vector (axial) coupling. Mixing together a vector coupling and a axial vector coupling is ill defined, and the result is that the weak interaction does not conserve parity.

We do not have a valid picture for how the weak force couples to composite particles such as the different baryons. This is a result of our limited understanding of nucleon interactions. To make up for this fact an effective vertex factor is introduced.

$$\left(1-\gamma^5\right) \rightarrow \left(C_V - C_A\gamma^5\right) \tag{3.22}$$

Here C_V is a correction to the vector coupling and C_A to the axial vector. Experiments show that for weak interactions involving protons and neutrons,

$$C_V = 1.000 \pm 0.003 \quad C_A = 1.26 \pm 0.002 \tag{3.23}$$

but they can be different for other composite particles. As we can see the vector current is almost not modified by the strong interactions. It is hypothesised that it is conserved by some unknown rule. Called the "Conserved Vector Current" (CVC). In the same way the axial current is almost conserved and get a similar rule, "Partially Conserved Axial Current" (PCAC).

Nucleon-Nucleon Interactions

In Nucleon-Nucleon(N-N) interactions the strong force comes into play. The most widely accepted fundamental theory for the strong force is Quantum chromodynamics (QCD). QCD is a non-Abelian generalization of the gauge-theory QED. Which uses the symmetry group SU(3) coupled to fermions (quarks) to make a theory for the strong force. Quarks is the elementary fermions that makes up protons, neutrons and other hadrons. In the same way a bound quark-antiquark pair makes up mesons.

Our N-N interactions is completely determined by QCD. However the coupling constants of QCD is too large to use perturbation theory at least at our energies. The strong force coupling constant is approximately 1. For comparison the electro-magnetic coupling is 1/137. So we cant use Feynman's method for calculating the cross-sections using QCD. However the strong coupling is an inverse function of scale (energy), which makes the theory re-normalizable at high energies.

This phenomena is behind Quark confinement. The quark-quark potential is a linearly rising function of the separation distance r:

$$V(r) = \sigma r - \frac{k}{r}$$

However once we separate the quarks away from each other we generate quark-antiquark pairs out of the vacuum that makes the potential level off. This problem is usually circumvented by the so-called “bag models” where we say that the quarks are confined to a bag. This then boils down to a problem of the size of the bag. As long as the bag is small ($R < 0.5fm$), in essence as long as the energies involved are relatively small, there should be enough space for conventional hadrons like nucleons, mesons, and isobars to represent the essential degrees of freedom for most physics phenomena. Meson exchange should therefor be a valid picture. But we have to be careful, this might not be the case inside the neutron star. This model breaks down in the extreme densities which we might encounter in the neutron star core. But for the rest of the neutron star the appropriate procedure would be to construct the nuclear force from meson-nucleon and meson-isobar vertices in what is called a One-Boson-Exchange model (OBE).

$$V(r) = \sum_{\text{vertices}} V_{\text{vertex}}(r, \tau_1 \cdot \tau_2, \sigma_1 \cdot \sigma_2, m, g, \Lambda) \tag{3.24}$$

The propagator for this effective interaction would be the standard meson-propagator,

$$\frac{-i(g_{\mu\nu}) - q_\mu q_\nu / M^2}{q^2 - M^2} \tag{3.25}$$

Where M is the meson mass, and q is the transfered energy and momenta.

In general you can make any Lorentz invariant interaction by using the Fermi's invariant γ-matrices.

$$S = 1^{(1)}1^{(2)}, \quad V = \gamma_\mu^{(1)}\gamma_\mu^{(2)}, \quad T = \frac{1}{2}\sigma_{\mu\nu}^{(1)}\sigma_{\mu\nu}^{(2)}, \tag{3.26}$$

$$A = i\gamma_5^{(1)}\gamma_\mu^{(1)} i\gamma_5^{(2)}\gamma_\mu^{(2)}, \quad P = \gamma_5^{(1)}\gamma_5^{(2)}. \tag{3.27}$$

For testing purposes in this thesis we will just use a one-pion-exchange model. But the program is made in such as way that you could parameterize the interaction using any matrices desirable.

3.3 β-decay in vacuum

In this section we will look at the simplest form of neutrino emission possible. The decay of the neutron into a proton, electron and a anti-electron-neutrino in vacuum (Figure 3.1).

$$n \to p + l + \bar{\nu}_l, \tag{3.28}$$

Using Fermi's golden rule Eq. (3.7). Working in the neutron rest frame. We

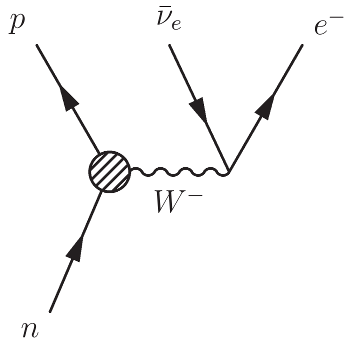

Figure 3.1: β-decay

get,

$$d\Gamma = \frac{|\mathscr{M}|^2}{\hbar}\left(\frac{1}{2E_n}\right)\left(\frac{d^3\mathbf{p}_{\nu_e}}{(2\pi)^3 2E_{\nu_e}}\right)\left(\frac{d^3\mathbf{p}_p}{(2\pi)^3 2E_p}\right)\left(\frac{d^3\mathbf{p}_e}{(2\pi)^3 2E_e}\right)\times(2\pi)^4\delta(p_n - p_{\nu_e} - p_p - p_e). \tag{3.29}$$

Next step is to calculate the Feynman amplitude and evaluate the phase space factor.

3.3.1 Feynman amplitude

Evaluating the Feynman diagram (Figure. 3.1) using the Feynman rules, the Feynman amplitude becomes,

$$\mathscr{M} = \left\{\bar{u}(p)\frac{-ig_W}{2\sqrt{2}}\gamma^\mu(C_V - CA\gamma^5)u(n)\right\}\left\{\frac{ig_{\mu\nu}}{M_W^2}\right\} \tag{3.30}$$

$$\times \left\{\bar{u}\left(e^-\right)\frac{-ig_W}{2\sqrt{2}}\gamma^\nu(1-\gamma^5)v(\nu_e)\right\} \tag{3.31}$$

$$= \frac{g_W^2}{8M_W^2}\bar{u}(p)\gamma^\mu(C_V - CA\gamma^5)u(n)\,\bar{u}\left(e^-\right)\gamma^\nu(1-\gamma^5)v(\nu_e) \tag{3.32}$$

As an example I will evaluate this expression using the rules in Appendix B, its important to note that these rules are just algebraic simplifications. It is possible to just expand all the factors in their proper matrix representations and do the matrix multiplications and the contractions. In this example $C_V = C_A = 1$. This simplifies the calculations a great deal. But it is no problem evaluating the expression with $C_V = 1 \quad C_A = 1.26$. It is just a lot of straight forward algebra, just stick to the rules in Appendix B.

Using Cashimir's trick, we get

$$\begin{aligned}\sum_{\text{spin}} |\mathscr{M}|^2 &= \left(\frac{g_W^2}{8M_W^2}\right)^2 && (3.33)\\ &\times \; Tr[\gamma^\mu(1-\gamma^5)(\not{p}_n - m_n)\gamma^\nu(1-\gamma^5)(\not{p}_p - m_p)] && (3.34)\\ &\times \; Tr[\gamma^\mu(1-\gamma^5)(\not{p}_{\nu_e})\gamma^\nu(1-\gamma^5)(\not{p}_e - m_e)]. && (3.35)\end{aligned}$$

Evaluating the traces we get for the first trace,

$$8\left[p_n^\mu p_p^\nu + p_n^\nu p_p^\mu - g^{\mu\nu}(p_n \cdot p_p) - i\epsilon^{\mu\nu\lambda\sigma} p_{n\lambda} p_{p\sigma}\right], \tag{3.36}$$

and similarly for the second trace,

$$8\left[p_{\nu_e\mu} p_{e\nu} + p_{\nu_e\nu} p_{\nu_e\mu} - g_{\mu\nu}(p_{\nu_e} \cdot p_e) - i\epsilon_{\mu\nu\kappa\tau} p_{\nu_e}^\kappa p_e^\tau\right]. \tag{3.37}$$

when multiplied together gives,

$$\sum_{\text{spin}} |\mathscr{M}|^2 = 4\left(\frac{g_W}{M_W}\right)^4 (p_n \cdot p_{\nu_e})(p_p \cdot p_e). \tag{3.38}$$

Using $C_V = 1, C_A = 1.26$ vi get for the Feynman amplitude,

$$\begin{aligned}\sum_{\text{spin}} |\mathscr{M}|^2 &= \frac{1}{2}\left(\frac{g_W}{M_W}\right)^4 [(p_n \cdot p_{\nu_e})(p_p \cdot p_e)(1 + C_A)^2 && (3.39)\\ &+ \; (p_n \cdot p_e)(p_{\nu_e} \cdot p_p)(1 - C_A)^2 && (3.40)\\ &- \; m_p m_n (p_{\nu_e} p_e)\left(1 - C_A^2\right)] && (3.41)\end{aligned}$$

As you can see if we set $C_A = 1$ we get back the expression in Eq.3.38.

3.3.2 Phase Space factor

Simplifying Eq. 3.29 and the using the rest frame of the neutron,

$$d\Gamma = \frac{2\langle|\mathscr{M}|^2\rangle}{(4\pi)^5 m_n} \frac{d^3\mathbf{p}_{\nu_e} d^3\mathbf{p}_p d^3\mathbf{p}_{e^-}}{E_{\nu_e} E_p E_{e^-}} \delta(m_n - p_{\nu_e} - p_p - p_e). \tag{3.42}$$

Using the momentum part of the delta-function to do the momentum integral over the proton and evaluating the reminding phase space factor using spherical-coordinates,

$$d\Gamma = \frac{2\langle|\mathscr{M}|^2\rangle}{(4\pi)^5 m_n}\frac{|\mathbf{p}_{\nu_e}|^2 d\mathbf{p}_{\nu_e}\sin\theta_{\nu_e}d\theta_{\nu_e}d\phi_{\nu_e}|\mathbf{p}_{e^-}|^2 d\mathbf{p}_{e^-}\sin\theta_{e^-}d\theta_{e^-}d\phi_{e^-}}{E_{\nu_e}E_pE_{e^-}}\delta(m_n - E_{\nu_e} - E_p - E_e). \tag{3.43}$$

Where

$$E_{\nu_e} = |\mathbf{p}_{\nu_e}|, \quad E_p = \sqrt{m_p^2 + (\mathbf{p}_{e^-}\cdot\mathbf{p}_{\nu_e})^2}, \quad E_{e^-} = \sqrt{m_{e^-}^2 + \mathbf{p}_{e^-}^2} \tag{3.44}$$

Note that,

$$|\mathbf{p}_{\nu_e}|^2 d\mathbf{p}_{\nu_e}\sin\theta_{\nu_e}d\theta_{\nu_e}d\phi_{\nu_e} = |\mathbf{E}_{\nu_e}|^2 d\mathbf{E}_{\nu_e}\sin\theta_{\nu_e}d\theta_{\nu_e}d\phi_{\nu_e} \tag{3.45}$$

and

$$E_p = \sqrt{m_p^2 + |\mathbf{p}_{e^-}|^2 + |\mathbf{p}_{\nu_e}|^2 - 2|\mathbf{p}_{\nu_e}||\mathbf{p}_{e^-}|\cos\theta_{\nu_e}}. \tag{3.46}$$

We can do the substitution,

$$dE_p = \frac{E_{\nu_e}|\mathbf{p}_{e^-}|\sin\theta_{\nu_e}d\theta_{\nu_e}}{E_p}. \tag{3.47}$$

The result after doing the ϕ_{ν_e} integral is,

$$d\Gamma = \frac{\langle|\mathscr{M}|^2\rangle}{(4\pi)^4 m_n}\frac{dE_{\nu_e}dE_p|\mathbf{p}_{e^-}|^2 d\mathbf{p}_{e^-}\sin\theta_{e^-}d\theta_{e^-}d\phi_{e^-}}{E_{e^-}|\mathbf{p}_{e^-}|}\delta(m_n - E_{\nu_e} - E_p - E_e). \tag{3.48}$$

After the $d\theta_{e^-}$ and $d\phi_{e^-}$ integrals and noting that,

$$|\mathbf{p}_{e^-}|d|\mathbf{p}_{e^-}| = E_{e^-}dE_{e^-}, \tag{3.49}$$

the transition rate becomes,

$$d\Gamma = \frac{\langle|\mathscr{M}|^2\rangle}{(4\pi)^3 m_n}dE_{\nu_e}dE_pdE_{e^-}\delta(m_n - E_{\nu_e} - E_p - E_e). \tag{3.50}$$

Using the delta function and Eq. 3.46 to evaluate the dE_p integral we get the following restrictions on the dE_{ν_e} integral

$$E_{\nu_e\pm} = \frac{\frac{1}{2}\left(m_n^2 - m_p^2 + m_{e^-}^2\right) - m_nE_{e^-}}{m_n - E_{e^-} \mp |\mathbf{p}_{e^-}|} \tag{3.51}$$

So the final result for this integral is

$$d\Gamma = \frac{\langle|\mathscr{M}|^2\rangle}{(4\pi)^3 m_n}dE_{\nu_e}dE_{e^-} \tag{3.52}$$

with the restrictions from Eq. 3.51 on the dE_{ν_e} integral.

3.3.3 Result

It is very difficult to do the integral in Eq. 3.52 analytically. But the result numerically for the decay-rate with $C_A = 1.0$ is

$$\tau = \frac{1}{\gamma} = 1316s \tag{3.53}$$

and for $C_A = 1.26$

$$\tau = 914s \tag{3.54}$$

The experimental results for neutrino decay in vacuum is

$$\tau = 891s \pm 16s \tag{3.55}$$

The discrepancy in the results is because of the underlying strong interactions.

3.4 The direct URCA process within the neutron star

The direct URCA process is strongly suppressed inside the neutron star, but it might be the source neutrino emission when the proton fraction is high enough [13].

3.4.1 Feynman amplitude

The Feynman amplitude for the direct URCA process the same as for β-decay in vacuum Eq. (3.41).

3.4.2 Phase Space factor

Starting out with Fermi's Golden Rule Eq. (3.7) and integrating over all initial states for the neutron and multiplying with the neutrino energy,

$$d\varepsilon_\nu = |\mathscr{M}|^2 \left(\frac{d^3\mathbf{p}_n}{(2\pi)^3 2E_n}\right)\left(\frac{d^3\mathbf{p}_{\nu_e}}{(2\pi)^3 2E_{\nu_e}}\right)\left(\frac{d^3\mathbf{p}_p}{(2\pi)^3 2E_p}\right)\left(\frac{d^3\mathbf{p}_e}{(2\pi)^3 2E_e}\right) \times (2\pi)^4 \delta(p_n - p_{\nu_e} - p_p - p_e). \tag{3.56}$$

The integration is over all allowed initial and final states. To include this in the calculations we introduce the Fermi-Dirac distribution functions,

$$f(E_j) = \left[e^{(E_j-\mu_j)/k_BT} + 1\right]^{-1} \tag{3.57}$$

This gives us the following expression for the emissivity,

$$d\varepsilon_\nu = |\mathscr{M}|^2 \left(\frac{d^3\mathbf{p}_n}{(2\pi)^3 2E_n}\right)\left(\frac{d^3\mathbf{p}_{\nu_e}}{(2\pi)^3 2E_{\nu_e}}\right)\left(\frac{d^3\mathbf{p}_p}{(2\pi)^3 2E_p}\right)\left(\frac{d^3\mathbf{p}_e}{(2\pi)^3 2E_e}\right) \times (2\pi)^4\delta(p_n - p_{\nu_e} - p_p - p_e) \times f(E_n)(1-f(E_e))(1-f(E_p)) \quad (3.58)$$

In doing the integration over the angles I will follow Appendix F of Shapiro and Teukolsky[17]. First, let us look at just the integration over the solid angles Ω. In Eq. 3.58 this part is given by,

$$\int d\Omega_n \int d\Omega_\nu \int d\Omega_e \int d\Omega_p \delta^{(3)}(\mathbf{p}_n - \mathbf{p}_e - \mathbf{p}_p - \mathbf{p}_\nu). \quad (3.59)$$

We rewrite the δ function as

$$\frac{\delta(|\mathbf{p}_p| - |\mathbf{p}_n - \mathbf{p}_e - \mathbf{p}_\nu|)}{|\mathbf{p}_p|^2}\delta(\Omega_p - \Omega_{n-e-\nu}). \quad (3.60)$$

The integral over Ω_p yields then

$$\int d\Omega_n \int d\Omega_\nu \int d\Omega_e \frac{\delta(|\mathbf{p}_p| - |\mathbf{p}_n - \mathbf{p}_e - \mathbf{p}_\nu|)}{|\mathbf{p}_p|^2}. \quad (3.61)$$

Choosing

$$x = \mathbf{p}_n - \mathbf{p}_\nu, \quad (3.62)$$

and the z-axis for p_e along x and rewriting the last δ-function as (setting $|\mathbf{p}_i| = p_i$)

$$\delta(|\mathbf{p}_p| - |\mathbf{p}_n - \mathbf{p}_e - \mathbf{p}_\nu|) = \frac{\delta(cos\theta_{xe} - (p_p^2 - x^2 - p_e^2)/(2xp_e))}{xp_e/p_p}, \quad (3.63)$$

one obtains for the integration over the angles

$$\frac{2\pi}{|\mathbf{p}_p||\mathbf{p}_e|}\int d\Omega_n \int d\Omega_\nu \frac{1}{|\mathbf{p}_n - \mathbf{p}_\nu|}. \quad (3.64)$$

The assumption we will make is that $|\mathbf{p}_n| >> |\mathbf{p}_\nu|$. One should keep this assumption in mind because if one is to study such processes in supernova environment where neutrinos can be rather energetic, the above approximation is no longer valid.

The above integral is then

$$\frac{2\pi}{|\mathbf{p}_p||\mathbf{p}_n||\mathbf{p}_e|}\int d\Omega_n \int d\Omega_\nu. \quad (3.65)$$

The integration over the angles depends now on the angle dependence in the squared Feynman amplitude. If there is no angle dependence, as assumed by Shapiro and Teukolsky (and most workers in the field), the integral is trivial and gives just

$$\frac{(4\pi)^3}{2|\mathbf{p}_p||\mathbf{p}_n||\mathbf{p}_e|}. \tag{3.66}$$

If we however insist on the angle dependence in Eq. (3.41), we need to do the integration explicitly.

The ugly part begins now, namely the integration over the momenta in Eq. (3.58).

Let us then look at the first of the above terms. The integral to be performed is

$$\begin{aligned}\varepsilon_\nu = \frac{1}{2^{12}\pi^8}\int\frac{p_n^2dp_n}{2E_n}\int\frac{p_p^2dp_p}{2E_p}\int\frac{p_e^2dp_e}{2E_e}\int\frac{p_\nu^2dp_\nu}{2}\\ \times f(E_n)(1-f(E_e))(1-f(E_p))\delta(E_n-E_e-E_p-E_\nu)\\ \times\frac{(4\pi)^3}{2|\mathbf{p}_p||\mathbf{p}_n||\mathbf{p}_e|}|\mathscr{M}|^2\end{aligned} \tag{3.67}$$

Using $E_\nu = p_\nu$ and for the other particles $E_jdE_j = p_jdp_j$ (this also cancels the denominators $|\mathbf{p}_p||\mathbf{p}_n||\mathbf{p}_e|$) one obtains

$$\begin{aligned}\varepsilon_\nu = \frac{1}{2^7\pi^5}\int dE_n\int dE_p\int dE_e\int E_\nu^2dE_\nu\\ f(E_n)(1-f(E_e))(1-f(E_p))|\mathscr{M}|^2\delta(E_n-E_e-E_p-E_\nu)\end{aligned} \tag{3.68}$$

Inserting the Fermi-Dirac functions we have

$$\begin{aligned}\varepsilon_\nu = \frac{1}{2^7\pi^5}\int dE_n\int dE_p\int dE_e\int E_\nu^2dE_\nu|\mathscr{M}|^2\delta(E_n-E_e-E_p-E_\nu)\\ \times\frac{1}{1+\exp\beta(E_n-\mu_n)}\times\frac{1}{1+\exp-\beta(E_e-\mu_e)}\times\frac{1}{1+\exp-\beta(E_p-\mu_p)}.\end{aligned}$$

Now we introduce new variables $\frac{x_i}{\beta} = E_i - \mu_i$ and $x_\nu = E_\nu\beta$. We can then rewrite the latter equation as follows

$$\begin{aligned}\varepsilon_\nu = \frac{(k_BT)^5}{2^7\pi^5}\int dx_n\int dx_p\int dx_e\int x_\nu^2dx_\nu\delta(x_n-x_e-x_p-x_\nu)|\mathscr{M}|^2\\ \times\frac{1}{1+e^{x_n}}\frac{1}{1+e^{-x_p}}\frac{1}{1+e^{-x_e}}.\end{aligned} \tag{3.69}$$

3.4.3 Result

Eq. 3.69 can be simplified further. But is now ideal for numerical simulations. The rest of the integration process is rather cumbersome. In the end the result is [13].

$$\varepsilon_\nu = \frac{457\pi}{10080} G_F^2 cos^2\theta_C (1 + 3g_A^2) \frac{m_n m_p \mu_e}{\hbar^{10} c^5} (k_B T)^6 \Theta(p_F^e + p_F^p - p_F^n). \tag{3.70}$$

Where G_F is the Fermi coupling. Θ_C is the Cabbibo angle. g_A is the axial vertex correction. m_n, m_p and μ_e is the neutron mass, proton mass and the electrons chemical potential. The factor $\Theta(p_F^e + p_F^p - p_F^n)$ is $+1$ for arguments larger than 0. This is the requirement for the process to occur.

3.5 The modified URCA process

The last process we are going to look at is the modified URCA process.

$$n + n \rightarrow n + p + l + \bar{\nu}_l \tag{3.71}$$

The direct diagrams contributing to the modified URCA process is illustrated in Figure 3.2. In addition to these three diagrams there is 3 more diagrams analog to the ones in Figure 3.2 which the neutron and proton labels on the outgoing nucleon lines interchanged (and with the location of the weak interaction shifted accordingly). Since they involve a different final state from the diagrams in Figure3.2,they must be added incoherently with those diagrams. Moreover, after the phase space integrals are performed, they give a contribution to the Urca emissivity that is identical to that of the Figure 3.2 diagrams. Hence, they maybe included in the calculation by merely doubling the emissivity obtained with the Figure 3.2 diagrams alone.

3.5.1 Result

Unfortunately the calculation of the modified URCA process was not completed by the time of this thesis deadline. My colleague, Sutharsan Arumugam, has studied a similar process, Neutrino Pair Bremsstrahlung, in his thesis “Cooling of Neutron Stars” [1].

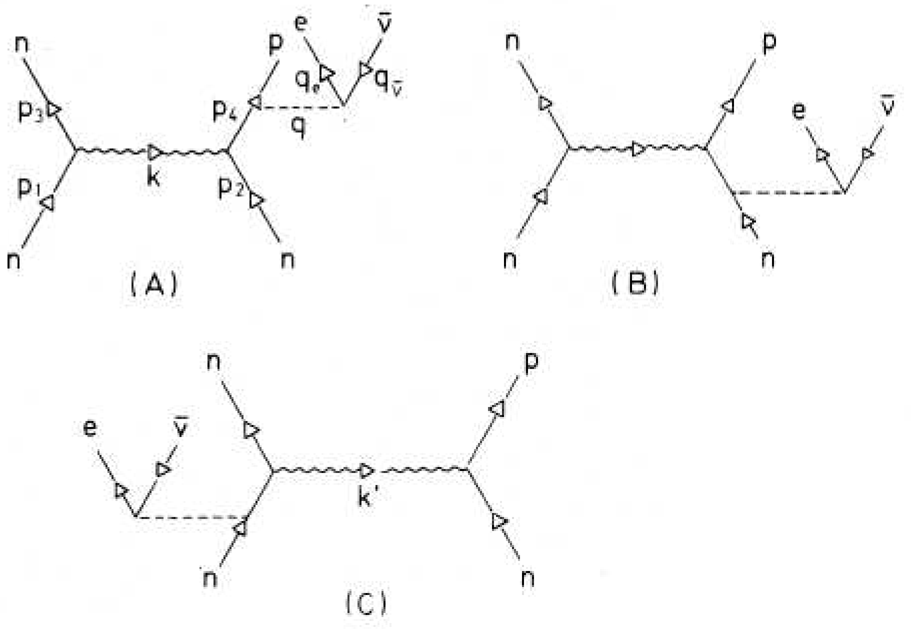

Figure 3.2: The 3 diagrams for the modified URCA process.

Chapter 4

The Program

This part of the thesis will explain how to use the programing tool I have developed to solve Quantum Field Theory (QFT) problems. It's called Feynman Amplituds. FA for short. The tool is implemented in the form of a C++ library. This library contains objects by which you can construct and calculate Feynman-amplitudes. The main focus of the tool has been to make it as general as possible and as easy to extend as possible. Section 4.1.1 will give a short introduction to the program as well as some explanations of some design decisions. Section 4.2 shows how to use the library. In chapter 5 we will test the program by implementing the different URCA processes.

4.1 Introduction

4.1.1 Motivation

The main focus of this thesis is to numerically analyse the processes going on in the interior of a neutron star. The dominant ones being the direct (for high proton-fractions) and modified URCA processes. $n \rightarrow p + \bar{\nu}_e + e^-$ and $(n,p) + n \rightarrow (n,p) + p + \bar{\nu}_e + e^-$ respectively. There are also a host of other more exotic processes to consider as we have seen in chapter 3.

The main thing we are interested in for each of the processes is the decay rate and the neutrino emissivity, as explained in chapter 3. This is a two part problem. The Feynman-amplitude and the phase-space factor both needs to be evaluated. This involve an integral over allowed momentum space, this integration can be done numerically by standard integration routines once the surface of momentum space that conserve energy and momentum has been determined. The integration technique used in this thesis is Gaussian-Quadrature. Gaussian-Quadrature was chosen for simplicity and ease of use. A better choice might have been Monte-Carlo method, which I will try to use for further work.

Because of the needed to look at a lot of different processes, FA is designed to be as general as possible. The requirement for this program was that it could calculate the Feynman-amplitude for the different processes with the momentum and spin of the external particles as variables. It should be able to do this with minimal alteration of code for each process and also be able to add medium effects to the process, such as custom made interaction matrices.

There are of course a lot of other programs out there designed to these tasks and more. REDUCE and FORM to name a few. Had I known how time consuming it would be to build this package, I would rather have used them instead. There are also some programs who automate the integration process.

4.1.2 How it works

Perturbative QFT as explained in chapter 3, is done with the use of the tools developed by Feynman, namely Feynman diagrams and the Feynman rules that govern them. Any standard book on QFT will contain them. From the Feynman rules one can easily sett up the expression for the Feynman amplitude. For example for simple electron scattering show in figure 4.1. The diagram corresponds to the following expression:

$$\sum_r |\mathscr{M}|^2 = \sum_r \bar{U}^{r2}(p2)\{-ie\gamma^\mu\}u^{r1}(p1)iD_F^{\mu\nu}(k)\bar{u}^{r4}(p4)\{-ie\gamma^\nu\}u^{r3}(p3)$$

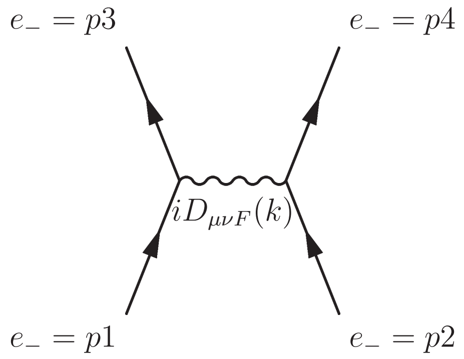

Figure 4.1: Electron scattering

The different parts in this expression is just tensors and can be expressed by simple matrices in any computer language. The problem is to sum over the different indices and do the spin sum. Analytically this is done using some algebraic tricks, namely Dirac algebra and trace identities as seen in appendix B. But it can also be done by simply multiplying and summing, this is the way the FA works. This is of course a very crude way of doing it, but it is easy to implement numerically.

The other way of doing this type of calculation is by implementing a program that can use the algebraic rules to simplify the expression. This reduces the problem to one sum, so the calculation scales very well. However this is a non-trivial thing to implement on a computer and is not fit for a Masters thesis. This is how the standard packages for doing these calculations do it. REDUCE and FORM as mentioned earlier. If I am going to look at any higher order process I will probably use one of these programs.

Advantages

There is a few advantages of doing it the brute-force way. The Feynman amplitude is expressed in a totally generic way as a function of momentum. This means that once you have implemented the calculation for the cross-section using Fermi's golden rule in a generic way. You are free to calculate any process you are interested in that have the same phase-space dependence. In other words, once you have made a function that does the phase-space part for a process, for example scattering of a certain number of particles. You can use the same function to calculate any other scattering rate with the same number of incoming and outgoing particles just by giving the function the generic function for the Feynman amplitude. For example if I calculate electron-electron scattering, I can use the same function to look at neutrino-electron scattering or OBE scattering of nucleons. The only thing that you will need to pass is the integration parameters.

Another advantage is the algebraic way of doing things requires that you sum over initial and final spins to take advantage of the completeness relation, and trace identities. Since it is rare to know incoming spins of particles this is usually not a problem, so most calculations averages over initial and final spins. But with doing it the brute-force way you can calculate the amplitudes for any spin configuration your interested in.

The third advantage is that it is very easy to extend the brute-force way of doing it. You can make custom none-Lorentz invariant interactions if you like!

Disadvantages

There are one huge problem by doing it the brute force way. It is order of magnitude slower than than doing it algebraically. This is a huge problem, but for the fairly simple processes we are interested in the computer time used to calculate an amplitude is fairy short (0.5s and down). Another problem that is related to the first one is that by doing it brute-force makes the program scale very poorly. Because of the sum over the Dirac indices, the number of calculation goes as 4^N where N is the number of indices. And because of the spin sum it goes as i^N where N is the number of particles and i is the average number of spin states of the particle. As we can see calculating higher order processes using this method will take years. But for relatively small particle numbers, and for processes with few Dirac indices. This way of doing it is valid.

Another disadvantage comes from the generality. Since the Feynman amplitude is expressed as a function of the momentum vectors. The phase-space integration has to produce these vectors. This can make the integration a bit cumbersome, since whenever you change variables, you have to manage to get back to the momentum vectors to calculate the Feynman amplitude.

4.1.3 Parallel Processing and Distributed Processing

Most numerical-physics problems today need more computer power than one pc (or one CPU) can supply. This project is using a really slow method for solving QFT so we need all the processing power we can get hold off.

This is where parallel processing(PP) and distributed processing (DP) comes in. PP is simply to code your program in such a way that it can use more than one CPU's on one machine in parallel. This is usually done by threading. DP is extending this concept to more than one machine. To make your programs use of DP you need to split up the workload of your program into pieces. So that each piece can be sent to a different computer for processing. To do this you need to have prepared a set of computers to handle DP, and you need a library for your programming language that can handle the communication between the computers. I'm not going to go into technical details of PP or DP as that is not in the scope of this thesis.

The DP specification I have been using on this thesis is MPI. The implementation is MPICH2.

MPI works in a rather straightforward way. Once you start your program on a node in the cluster, every other node starts up the same program. Every node in the cluster is given a number(rank) that you can use to refer to that node. The node you started the program on, the root node, is given rank 0. You can retrieve that number by calling the MPI method getRank(). You then simply use that number to determine what part of the workload that part of the program is going to take care of.

This method of doing things really hands itself to Gaussian Quadrature integration, and most other form of integration for that matter. You can simply distribute a set of integration points to the different computers, and make then send the results back to the root node after it is completed. If you want an example you can look in Appendix C

4.2 Usage

In this section we will go through the basic usage of FA. Some complete examples for electron-electron scattering and β-decay of the neutron is given in the end.

4.2.1 Installation

As mentioned earlier FA is implemented as a C++ library. So to use it you simply have to direct your compiler to the FA headers directory and library folder. FA depends on Blitz++ so you need to have that installed as well. So to use FA simply include fa.h to your C++ program.

```
#include bfft.h
```

4.2.2 Basic usage

The main uses of FA is to solve Feynman amplitudes numerically. It does this by first building up the expression for the Feynman amplitude. After the expression has been made you can simply call a number of methods on it to do different tasks; contract the indices, do the spin sum or calculate the total amplitude etc.

One of the focuses of FA is generality and extendability. Therefore FA is coded in an object-oriented way. The main objects is the amplitude and particle objects. The amplitude contains the information about the expression, while the particles contain the information about the particles, mass, spin, momentum etc. You need one particle object for external particle. The amplitude object is simply instanced by:

```
Amplitude amp = new Amplitude();
```

And the particle object:

```
Particle* p1 = new particle_type();
```

The particle objects need to instanced by calling the constructor of the given particle you are interested in i.e. to make an electron and a W^-:

```
Particle* p1 = new Electron();
Particle* wneg = new Wneg();
```

The amplitude consists of a linked-list with each of the different elements of the expression i.e. The different spinors, vertexes etc. Each element has a corresponding object. And as for particles, you need to specify what type the different elements by using the constructor. There are 3 main element types, the spinor, vertex and propagator. Example of spinor objects:

```
Element* u1 = new U(p1);
Element* ub1 = new Ubar(p2);
```

Example of vertex objects:

```
Element* v1 = new Vertex_EM(indice01);
Element* v1 = new Vertex_Weak(indice01);
Element* v1 = new Vertex_Weak(G_v,G_a,indice01);
```

Example of propagator objects:

```
Element* fprop = new FermionProp(p1);
Element* bprop = new BosonProp(wneg,indice01 ,indice02);
```

As you can see the vertexes and propagators need to be created with a set of indices. These are just numbers ranging from 1 (equivalent of μ) up to the number of indices in the expression. Also note that some of the constructors can be overloaded to pass more information to the given object. For example the weak vertex object can be created with other values for the vector and axial vector corrections.

After all the elements of the expression has been created you need to add them to the amplitude by using the method add() in the amplitude object. The elements need to be added in the correct order from left to right. That means the first element to be added should be the one to the left in the expression, usually a spinor. Example:

```
amp->add(ub1)
amp->add(v1)
amp->add(u1)
amp->add(bprop)
etc.
```

Once this is done, the amplitude object is finished and ready for use. But before you can use it for anything useful. You need to set the impulses and spinstates of the different incoming and outgoing particles. This can be done with the particle method setImpulse(Impulse i) and setSpinstate(int spinstate). The Impulse object is simply an blitz++ array.

To create a electron with impulse $1MeV/c$ along the Z-axis, with spin along the Z-axis.

```
Particle* p1 = new Electron();
Impulse i;
i = 0,0,1;
p1->setImpulse(i)
p1->setSpinstate(1);
```

After the states of all the particles has been set, you can use the method calculateAmplitude() to get the amplitude of the given process. If you want to do the complete spinssum the amplitude object has a spinSum() method, simply call it and it will do a sum of all possible spin combinations.

So to sum it all up, heres a example of electron-positron scattering (figure 4.2).

$$\sum_r |\mathscr{M}|^2 = \sum_r \bar{U}(p2)\{-ie\gamma^\mu\}u1(p1)iD^{F\mu\nu}(k)\bar{u}(p4)\{-ie\gamma^\nu\}u(p3)$$

First make the particles and amplitude:

```
Amplitude amp = new Amplitude();
Particle* p1 = new Electron();
Particle* p2 = new Positron();
Particle* p3 = new Electron();
Particle* p4 = new Positron();
Particle* photon = new Photon();
```

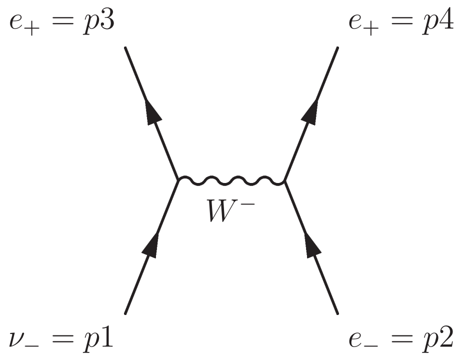

Figure 4.2: Electron-Positron scattering

Then the different elements:

```
    Element* ub1 = new Ubar(p3);
    Element* v1 = new Vertex_EM(1);
    Element* u1 = new U(p1);
    Element* bprop = new PhotonProp(photon,1,2);
    Element* ub2 = new Ubar(p4);
    Element* v2 = new Vertex_EM(2);
    Element* u2 = new U(p2);
```

Then add them to the amplitude object:

```
    amp->add(ubar1);
    amp->add(v1);
    amp->add(u1);
    amp->add(bprop);
    amp->add(ubar2);
    amp->add(v2);
    amp->add(u2);
```

Set impulses:

```
  Impulses i1,i2,i3,i4;
  i1 = 0,0,1
  i2 = 0,0,-1
  i3 = 0,0,1
  i3 = 0,0,-1
  p1->setImpulse(i1);
  p2->setImpulse(i2);
  p3->setImpulse(i3);
  p4->setImpulse(i4);
```

And in the end calculate the spinsum:

```
  cout << "Feynman Amplitude = " << amp->calculateSpin() << endl;
```

And thats all there is to it. I have added some complete examples of in Appendix C section C.1.2.

4.3 Testing

Each object of FA the FA source code has been tested independently both for coding errors as well as consistency physically. We have also used FA to calculate a host of different Feynman amplitudes. The real test is to calculate the URCA processes described in chapter 3 using FA. This is the topic of the chapter 5. You can view the code of the tests in appendix C.

Chapter 5

Results

In this chapter we present numerical simulations of the processes discussed in chapter 3 using the program Feynman Amplitudes(FA). The physical interpretation of the results is secondary, as the results obtained in these simulations does not improve on the most current calculations of the URCA processes. These processes has been discussed in great detail in other articles such as the original article of Maxwell and Friman [6]. As long as we don't have a good interaction matrix we can not hope to improve on the results, but comparing the numerical results of FA with the current approximations should give us insights into how well FA is working. All the integration routines is implemented using Gaussian-Quadrature.

5.1 β-decay comparison

The first comparisons is between FA and the analytical expression for the β-decay Eq. (3.52). This test is to see how well FA responds to the C_A parameter. In this test we have chosen the β-decay decay-rate at different values of the axial vertex coupling correction C_A. The source code of the test can be view in Appendix C Section C.2. The output of the program is for $C_A = 1.0$:

```
Decay-rate=1318.54
```

and for $C_A = 1.26$:

```
Decay-rate=927.328
```

In comparison the analytical result from chapter 3, for $C_A = 1.0$, $\tau = 1316$ and for $C_A = 1.26$, $\tau = 914$. The electron energy distribution from neutron β-decay is shown in fig 5.1 for $C_A = 1.0$ and fig 5.2 for $C_A = 1.26$. As you can see the numerical and analytical results match quite well. There is a relative difference of $\approx 1\%$. This is probably because of some numerical issues with FA, hopefully they will be resolved shortly.

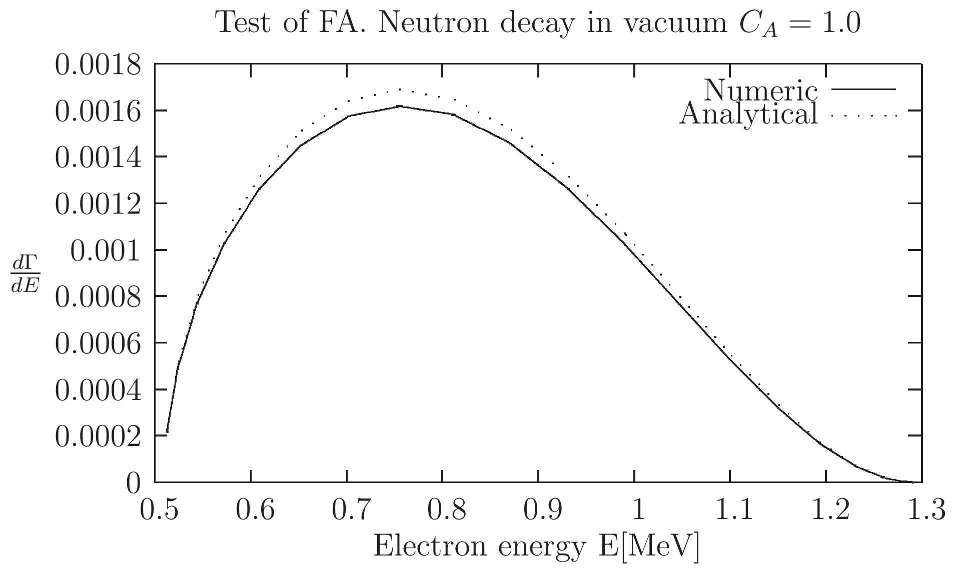

Figure 5.1: Numerical and Analytical calculation of the electron energy distribution from neutron beta decay. $C_A = 1.0$

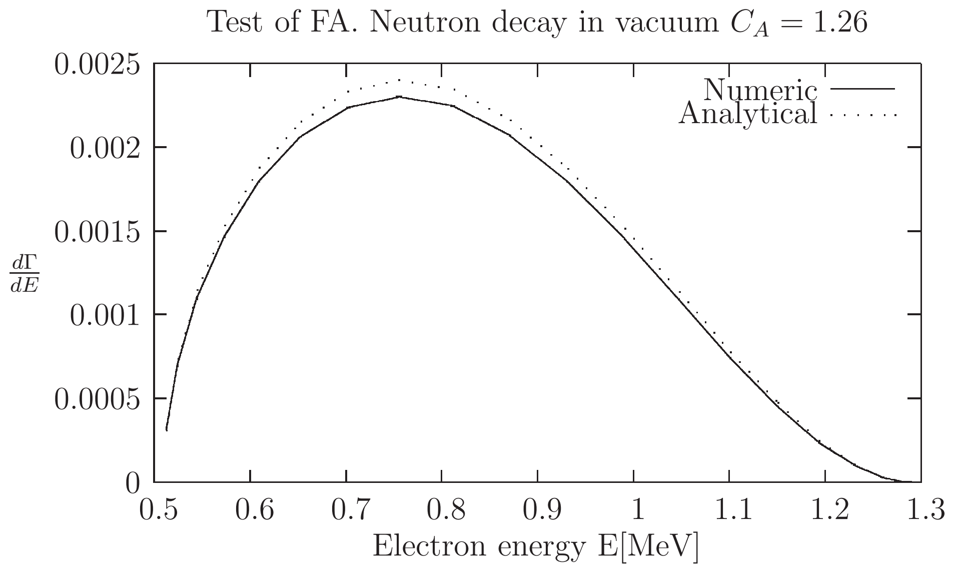

Figure 5.2: Numerical and Analytical calculation of the electron energy distribution from neutron beta decay. $C_A = 1.26$

5.2 Direct URCA process comparison

In the case of the Direct URCA process we compare our numerical results with expression obtained by Prakash in chapter 3, Eq. (5.1).

$$\epsilon_{URCA} = \frac{457\pi}{10080} G_F^2 \cos\theta_C{}^2 (1 + 3g_A^2) \frac{m_n m_p \mu_e}{\hbar^{10} c^5} (k_B T)^6 \Theta_t. \tag{5.1}$$

Where G_F is the Fermi coupling. Θ_C is the Cabbibo angle. g_A is the axial vertex correction. m_n, m_p and μ_e is the neutron mass, proton mass and the electrons chemical potential respectively. The factor $\Theta_t = \Theta(p_F^e + p_F^p - p_F^n)$ is $+1$ for arguments larger than 0. This is the requirement for the process to occur at all. We are using the three EoS from chapter 2 section 2.3.3.

1. Standard nuclear matter using 2-body interactions.

2. Standard nuclear matter using 3-body interactions.

3. Nuclear matter with hyperons using 2-body interactions.

For the EoS involving hyperons the direct URCA process is never allowed. The equality $p_F^e + p_F^p - p_F^n > 0$ is never fulfilled for the given densities. The values for the temperature used in these tests is $T = 10^8$ and $T = 10^9$. The comparisons are in figures 5.3, 5.5, 5.7 and 5.9. And the relative differences are in figures 5.4, 5.6, 5.8 and 5.10.

When doing the comparison's we have to be aware that Eq. (5.1) is just an approximation and is not accurate analytically so we can expect a discrepancy. As we can see from the figures 5.4, 5.6, 5.8 and 5.10 the discrepancy is a factor of about 1.5-2.5. This is acceptable.

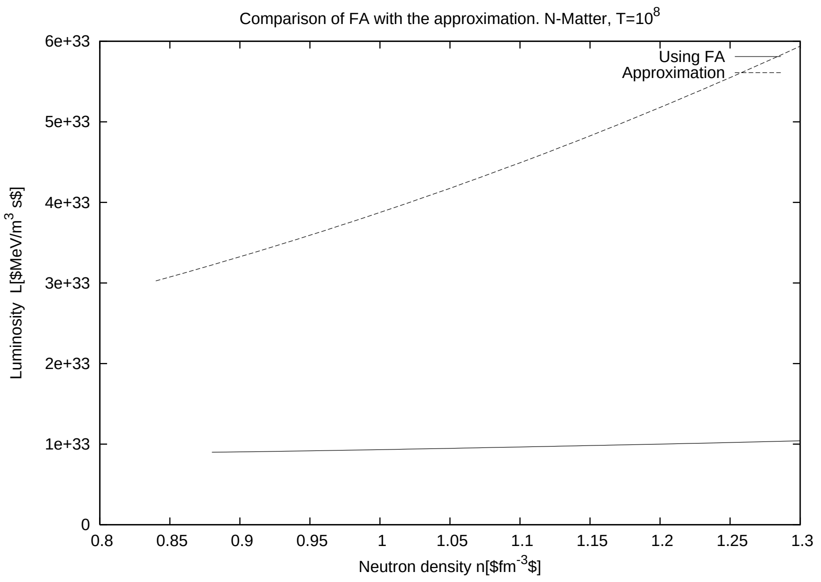

Figure 5.3: Comparison between the results from FA and the Approximations given in Eq. 5.1. Temperature $T = 10^8$, using the EoS for nuclear matter.

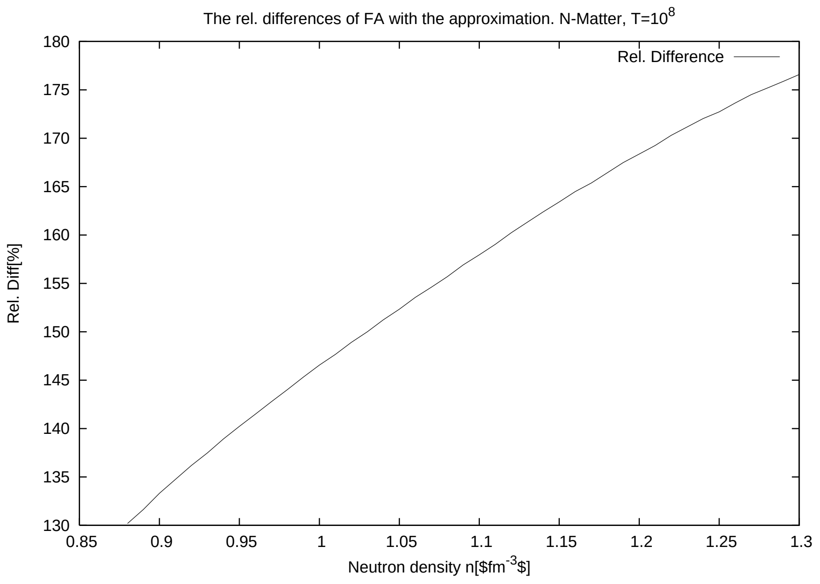

Figure 5.4: Difference between the results from FA and the Approximations given in Eq. 5.1. Temperature $T = 10^8$, using the EoS for nuclear matter.

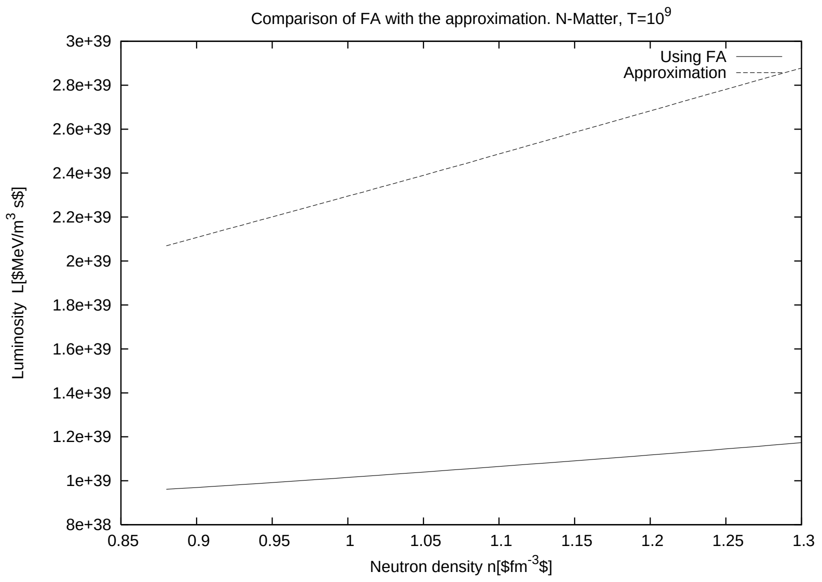

Figure 5.5: Comparison between the results from FA and the Approximations given in Eq. 5.1. Temperature $T = 10^9$, using the EoS for nuclear matter.

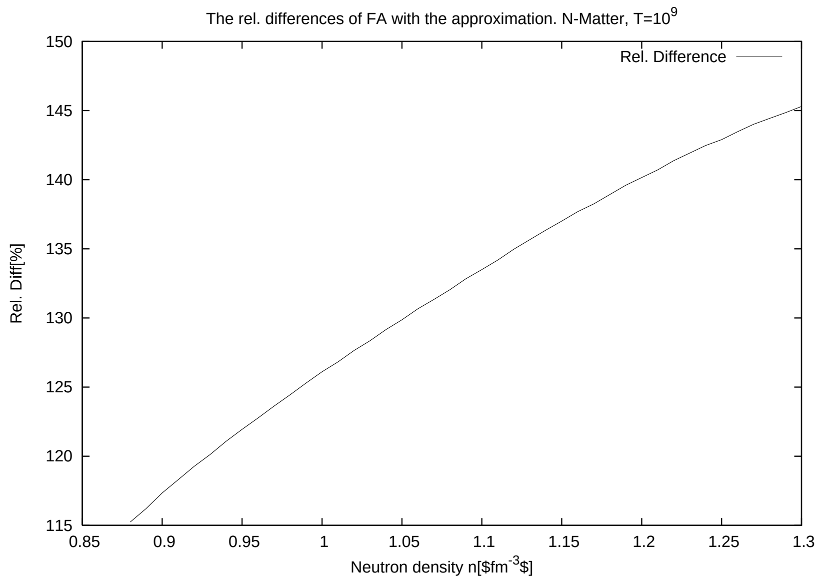

Figure 5.6: Difference between the results from FA and the Approximations given in Eq. 5.1. Temperature $T = 10^9$, using the EoS for nuclear matter.

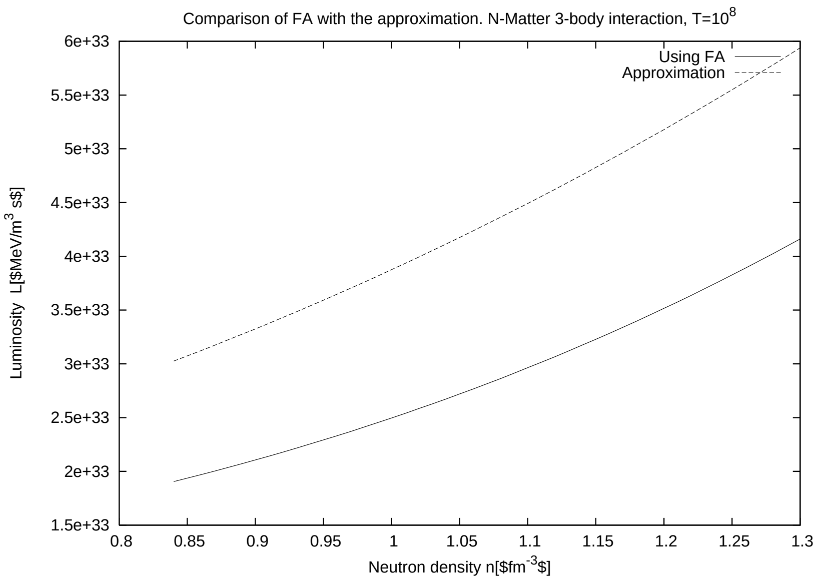

Figure 5.7: Comparison between the results from FA and the Approximations given in Eq. 5.1. Temperature T=10^8, using the EoS for nuclear matter with 3-body interactions.

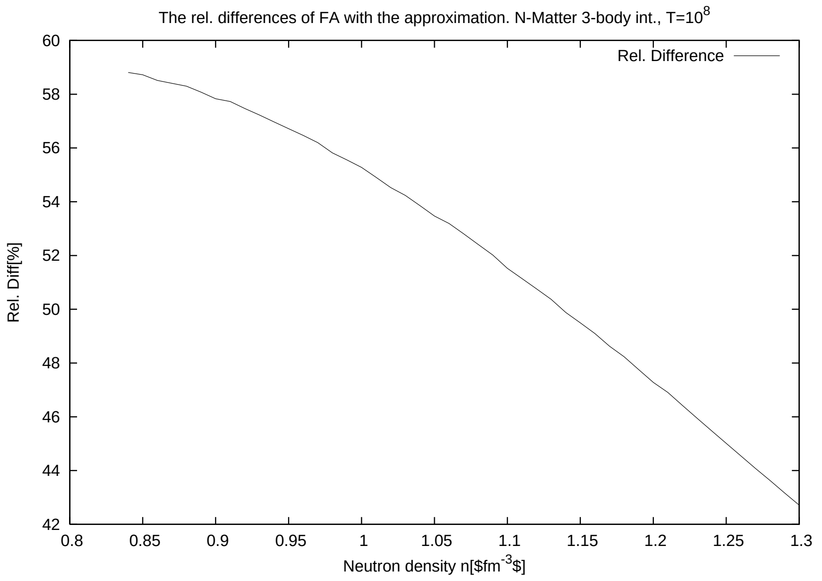

Figure 5.8: Difference between the results from FA and the Approximations given in Eq. 5.1. Temperature $T = 10^8$, using the EoS for nuclear matter with 3-body interactions.

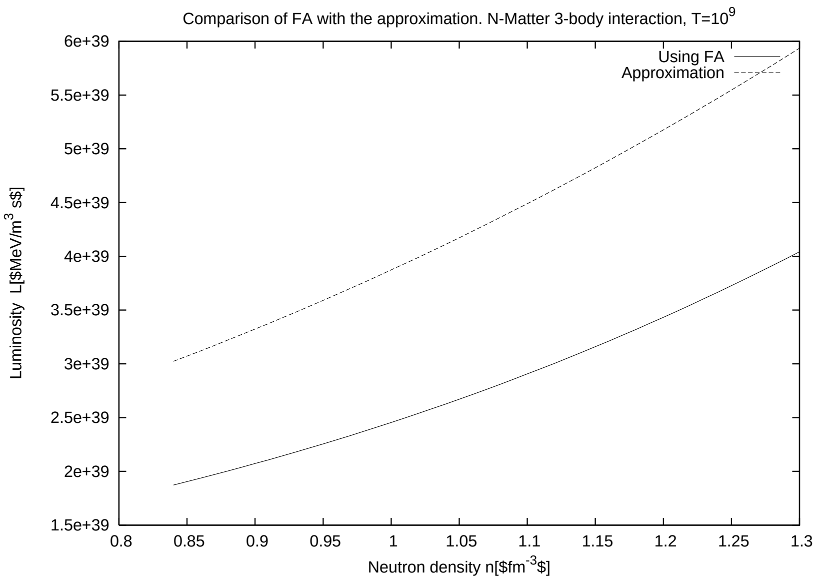

Figure 5.9: Comparison between the results from FA and the Approximations given in Eq. 5.1. Temperature T=10.8, using the EoS for nuclear matter with 3-body interactions.

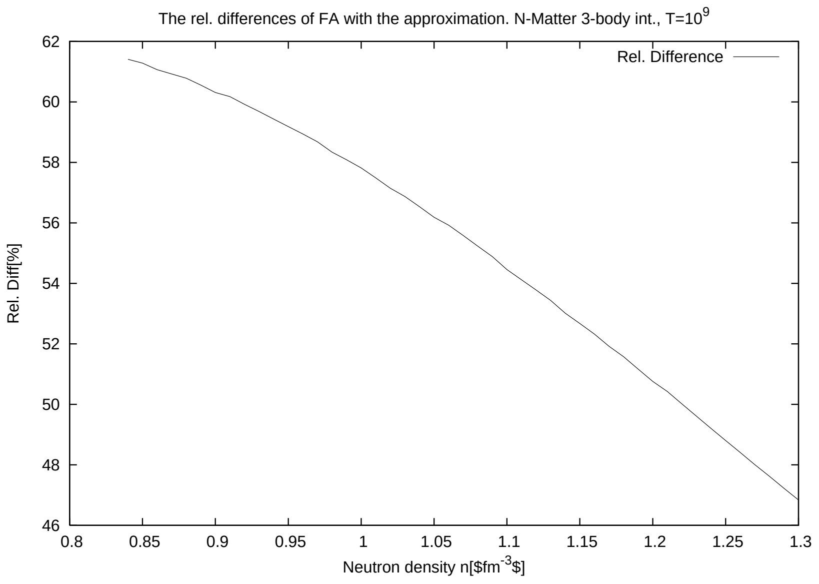

Figure 5.10: Difference between the results from FA and the Approximations given in Eq. 5.1. Temperature T=10.7, using the EoS for nuclear matter with 3-body interactions.

5.3 Modified URCA process comparison

Unfortunately at the time of the deadline of this thesis these results was not yet obtained. My colleague, Sutharsan Arumugam, has studied a similar process, Neutrino Pair Bremsstrahlung, in his thesis "Cooling of Neutron Stars" [1].

Chapter 6

Concluding remarks

The main goal of this thesis has been to develop software to calculate the Feynman amplitude of a arbitrary process within a neutron star. Where the main focus has been on the URCA processes. The URCA processes are particularly important because they are the main source of neutrino emission, and thereby give the biggest contribution to the luminosity in the neutron star.

As we have seen in chapter 5 the program developed, Feynman Amplitudes(FA), can be used successfully to calculate the emissivity of different processes in the neutron star. It is important to note that the calculations used in this thesis are just tests, but are close to the most accurate calculations done of these processes. We are now in a position to improve on the calculational scheme of Maxwell and Friman [6] which has been the used for almost 30 years. It is clearly time for an improvement! In this regard we can employ self-consistent calculational schemes of the nucleon-nucleon interaction taking medium effect into consideration. This would give more accurate results and thereby a better understanding of the underlying nuclear physics. An example of a such calculational scheme is outlined in an article by C. J. Horowitz and Brian D. Serot [12]. But in general, by using FA we can use any interaction matrix we want. Any one-boson interaction can be described by the following interaction matrix, parameterized by the Fermi invariant γ-matrices.

$$V = \bar{u}(\mathbf{p}_1{}')\bar{u}(\mathbf{p}_2{}')\left(\sum_{j=1}^{5} v_j(\mathbf{k})F_j\right)u(\mathbf{p}_1)u(\mathbf{p}_2). \tag{6.1}$$

Here v_j is the parameterization values and F_j $(j = 1, ..., 5)$ denote the Fermi invariants defined as:

$$S = 1^{(1)}1^{(2)}, \quad V = \gamma_\mu^{(1)}\gamma_\mu^{(2)}, \quad T = \frac{1}{2}\sigma_{\mu\nu}^{(1)}\sigma_{\mu\nu}^{(2)}, \tag{6.2}$$

$$A = i\gamma_5^{(1)}\gamma_\mu^{(1)} i\gamma_5^{(2)}\gamma_\mu^{(2)}, \quad P = \gamma_5^{(1)}\gamma_5^{(2)}, \tag{6.3}$$

FA is made to be completely generic and can be used to calculate any Feynman amplitude. So once a computational scheme has been established for a given process it is very easy to study similar processes. Processes of interest might be URCA processes involving hyperons.

$$\begin{aligned} \Lambda &\rightarrow p + e^- + \bar{\nu}_e \\ \Sigma^- &\rightarrow n + e^- + \bar{\nu}_e \\ \Sigma^- &\rightarrow \Lambda + e^- + \bar{\nu}_e \end{aligned}$$

or processes involving Δ-isobars. But in theory FA could be used to study any process using any interaction.

FA is still a work in progress, as mentioned in chapter 4 it is not properly optimized yet, lots of improvements could be done in this aspect. Other work for the future would be to automate the integrations routines over the phase-space. This is no simple task, but once done would improve the efficiency in which computuational schemes could be implemented.

In summary, numerical simulations of the processes inside the neutron star has the potential to give a better understanding of nuclear physics, by making a better connection between the EoS and the astrophysical observables. It is now possible to make a self-consistent calculation where you can insert matter effects directly into the nucleon-nucleon interaction. This has not been done properly before, and should lead to new insights. The field of neutron star physics is in an exciting phase and the years to come could bring some important discoveries, both in neutron star physics and in nuclear physics in general.

Appendix A

Constants Cheat-Sheet

In this work I will use natural units most of the time.

$$\hbar = c = 1$$

I vil only revert back to c.g.s units to compare with other results or when otherwise favourable
Unless spescified otherwise units of energy will be in MeV.

A.1 Constants

$$\begin{aligned} c &= 299792458 \quad \text{m/s} \\ \hbar &= 6.58211e-22 \quad \text{MeV} \cdot s \\ k_B &= 8.617343e-11 \quad \text{MeV/kelvin} \end{aligned}$$

Appendix B

The Dirac Equation

In this appendix we introduce the Dirac equation and derive the main results relating to this equation. In evaluating the Feynman diagrams we are often dealing with polarization sums and traces of γ-matrices. The very powerful and elegant methods for performing these polarization sums are to be worked out here. Moreover, we develop some useful techniques for calculating the traces of products of γ-matrices.

B.1 The Dirac Equation

The Dirac equation for particles of rest mass m can be written

$$i\hbar \frac{\partial \psi(x)}{\partial t} = \left[c\boldsymbol{\alpha} \cdot (-i\hbar \boldsymbol{\nabla}) + \beta mc^2 \right] \psi(x) \tag{B.1}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ and β are 4×4 Hermitian matrices satisfying

$$[\alpha_i, \alpha_j]_+ = 2\delta_{ij}, \qquad [\alpha_i, \beta]_+ = 0, \qquad \beta^2 = 1, \qquad i, j = 1, 2, 3. \tag{B.2}$$

With

$$\gamma^0 = \beta, \qquad \gamma^i = \beta\alpha_i \tag{B.3}$$

the Dirac equation becomes

$$(i\hbar\gamma^\mu \partial_\mu - mc)\, \psi(x) = 0 \tag{B.4}$$

with the 4×4 Dirac matrices γ^μ, $\mu = 0, ..., 3$, satisfying the anticommutation relations

$$[\gamma^\mu, \gamma^\nu]_+ = 2g^{\mu\nu} \tag{B.5}$$

and the Hermiticity conditions $\gamma^{0\dagger} = \gamma^0$ and $\gamma^{j\dagger} = -\gamma^j$ for $j = 1, 2, 3$, which can be combined into

$$\gamma^{\mu\dagger} = \gamma^0 \gamma^\mu \gamma^0 \tag{B.6}$$

Note that we express the partial derivative in Eq. (B.4) in shorthand by $\partial_\mu = \partial/\partial x^\mu$.

Correspondingly, $\psi(x)$ is a spinor wavefunction with four components $\psi_\alpha(x)$, $\alpha = 1, ..., 4$. The adjoint field $\bar{\psi}(x)$ is defined by

$$\bar{\psi}(x) = \psi^\dagger(x)\gamma^0 \tag{B.7}$$

and satisfies the adjoint Dirac equation

$$(i\hbar\partial_\mu\gamma^\mu + mc)\,\bar{\psi}(x) = 0 \tag{B.8}$$

The Dirac equations (B.4) and (B.8) can be derived from the Lagrangian density

$$\mathscr{L} = c\bar{\psi}(x)\,(i\hbar\gamma^\mu\partial_\mu - mc)\,\psi(x) \tag{B.9}$$

to which we may let act by the Euler-Lagrange equation

$$\frac{\partial\mathscr{L}}{\partial\psi(x)} - \partial_\mu\left(\frac{\partial\mathscr{L}}{\partial(\partial_\mu\psi(x))}\right) = 0 \tag{B.10}$$

A fifth anticommuting γ-matrix is defined by

$$\gamma^5 \equiv i\gamma^0\gamma^1\gamma^2\gamma^3, \tag{B.11}$$

and γ^5 has the properties

$$[\gamma^\mu, \gamma^5]_+ = 0, \qquad \left(\gamma^5\right)^2 = 1, \qquad \gamma^{5\dagger} = \gamma^5. \tag{B.12}$$

We also define the matrix γ_5 through

$$\gamma_5 \equiv \frac{i}{4!}\epsilon_{\rho\mu\sigma\nu}\gamma^\rho\gamma^\mu\gamma^\sigma\gamma^\nu = \gamma^5, \tag{B.13}$$

where the totally antisymmetric Levi-Civita symbol $\epsilon_{\rho\mu\sigma\nu}$ is equal to $+1$ if (ρ, μ, σ, ν) is an even permutation of $(0, 1, 2, 3)$, -1 if it is an odd permutation, and 0 if any index is repeated.

There exists several representations of γ-matrices. One of the most frequently used representations for γ-matrices is the original Dirac representation. In terms of the Pauli 2×2 spin matrices

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{B.14}$$

the Dirac matrices can in this representation be written as

$$\gamma^0 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad \gamma^i = \begin{pmatrix} 0 & \sigma^i \\ -\sigma^i & 0 \end{pmatrix}, \qquad i = 1, 2, 3, \tag{B.15}$$

and

$$\gamma^5 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{B.16}$$

In fact, these five γ-matrices form the Dirac basis and this, in turn, are appropriate for acting on Dirac spinors written in this basis.

The γ-matrices act like a vector, and their contraction with a vector acts like a scalar. So, it is common to write a covariant four-vector using Feynman slash notation, defined by

$$\not{p} \equiv p_\mu \gamma^\mu \tag{B.17}$$

B.2 Solutions to the Dirac Equation

The Dirac equation (B.4) possesses free-particle solutions

$$\psi^+(x) = K^+ u(p,s)\mathbf{e}^{-ipx/\hbar}, \qquad \psi^-(x) = K^- v(p,s)\mathbf{e}^{ipx/\hbar} \tag{B.18}$$

where $K^\pm$ are normalization factors, given by

$$K^\pm = \sqrt{\frac{c}{2EV}} \tag{B.19}$$

Note that $p = (E/c, \mathbf{p})$, $E = \sqrt{|\mathbf{p}|^2c^2 + m^2c^4}$ and $px = Et - \mathbf{p}\cdot\mathbf{x}$. By putting the solutions (B.18) into the Dirac equation (B.4), we obtain the momentum space Dirac equations

$$(\not{p} - mc)u(p,s) = 0, \qquad (\not{p} + mc)v(p,s) = 0 \tag{B.20}$$

These has two positive energy solutions

$$u(p,s) = N^+ \begin{pmatrix} \chi_s \\ \frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\chi_s \end{pmatrix}, \qquad s = 1,2, \tag{B.21}$$

and two negative energy solutions

$$v(p,s) = N^- \begin{pmatrix} \frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\chi'_s \\ \chi'_s \end{pmatrix}, \qquad s = 1,2, \tag{B.22}$$

which are then interpreted as positive energy antiparticle solutions. Here $N^\pm$ are some other normalization factors, defined by

$$N^\pm = \sqrt{\frac{E + mc^2}{c}} \tag{B.23}$$

The index $s = 1, 2$ signifies the two possible eigenstates of spin of a spin-1/2 particle: spin up and spin down. These two spin states are represented by the Pauli spinors

$$\chi_1 \equiv \chi'_2 \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \chi_2 \equiv \chi'_1 \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{B.24}$$

which are mutually orthogonal, and consequently the Dirac spinors $u(p, s)$ and $v(p, s)$ involving these two states become also orthogonal. Furthermore, we normalize the spinors $u(p, s)$ and $v(p, s)$ so that

$$u^\dagger(p, s)u(p, s) = v^\dagger(p, s)v(p, s) = \frac{2E}{c} \tag{B.25}$$

They then satisfy the orthonormality relations

$$\begin{aligned} u^\dagger(p, s)u(p, s') = v^\dagger(p, s)v(p, s') &= \frac{2E}{c}\delta_{ss'} \\ u^\dagger(p, s)v(-p, s') &= 0 \end{aligned} \tag{B.26}$$

It might seem that $u(p, s = 1)$ describes a particle with spin up, $u(p, s = 2)$ a particle with spin down, and so on, but this is not quite the case. For Dirac particles the spin matrices are

$$\mathbf{S} = \frac{\hbar}{2}\boldsymbol{\Sigma}, \qquad \text{with } \boldsymbol{\Sigma} \equiv \begin{pmatrix} \boldsymbol{\sigma} & 0 \\ 0 & \boldsymbol{\sigma} \end{pmatrix} \tag{B.27}$$

and the Dirac spinors $u(p, s)$ and $v(p, s)$ are not eigenstates of Σ_z. As a matter of fact, it is impossible to construct spinors that satisfy Eq. (B.20) and are, at the same time, eigenstates of S_z. The reason is that $\mathbf{S}$ by itself is not a conserved quantity; only the total angular momentum, $\mathbf{L} + \mathbf{S}$, is conserved here. However, if we orient the z-axis so that it points along the direction of motion (in which case $p_x = p_y = 0$) then $u(p, s)$ and $v(p, s)$ are eigenspinors of S_z.

B.3 Completeness Relations

In most practical situations, the interacting particles are unpolarized, i.e. the initial and final spin orientations are arbitrary. It then becomes necessary to average and sum over polarization states of initial and final particles respectively. For this purpose, we here show explicitly how to perform the summation over spin states.

To start with, we take the transpose conjugate of the positive energy spinor $u(p,s)$ (B.21):

$$\bar{u}(p,s) = u^\dagger(p,s)\gamma^0 = N^{+*}\begin{pmatrix}\chi_s^\dagger & -\chi_s^\dagger\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\end{pmatrix} \tag{B.28}$$

We are ready now to sum over the two spin states of a particle:

$$\begin{aligned}\sum_s u(p,s)\bar{u}(p,s) &= \sum_s |N^+|^2\begin{pmatrix}\chi_s\\ \frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\chi_s\end{pmatrix}\begin{pmatrix}\chi_s^\dagger & -\chi_s^\dagger\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\end{pmatrix}\\ &= \frac{E+mc^2}{c}\begin{pmatrix}1 & -\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\\ \frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2} & -\frac{c^2(\boldsymbol{\sigma}\cdot\mathbf{p})^2}{(E+mc^2)^2}\end{pmatrix}\end{aligned} \tag{B.29}$$

where in the last step we have applied $\chi_s\chi_s^\dagger = 1$. From the last equation, we notice that

$$(\boldsymbol{\sigma}\cdot\mathbf{p})^2 = \mathbf{p}^2 = \frac{(E+mc^2)(E-mc^2)}{c^2} \tag{B.30}$$

where we have used the properties $\sigma_i^2 = 1$ and $[\sigma_i,\sigma_j]_+ = 2\delta_{ij}$.

On substituting Eq. (B.30) in Eq. (B.29), we obtain

$$\begin{aligned}\sum_s u(p,s)\bar{u}(p,s) &= \begin{pmatrix}\frac{E+mc^2}{c} & -\boldsymbol{\sigma}\cdot\mathbf{p}\\ \boldsymbol{\sigma}\cdot\mathbf{p} & -\frac{E-mc^2}{c}\end{pmatrix}\\ &= \gamma^0\frac{E}{c} - \boldsymbol{\gamma}\cdot\mathbf{p} + \mathbf{1}\cdot mc\\ &= \gamma^\mu p_\mu + mc \equiv \not{p} + mc\end{aligned} \tag{B.31}$$

We follow the same lines of reasoning in order to show the summation over spin states of an antiparticle. Firstly, we take the adjoint of the negative energy spinor $v(p,s)$ (B.22):

$$\bar{v}(p,s) = v^\dagger(p,s)\gamma^0 = N^{-*}\begin{pmatrix}\chi_s^\dagger\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2} & -\chi_s^\dagger\end{pmatrix} \tag{B.32}$$

We now go on to sum over the two spin states of an antiparticle:

$$\begin{aligned}\sum_s v(p,s)\bar{v}(p,s) &= \sum_s |N^-|^2\begin{pmatrix}\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\chi_s\\ \chi_s\end{pmatrix}\begin{pmatrix}\chi_s^\dagger\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2} & -\chi_s^\dagger\end{pmatrix}\\ &= \frac{E+mc^2}{c}\begin{pmatrix}\frac{c^2(\boldsymbol{\sigma}\cdot\mathbf{p})^2}{(E+mc^2)^2} & -\frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2}\\ \frac{c\boldsymbol{\sigma}\cdot\mathbf{p}}{E+mc^2} & -1\end{pmatrix}\end{aligned} \tag{B.33}$$

where the last step follows by $\chi_s \chi_s^\dagger = 1$. On proceeding the spin sum, we make use of Eq. (B.30)

$$\begin{aligned}\sum_s v(p,s)\bar{v}(p,s) &= \begin{pmatrix} \frac{E-mc^2}{c} & -\boldsymbol{\sigma}\cdot\mathbf{p} \\ \boldsymbol{\sigma}\cdot\mathbf{p} & -\frac{E+mc^2}{c} \end{pmatrix} \\ &= \gamma^0 \frac{E}{c} - \boldsymbol{\gamma}\cdot\mathbf{p} - \mathbf{1}\cdot mc \\ &= \gamma^\mu p_\mu - mc \equiv \not{p} - mc \end{aligned} \tag{B.34}$$

B.4 Casimir's Trick

The summations over polarization states are easily performed using the following trick. The Casimir's trick reduces everything down to a problem of calculating the trace of some complicated product of γ-matrices. In contrast to the spin sums from the last section, we now deal with summands of the form $\bar{u}(p,s)\mathcal{A}u(p,s)$, in which the spin states are to be summed up. Here $\mathcal{A}$ is some product of γ-matrices. First we write the summand $\bar{u}(p,s)\mathcal{A}u(p,s)$ with explicit spinor indices $i,j = 1,2,3,4$:

$$\begin{aligned}\mathbf{Tr}(\widetilde{\mathcal{A}}) = \sum_s \bar{u}(p,s)\mathcal{A}u(p,s) &= \sum_s \sum_i \sum_j \bar{u}_i(p,s)\mathcal{A}_{ij}u_j(p,s) \\ &= \sum_i \sum_j \mathcal{A}_{ij} \sum_s u_j(p,s)\bar{u}_i(p,s)\end{aligned} \tag{B.35}$$

where in the last step we have moved $\bar{u}_i(p,s)$ behind $u_j(p,s)$ ($\bar{u}_i(p,s)$ is just a number, element of vector $\bar{u}(p,s)$, so it commutes with everything).

Using the completeness relation (B.31), we have

$$\begin{aligned}\mathbf{Tr}(\widetilde{\mathcal{A}}) = \sum_s \bar{u}(p,s)\mathcal{A}u(p,s) &= \sum_i \sum_j \mathcal{A}_{ij}\left(\not{p}+m\right)_{ji} \\ &= \sum_i \sum_j \left(\not{p}+m\right)_{ji}\mathcal{A}_{ij} = \mathbf{Tr}\left[\left(\not{p}+m\right)\mathcal{A}\right]\end{aligned} \tag{B.36}$$

Notice that there are no spinors left; once we do the summation over spins, it all reduces to matrix multiplication and taking the trace. If either positive energy spinor $u(p,s)$ (in Eq. (B.36)) is replaced by a negative energy spinor $v(p,s)$, the corresponding mass on the right-hand side switches sign.

B.5 Contraction Identities and Traces

The manipulation of expressions involving γ-matrices is often greatly facilitated by the use of the following algebraic identities, which follow easily from the anticommutation relations (B.5). In terms of the metric

$$g^{\mu\nu} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{B.37}$$

we have:

a) $\gamma^\mu \gamma_\mu = \frac{1}{2} g_{\mu\nu}\big(\gamma^\mu\gamma^\nu + \gamma^\nu\gamma^\mu\big) = g_{\mu\nu}g^{\mu\nu} = 4$

b) $\gamma^\mu\gamma^\alpha\gamma_\mu = \big(2g^{\mu\alpha} - \gamma^\alpha\gamma^\mu\big)\gamma_\mu = 2\gamma^\alpha - 4\gamma^\alpha = -2\gamma^\alpha$

c) $\gamma^\mu\gamma^\alpha\gamma^\beta\gamma_\mu = \big(2g^{\mu\alpha} - \gamma^\alpha\gamma^\mu\big)\gamma^\beta\gamma_\mu = 2\gamma^\beta\gamma^\alpha + 2\gamma^\alpha\gamma^\beta = 4g^{\alpha\beta}$

d) $\gamma^\mu\gamma^\alpha\gamma^\beta\gamma^\eta\gamma_\mu = \big(2g^{\mu\alpha} - \gamma^\alpha\gamma^\mu\big)\gamma^\beta\gamma^\eta\gamma_\mu = 2\gamma^\beta\gamma^\eta\gamma^\alpha - 4\gamma^\alpha g^{\beta\eta} = -2\gamma^\eta\gamma^\beta\gamma^\alpha$

e) $\gamma^\mu\gamma^\alpha\gamma^\beta\gamma^\eta\gamma^\delta\gamma_\mu = \big(2g^{\mu\alpha} - \gamma^\alpha\gamma^\mu\big)\gamma^\beta\gamma^\eta\gamma^\delta\gamma_\mu = 2\big(\gamma^\beta\gamma^\eta\gamma^\delta\gamma^\alpha + \gamma^\alpha\gamma^\delta\gamma^\eta\gamma^\beta\big)$

The completely antisymmetric Levi-Civita symbol $\epsilon_{\rho\mu\sigma\nu}$, introduced in Eq. (B.13), satisfies the following contraction identity:

$$\epsilon^{\alpha\mu\beta\nu}\epsilon_{\rho\mu\sigma\nu} = -2\big(g^\alpha_\rho g^\beta_\sigma - g^\alpha_\sigma g^\beta_\rho\big) \tag{B.38}$$

We next list some rules and relations which are extremely useful in evaluating the trace of a product of γ-matrices.

1. For any two $n \times n$ matrices A and B, and any scalar c, we have

$$\begin{aligned} \mathbf{Tr}(A+B) &= \mathbf{Tr}(A) + \mathbf{Tr}(B) \\ \mathbf{Tr}(AB) &= \mathbf{Tr}(BA) \\ \mathbf{Tr}(cA) &= c\mathbf{Tr}(A) \end{aligned} \tag{B.39}$$

2. The trace of the product of an odd number of γ-matrices is zero. If $(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu)$ contains an odd number of γ-matrices, then

$$\begin{aligned} \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\big) &= \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\gamma^5\gamma^5\big) \\ &= -\mathbf{Tr}\big(\gamma^5\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\gamma^5\big) \end{aligned} \tag{B.40}$$

where the last step follows by moving the γ^5-matrix in the front of the γ-matrices. By the cyclic property of the traces (Eq. (B.39)), we realize that

$$\begin{aligned} \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\big) &= -\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\gamma^5\gamma^5\big) \\ &= -\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta...\gamma^\mu\gamma^\nu\big) = 0 \end{aligned} \tag{B.41}$$

3. For a product of an even number of γ-matrices:

$$\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\big) = \mathbf{Tr}\big(2g^{\alpha\beta} - \gamma^\beta\gamma^\alpha\big) = 2g^{\alpha\beta}\mathbf{Tr}(1) - \mathbf{Tr}\big(\gamma^\beta\gamma^\alpha\big) = 8g^{\alpha\beta} - \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\big)$$

with $\mathbf{Tr}(1) = 4$. Moving the last term over to the left-hand side results in

$$\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\big) = 4g^{\alpha\beta} \tag{B.42}$$

Correspondingly

$$\begin{aligned}
\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\gamma^\rho\gamma^\sigma\big) &= \mathbf{Tr}\big[\gamma^\alpha\gamma^\beta\big(2g^{\rho\sigma} - \gamma^\sigma\gamma^\rho\big)\big] \\
&= 2g^{\rho\sigma}\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\big) - \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\gamma^\sigma\gamma^\rho\big) \\
&= 8g^{\rho\sigma}g^{\alpha\beta} - \mathbf{Tr}\big[\gamma^\alpha\big(2g^{\beta\sigma} - \gamma^\sigma\gamma^\beta\big)\gamma^\rho\big] \\
&= 8g^{\rho\sigma}g^{\alpha\beta} - 2g^{\beta\sigma}\mathbf{Tr}\big(\gamma^\alpha\gamma^\rho\big) + \mathbf{Tr}\big(\gamma^\alpha\gamma^\sigma\gamma^\beta\gamma^\rho\big) \\
&= 8g^{\rho\sigma}g^{\alpha\beta} - 8g^{\beta\sigma}g^{\alpha\rho} + \mathbf{Tr}\big[\big(2g^{\alpha\sigma} - \gamma^\sigma\gamma^\alpha\big)\gamma^\beta\gamma^\rho\big] \\
&= 8g^{\rho\sigma}g^{\alpha\beta} - 8g^{\beta\sigma}g^{\alpha\rho} + 2g^{\alpha\sigma}\mathbf{Tr}\big(\gamma^\beta\gamma^\rho\big) - \mathbf{Tr}\big(\gamma^\sigma\gamma^\alpha\gamma^\beta\gamma^\rho\big) \\
&= 8g^{\rho\sigma}g^{\alpha\beta} - 8g^{\beta\sigma}g^{\alpha\rho} + 8g^{\alpha\sigma}g^{\beta\rho} - \mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\gamma^\rho\gamma^\sigma\big)
\end{aligned}$$

Rearranging the last equation provides

$$\mathbf{Tr}\big(\gamma^\alpha\gamma^\beta\gamma^\rho\gamma^\sigma\big) = 4\big(g^{\alpha\beta}g^{\rho\sigma} - g^{\alpha\rho}g^{\beta\sigma} + g^{\alpha\sigma}g^{\beta\rho}\big) \tag{B.43}$$

4. Since γ^5 is the product of an even number of γ-matrices, it follows that $\mathbf{Tr}(\gamma^5\gamma^\alpha) = \mathbf{Tr}(\gamma^5\gamma^\alpha\gamma^\beta\gamma^\rho) = 0$. When γ^5 is multiplied by an even number of γ's, we find

$$\begin{aligned}
&\mathbf{Tr}\big(\gamma^5\big) = \mathbf{Tr}\big(\gamma^5\gamma^\alpha\gamma^\beta\big) = 0 \\
&\mathbf{Tr}\big(\gamma^5\gamma^\alpha\gamma^\beta\gamma^\rho\gamma^\sigma\big) = -4i\epsilon^{\alpha\beta\rho\sigma}
\end{aligned} \tag{B.44}$$

Appendix C

Code

C.1 Amplitude

C.1.1 amplitude.h

```
#ifndef AMPS
#define AMPS

#include <bfft.h>
#include <amplitude.h>

Amplitude* NeutronDecayElectron();
Amplitude* NeutronDecayMuon();
Amplitude* ModifiedUrca();

#endif
```

C.1.2 amplitude.cpp

```
#include "amplitudes.h"

Amplitude* NeutronDecayElectron()
{
    /* Neutron Beta Decay with Electrons*/

    // Make an amplitude
    Amplitude* amp = new Amplitude();

    // Make particles
    Particle* p1 = new Neutron();
    Particle* p2 = new AntiNeutrino_e();
    Particle* p3 = new Proton();
    Particle* p4 = new Electron();
    Boson* wneg = new Wneg();

    //Set the number of incoming spinstates
    amp->setSpinFactor(2);

    // Make elements
    Element* ubar1 = new Ubar(p3);
    Element* v1 = new Vertex_Weak(1.0,1.0,1);
    Element* u1 = new U(p1);
    Element* bprop = new BosonPropNonrel(wneg,1,2);
    Element* ubar2 = new Ubar(p4);
    Element* v2 = new Vertex_Weak(2);
    Element* u2 = new U(p2);

    // Add it all to the amplitude
    amp->add(p1);
    amp->add(p2);
    amp->add(p3);
    amp->add(p4);

    amp->add(ubar1);
    amp->add(v1);
    amp->add(u1);
    amp->add(bprop);
    amp->add(ubar2);
    amp->add(v2);
    amp->add(u2);
    return amp;
}

Amplitude* NeutronDecayMuon()
{
    /* Neutron Beta Decay with Muons */

    // Make an amplitude
    Amplitude* amp = new Amplitude();

    // Make particles
    Particle* p1 = new Neutron();
    Particle* p2 = new AntiNeutrino_m();
    Particle* p3 = new Proton();
    Particle* p4 = new Muon();
    Boson* wneg = new Wneg();

    //Set the number of incoming spinstates
    amp->setSpinFactor(2);

    // Make elements
    Element* ubar1 = new Ubar(p3);
    Element* v1 = new Vertex_Weak(1.0,1.25,1);
    Element* u1 = new U(p1);
    Element* bprop = new BosonPropNonrel(wneg,1,2);
    Element* ubar2 = new Ubar(p4);
    Element* v2 = new Vertex_Weak(2);
    Element* u2 = new U(p2);

    // Add it all to the amplitude
    amp->add(p1);
    amp->add(p2);
    amp->add(p3);
    amp->add(p4);

    amp->add(ubar1);
    amp->add(v1);
    amp->add(u1);
    amp->add(bprop);
    amp->add(ubar2);
    amp->add(v2);
    amp->add(u2);
    return amp;
}


Amplitude* ModifiedUrca()
{
    // Make an ampletude
    Ampletude* amp = new Ampletude();

    Particle* p1 = new Neutron();
    Particle* p2 = new Neutron();
    Particle* p3 = new Neutron();
    Particle* p4 = new Proton();
    Particle* p5 = new AntiNeutrino_e();
    Particle* p6 = new Electron();
    Boson* wneg = new Wneg();
    Boson* pi0 = new Pi0();
    Fermion* neutron = new Neutron();

    // Make elements
    Element* ubar1 = new Ubar(p3);
    Element* v1 = new Vertex_Pion();
    Element* u1 = new U(p1);

    Element* piprop = new ScalarProp(pi0);

    Element* ubar2 = new Ubar(p4);
    Element* v3 = new Vertex_Weak(1,1.25,1);
    Element* fprop = new FermionProp(neutron);
    Element* v2 = new Vertex_Pion();
    Element* u2 = new U(p2);

    Element* bprop = new BosonPropNonrel(wneg,1,2);

    Element* ubar3 = new Ubar(p6);
    Element* v4 = new Vertex_Weak(2);
    Element* u3 = new U(p5);

    // Add it to the ampletude
    amp->add(ubar1);
    amp->add(v1);
    amp->add(u1);
```

```
      amp->add(piprop);

      amp->add(ubar2);
      amp->add(v3);
      amp->add(fprop);
      amp->add(v2);
      amp->add(u2);

      amp->add(bprop);

      amp->add(ubar3);
      amp->add(v4);
      amp->add(u3);

   return amp;
}
```

C.2 Neutron decay

C.2.1 neutrondecay.h

```
#ifndef ND
#define ND
#include <mpi.h>
#include "lib.h"
#include <fstream>
#include <iostream>

#include <bfft.h>
#include <amplitude.h>
#include "amplitudes.h"

struct Int_params
{
    double min, max;
    int steps;
};

class Parameters
{
    public:
    int size, my_rank;
    const char* outfile;
    double T,p;
    Array<Int_params,1> dim;
    Parameters(int d) :dim(d) {};
    ~Parameters();
};

Amplitude* NeutronDecayElectron();
double weakdecayrate(Amplitude* amp,Parameters* params);
Parameters* setParamerters(Amplitude* amp);
double getAnalyticalDecay(Amplitude* amp, double C_A);

#endif
```

C.2.2 neutrondecay.cpp

```
#include "neutrondecay.h"


int main(int argc, char* argv[])
{
  Amplitude* amp = NeutronDecayElectron();

  Parameters* params = setParamerters(amp);

  weakdecayrate(amp,params);
}

Parameters* setParamerters(Amplitude* amp)
{
  Parameters* params = new Parameters(2);

  double m_1,m_2,m_3,m_4;
  m_1 = amp->getParticle(1)->getMass();
  m_2 = amp->getParticle(2)->getMass();
  m_3 = amp->getParticle(3)->getMass();
  m_4 = amp->getParticle(4)->getMass();

  params->dim(0).steps = 20;

  params->dim(1).min = m_4;
  params->dim(1).max = m_1 - m_3;
  params->dim(1).steps = 20;

  params->outfile = "results.dat";

  return params;
}

double weakdecayrate(Amplitude* amp, Parameters* params )
{
  //Make variable names that are easier to handle.
  const char* outfile = params->outfile;

  Particle* p1 = amp->getParticle(1);
  Particle* p2 = amp->getParticle(2);
  Particle* p3 = amp->getParticle(3);
```

```
Particle* p4 = amp->getParticle(4);

double m_1, m_2, m_3, m_4;

m_1 = p1->getMass();
m_2 = p2->getMass();
m_3 = p3->getMass();
m_4 = p4->getMass();

//Declare variables
Impulse k1(3), k2(3), k3(3), k4(3);

double E1,E3;

double n, d, tmp;
double n_anal;

double p_2,p_3,p_4,theta;
double E2_min, E2_max, E2_sum;
double E4_min, E4_max, E4_sum;
int E4_steps, E2_steps;

double E2_sum_anal;
double E4_sum_anal;

//Set integration parameters
E2_steps = params->dim(0).steps;
E4_steps = params->dim(1).steps;

E4_min = params->dim(1).min;
E4_max = params->dim(1).max;

double *E4 = new double[E4_steps];
double *W_E4 = new double[E4_steps];
double *E2 = new double[E2_steps];
double *W_E2 = new double[E2_steps];
double *results = new double[E4_steps];
double *analytical_results = new double[E4_steps];
gauleg(E4_min, E4_max, E4, W_E4, E4_steps);

//We are using Neutron rest-frame
k1 = 0,0,0;
p1->setImpulse(k1);
```

```
E1 = p1->getEnergy();

E4_sum = 0;
E4_sum_anal = 0;
// Integrating over allowed Electron energy
for (int i = 0; i < E4_steps; i++)
{
  p_4 = p4->getAbsImpulse(E4[i]);
  //Fixing the electron along the Z-axis
  k4 = cartesian( p_4, 0, 0 );
  p4->setImpulse(k4);
  n = ( 0.5* (m_1*m_1 - m_3*m_3 + m_4*m_4) - m_1*E4[i] );
  d = (m_1 - E4[i] + p_4);
  E2_min = n/d;
  d = (m_1 - E4[i] - p_4);
  E2_max = n/d;

  gauleg(E2_min, E2_max, E2, W_E2, E2_steps);
  E2_sum = 0;
  E2_sum_anal = 0;
  //Integrating over the angle between the electron and neutrino
  for (int j = 0; j < E2_steps; j++ )
  {
    E3 = m_1 - E2[j] - E4[i];
    p_2 = p2->getAbsImpulse(E2[j]);
    p_3 = p3->getAbsImpulse(E3);

    theta = acos((p_3*p_3-p_2*p_2-p_4*p_4)/(2*p_2*p_4));

    k2 = cartesian(p_2, theta, 0);

    k3 = -( k2 + k4 );

    p2->setImpulse(k2);
    p3->setImpulse(k3);

    //Calculate the Feynman-amplitude for the current set
    //of impulses
    n =  amp->calculateSpinAvrage();
    n_anal = getAnalyticalDecay(amp,1.0);
    d =  pow(4*pi,3) * m_1 * hbar;

    E2_sum+= n/d * W_E2[j];
```

```
      E2_sum_anal+= n_anal/d * W_E2[j];
    }

    results[i] =  E2_sum;
    analytical_results[i] =  E2_sum_anal;

    E4_sum += W_E4[i]   * E2_sum;
    E4_sum_anal += W_E4[i]   * E2_sum_anal;
  }

  ofstream output;
  output.open(outfile);
  for (int i = 0; i < E4_steps;i++)
  {
    output << E4[i] << "␣"
      << results[i] << "␣"
      << analytical_results[i] << endl;
  }

  cout << "Decay-rate␣Numerical=" <<  1.0/E4_sum << endl;
  cout << "Decay-rate␣Analytical=" <<  1.0/E4_sum_anal << endl;
  output.close();

  delete [] E4;
  delete [] W_E4;
  delete [] E2;
  delete [] W_E2;
  delete [] results;
  delete [] analytical_results;

  return 1.0;
}

double getAnalyticalDecay(Amplitude* amp, double C_A)
{
  Particle* p1 = amp->getParticle(1);
  Particle* p2 = amp->getParticle(2);
  Particle* p3 = amp->getParticle(3);
  Particle* p4 = amp->getParticle(4);

  Tensor1 i1(4,1),i2(4,1),i3(4,1),i4(4,1);
  i1 = p1->getImpulseTensor();
  i2 = p2->getImpulseTensor();
  i3 = p3->getImpulseTensor();
  i4 = p4->getImpulseTensor();

  return 0.5 *pow(0.66/80403,4)
    * ( pow(1.0 + C_A,2) * contract(i1,i2) * contract(i3,i4)
    + pow(1.0 - C_A,2) * contract(i1,i4) * contract(i2,i3)
    - (1.0 - C_A*C_A) * p1->getMass() * p3->getMass() *contract(i2,i4));
}
```

C.3 Direct Urca

C.3.1 ducra.h

```
#ifndef DP
#define DP
#include <mpi.h>
#include "lib.h"
#include <fstream>
#include <iostream>

#include <bfft.h>
#include <amplitude.h>
#include "amplitudes.h"

struct Int_params
{
    double min, max;
    int steps;
};

class Parameters
{
   public:
    int size, my_rank;
    const char* outfile;
    double T,p;
    Array<Int_params,1> dim;
    Parameters(int d) :dim(d) {};
    ~Parameters();
};

double luminosity(Amplitude* amp, Parameters* params);
double* l3int(Amplitude* amp, Parameters* params,
    double E1_F, double E3_F, double E4_F, double beta);
double getAnalytcialL(double T, double m_1,double m_3a
     , double cp_4);
void readdatafile(Array<double,2>, int A, int B,
    const char* filename );
Parameters* setParamerters(Amplitude* amp);

#endif
```

C.3.2 ducra.cpp

```
#include "durca.h"

int main(int argc, char* argv[])
{
  MPI::Init (argc, argv);

  Amplitude* amp = NeutronDecayElectron();
  Parameters* params = setParamerters(amp);

  luminosity(amp,params);

  MPI::Finalize ();
}

Parameters* setParamerters(Amplitude* amp)
{
  //Todo: Read input from commandline or file.
  Parameters* params = new Parameters(3);

  params->size = MPI::COMM_WORLD.Get_size();
  params->my_rank = MPI::COMM_WORLD.Get_rank();

  double steps = 20;
  double treshold_min = -5;
  double treshold_max = 10 ;

  params->dim(0).min = treshold_min;
  params->dim(0).max = treshold_max;
  params->dim(0).steps = steps;

  params->dim(1).min = treshold_min;
  params->dim(1).max = treshold_max;
  params->dim(1).steps = steps;

  params->dim(2).min = treshold_min;
  params->dim(2).max = treshold_max;
  params->dim(2).steps = steps;

  params->outfile = "results.dat";

  return params;
```

```
}

//
//Generate the different data needed for the
//integration routine and loop over the different
//temeratures and densities specified.
//Todo: Read data from commandline or file.
double luminosity(Amplitude* amp, Parameters* params)
{
  Particle* p1 = amp->getParticle(1);
  Particle* p2 = amp->getParticle(2);
  Particle* p3 = amp->getParticle(3);
  Particle* p4 = amp->getParticle(4);

  double p_1_F, p_3_F, p_4_F;
  double E1_F, E3_F, E4_F;
  double m_1, m_3, m_4;

  double beta;
  int first_allowed_density_row ,rows_left;
  double *em_all;

  int rho_nr = 20;
  int T_nr = 2;
  int row = 129;
  int col = 8;

  Array<double,2> rho(row,col),cp(row,col);
  Array<double,1> T(2);
  rho, cp = 0;
  T = 1E8, 1E9;

  ofstream data;
  data.open(params->outfile);

  readdatafile( rho, row, col, "data/nuc_mat_rho.dat");
  readdatafile( cp, row, col, "data/nuc_mat_cp.dat");

  //Find at wich density URCA process is allowed
  //p_1_F <= p_3_F + p_4_F
  //We assume that that after a certain density the
  //process is always allowed.
  for (int i = 0; i < row; i++)
```

```
{
  p_1_F = hbarc * pow(rho(i,1)* 3 *pi*pi,1.0/3.0);
  p_3_F = hbarc * pow(rho(i,2)* 3 *pi*pi,1.0/3.0);
  p_4_F = hbarc * pow(rho(i,3)* 3 *pi*pi,1.0/3.0);

  if (p_1_F <= p_3_F + p_4_F )
  {
    first_allowed_density_row = i;
    rows_left = row - i;
    if (rows_left < rho_nr) {
      rho_nr = rows_left;
    }
    break;
  }

}

//Calculate the luminosity for differnet temeratures
//and densities.
for (int o = 0; o < T_nr;o++)
{
  beta = 1.0/(k_B*T(o));
  for (int q = first_allowed_density_row;
       q < row;
       q+=int(rows_left/rho_nr))
  {
    E1_F = cp(q,1);
    E3_F = cp(q,2);
    E4_F = cp(q,3);

    em_all = l3int(amp, params, E1_F, E3_F, E4_F, beta);

    if (params->my_rank == 0)
    {
      double L = 0;
      for (int i = 0; i < params->dim(0).steps;i++)
      {
        L+=em_all[i];
      }
      //Multiply the integration results with constnats
      L *= (pow(cos (cabbibo_angle),2)
          * pow( 1.0/beta, 5) * pow(2.0, 4.0) )
         / ( pow(4*pi,5) * hbar * pow(hbar*c,3) );
        double anal = getAnalytcialL( T(o), p1->getMass(),
            p3->getMass(), E4_F);
        cout << "#Rho = " << rho(q,0)
          << "   T = " << T(o)
          << "   Pfrac = " << rho(q,2)*100/rho(q,0)
          << "   Anal = " << anal
          << "   L = " << L
          << "   RelError = " << anal/L
          <<  endl;
        data << rho(q,0) << " "
          << T(o)<< " "
          <<   rho(q,2)*100/rho(q,0) << " "
          << anal << " "
          <<  L << " "
          << anal/L
          <<  endl;
      }
    }
  }
  data.close();
}

//Get analytical Luminosity
double getAnalytcialL(double T, double m_1,double m_3,
    double cp_4)
{
  double g_a = -1.26;
  return (457*pi / 10080)
    * pow( G_F * cos(cabbibo_angle),2)
    * ( 1.0 + 3*(g_a*g_a))
    * m_1 * m_3 * cp_4 * pow (k_B * T,6)
    * / ( hbar * pow(hbar * c, 3) ) ;
}

//Read datafile
void readdatafile(Array<double,2> buffer, int row, int col,
    const char* filename )
{
  ifstream infile;
  infile.open(filename);

  double datafield;
  for (int i = 0; i < row; i++)
  {
    for (int j = 0; j < col; j++)
    {
      infile >> buffer(i,j);
      if (infile.eof())
      {
        infile.close();
        return;
      }
    }
  }
  infile.close();
  return;
}

//Integration rutine
double* l3int(Amplitude* amp, Parameters* params,
    double E1_F, double E3_F, double E4_F, double beta)
{
  double E1, E2, E3, E4;
  double n, n_1, n_3, n_4;
  double x2, x2_min;
  double amplitude;

  Particle* p1 = amp->getParticle(1);
  Particle* p2 = amp->getParticle(2);
  Particle* p3 = amp->getParticle(3);
  Particle* p4 = amp->getParticle(4);

  double x1_min, x1_max, x1_sum;
  double x3_min, x3_max, x3_sum;
  double x4_min, x4_max, x4_sum;
  int x1_steps, x3_steps, x4_steps;

  x1_steps = params->dim(0).steps;
  x1_min   = params->dim(0).min;
  x1_max   = params->dim(0).max;

  x2_min = p2->getMass()*beta;

  x3_steps = params->dim(1).steps;
```

```
x3_min   = params->dim(1).min;
x3_max   = params->dim(1).max;

x4_steps = params->dim(2).steps;
x4_min   = params->dim(2).min;
x4_max   = params->dim(2).max;

x1_min =  x2_min + x3_min + x4_min;

//Using effective masses
p1->setMass(E1_F + x1_min/beta);
p3->setMass(E3_F + x3_min/beta);
p4->setMass(E4_F + x4_min/beta);

double *x1 = new double[x1_steps];
double *W_x1 = new double[x1_steps];
double *x3 = new double[x3_steps];
double *W_x3 = new double[x3_steps];
double *x4 = new double[x4_steps];
double *W_x4 = new double[x4_steps];

double *em = new double[x1_steps];
double *em_all = new double[x1_steps];

for (int i=0; i < x1_steps;i++) em[i]=0;
for (int i=0; i < x1_steps;i++) em_all[i]=0;

gauleg(x1_min, x1_max, x1, W_x1, x1_steps);

x1_sum = 0;
for (int i = params->my_rank; i < x1_steps; i+=params->size )
{
  E1 = x1[i]/beta + E1_F;
  p1->setAbsImpulseZ(E1);

  x3_max = x1[i] - x2_min - x4_min;
  gauleg(x3_min, x3_max, x3, W_x3, x3_steps);
  x3_sum = 0;
  for (int j = 0; j < x3_steps; j++ )
  {
    E3 = x3[j]/beta + E3_F;
    p3->setAbsImpulseZ(E3);
```

```
      x4_max = x1[i] - x2_min - x3[j];
      gauleg(x4_min, x4_max, x4, W_x4, x4_steps);
      x4_sum = 0;
      for (int k = 0; k < x4_steps; k++ )
      {
        E4 = x4[k]/beta + E4_F;
        p4->setAbsImpulseZ(E4);

        x2 = x1[i] - ( x3[j] + x4[k] );
        E2 = x2/beta;
        p2->setAbsImpulseZ(E2);

        n_1 = 1.0/(1.0 + exp(x1[i]));
        n_3 = 1.0/(1.0 + exp(-x3[j]));
        n_4 = 1.0/(1.0 + exp(-x4[k]));

        n =  n_1 * n_3 * n_4;

        amplitude = amp->calculateSpinAvrage() ;
        x4_sum += x2 * x2 * amplitude *  n *  W_x4[k] ;
        double p_1, p_2, p_3, p_4;
        p_1 = p1->getAbsImpulse();
        p_2 = p2->getAbsImpulse();
        p_3 = p3->getAbsImpulse();
        p_4 = p4->getAbsImpulse();
      }
      x3_sum += x4_sum * W_x3[j];
    }
    em[i]=x3_sum * W_x1[i];
  }

  MPI::COMM_WORLD.Reduce(em,em_all,x1_steps,
      MPI::DOUBLE,MPI::SUM, 0);

  delete [] x1;
  delete [] W_x1;
  delete [] x3;
  delete [] W_x3;
  delete [] x4;
  delete [] W_x4;
  delete [] em;

  return em_all;
```

```
}
```

Bibliography

[1] Sutharsan Arumugam.

[2] Walter Baade and Fritz Zwicky. Supernovae and cosmic rays. *Phys. Rev.*, 45, 1933.

[3] James Chadwick. On the possible existence of a neutron star. *Nature*, 129, 1932.

[4] Hong-Yee Chiu and E. E. Salpeter. Surface x-ray emission from neutron stars. *Phys. Rev. Lett.*, 12(15):413–415, Apr 1964.

[5] Duane A. Dicus. Stellar energy-loss rates in a convergent theory of weak and electromagnetic interactions. *Phys. Rev. D*, 6(4):941–949, Aug 1972.

[6] Friman, B. L. and Maxwell, O. V. Neutrino emissivities of neutron stars. *Astro. Phys. Journal*, 232:541–557, September 1979.

[7] G. Gamow and M. Schoenberg. Neutrino Theory of Stellar Collapse. *Physical Review*, 59:539–547, April 1941.

[8] R. Giacconi, H. Gursky, F. Paolini, and B.B Rossi. *Phys. Rev. Letters*, 9, 1962.

[9] David Griffiths.

[10] Henning Heiselberg and Morten Hjorth-Jensen. Phases of dense matter in neutron stars. *Physics Reports*, 328, 2000.

[11] A. Hewish, S. J. Bell, J. D. H. Pilkington, P. F. Scott, and R. A. Collins. Observation of a rapidly pulsating radio source. *Nature*, 217, 1968.

[12] C. J. Horowitz and Brian D. Serot. The relativistic two-nucleon problem in nuclear matter. *Nuclear Physics A*, 464:613–699, March 1987.

[13] J.M. Lattimer and M. Prakash. The physics of neutron stars. *Science*, 304, 2004.

[14] F. Mandl and G. Shaw. *Black Holes, White Dwarfs, and Neutron Stars.* 1983.

[15] J. R. Oppenheimer and G. M. Volkoff. On massive neutron cores. *Phys. Rev.*, 55, 1939.

[16] Michal E. Peskin and Daniel V. Schroeder. *An Introduction to Quantum Field Theory.* 1995.

[17] Stuart L. Shapiro and Saul A. Teukolsky. *Black Holes, White Dwarfs, and Neutron Stars.* 1983.

[18] C. Thompson and R.C. Duncan. *Astrophys. J.*, 408, 1994.

[19] D. Tubbs and D. Schramm. *Astrophys. J.*, 201, 1975.

[20] T.E. Strohmayer W. Zhang and J.H. Swank. *Astrophys. J.*, 482, 1997.

[21] R. Wijnands and M. van der Klis. *Nature*, 394, 1998.