NUMERICAL ANALYSIS OF THE STELLAR DYNAMICS OF A HYPOTHETICAL STRANGE QUARK STAR

By

STEVEN RICHARD NARF

Bachelor of Science
Mankato State University
Mankato, Minnesota
1992

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1995

NUMERICAL ANALYSIS OF THE STELLAR DYNAMICS OF A HYPOTHETICAL STRANGE QUARK STAR

Thesis Approved:

Satyanarayan Nandi

Thesis Adviser

Peter O. Shull Jr.

Paul Westhaus

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

First of all I would like to express my deepest thanks to my wonderful wife Tracy for putting up with me during this project. I could not have finished without her support and encouragement. For this I would like to say that I love you and thank you from the bottom of my heart for everything that you have done.

I wish to express my thanks and appreciation to both Dr. N.V.V.J. Swamy and Dr. S. Nandi for advising and working with me for the past few years on this project. I would also like to express my thanks and appreciations to my commitee members, Dr. P.A. Westhaus and Dr. P.O. Shull, whom I have had the privilege of working with.

My thanks and best wishes go out to my office-mates: Dave Muller for helping with the editing, Steve Gibbons for the never ending debates that made me think in a different way, Tesfaye Abraha for our lunch time talks and Bhaskar Dutta for just being himself when I asked him questions. Thanks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

The History

Introduction

Throughout the history of astrophysics, it has been a major challenge to determine the final fate of stellar objects like our sun. When a star exhausts all of its usable fuel in the core, then depending solely on its initial mass, the star can take one of several paths to form a white dwarf star, a neutron star, or even a black hole.

White dwarf stars are considered to be a very stable stellar configuration that is made up of a cold degenerate electron gas, where the star's internal outward pressure from the degenerate electrons prevents the star from collapsing further (by the Pauli principle). If the degenerate electron gas pressure is not enough to maintain stellar equilibrium against the star's gravitational field, the star will pass the point of being a stable white dwarf and approach a new phase of matter. This new state of matter is formed when the gravitational pressures becomes so great that the protons can grab electrons by means of inverse β-decay and convert into neutrons, which makes this matter more stable under these pressures. This matter is usually formed when the initial mass is large enough to initiate a catastrophic explosion called a supernova which leaves behind stellar dust and perhaps the innermost core. It is this innermost core that collapses under the core's gravitational field to form this degenerate neutron gas and it is called a neutron star. This neutron star, unlike its counterpart the white dwarf star, is now considered to be made up of a cold degenerate neutron gas. In this case, it is the degenerate neutron gas pressure which prevents this star from collapsing further.

Allowing for the possibility that some stars can collapse further than a neutron star, some questions come to mind about the final fate of such a stellar object. If the degenerate neutron gas pressure is not enough to maintain stellar equilibrium against the star's gravitational field, then what would be the outcome of such a star? It is believed that a star which is massive enough to collapse past neutronic degeneracy will ultimately become a black hole and this answer seems to be universally accepted at present. However, if we assume that a massive star can collapses past neutronic degeneracy, but does not form a black hole, then there must be a stable point between that of a neutron star and a black hole. If this is so, what would be the new form of this matter that would be formed at such high stellar densities and is it truly stable? At present, the answers to these questions are open to debate. This new stellar object, stable or not, has been presumed to be a strange quark star [1–3]. This thesis will explore the possibility of a stellar object that fits the above description and we will try to answer the above question.

In the next section we will bring forth a brief historical background on white dwarf and neutron stars and then discuss the possibility of having a quark star. The following section gives the history of a quark star by exploring three main models and the equations of state that were used for each of these models. The last section in Chapter I presents a brief historical summary of results concerning the possibility of the existence of strange quark stars. Chapter II has three parts: part one shows the mathematical approach to be used in this thesis, part two explains the equations of state that describe the quark matter, and part three explains the strange quark star model with the equations of state that describe this model. Part one of Chapter III contains the development of a free-fall and a non-free-fall tests that were performed on the Fortran computer code with a brief discussion of the results of the tests. Part two of Chapter III contains a discussion of the initial conditions used in our strange quark star non-free-fall program. At the end of this thesis there are three appendices. Appendix A shows the Fortran expressions for the equations of state of the quark matter that are used in the program along with the May and White equations that we had to correct. Appendix B contains the non-free-fall computer code for our strange quark star model. Appendix C contains the non-free-fall computer code that uses May and White's equations of state. We added this appendix because there is substantially more code incorporated into the program of Appendix B. In this way, the program in Appendix C forms the root of the program in Appendix B with the strange quark equations of state in place of May and White's. Our major contributions to this problem can be found in the last sections of Chapters II and III, all of Chapter IV, and the three appendices.

White Dwarf Stars To Quark Stars

In 1930 Chandrasekhar was working on a white dwarf star model* using a degenerate electron gas equation of state with special relativistic effects of the degenerate electrons included. He found that white dwarf stars have a maximum mass limit of $1.4M_{\odot}$. This result was his major discovery and $1.4M_{\odot}$ is now known as the Chandrasekhar limit for white dwarfs. Independently, Landau presented his explanation of the same model in 1932 and after the discovery of the neutron by Chadwick in the same year, Landau thought of the possibility of having a neutron star. Two years later in 1934 Baade and Zwicky also proposed this same idea [4]. During this time, the meaning of the Chandrasekhar limit was that if a star has an initial mass less then the $1.4M_{\odot}$ limit, a white dwarf star will be formed. However, if the initial mass is greater then the $1.4M_{\odot}$ limit and the star does not experience any kind of mass loss, then it was expected to catastrophically collapse under its own gravitational field indefinitely. We know now that this isn't completely true. We do accept the fact that a star with an initial mass less then the Chandrasekhar limit will form a white dwarf star, but we now believe that when the initial mass is greater then this limit, it will collapse further forming either a neutron star or a black hole.

In 1939, Oppenheimer and Volkoff presented a model for a new phase transition within stellar matter. When the pressures become high enough inside stellar bodies, a phase transition will takes place converting the star, which was mainly ionized atoms, into a degenerate gas of neutrons by the means of inverse β-Decay. The treatment of this model was to assume that the matter in the star was composed

*A uniform distribution of matter comprised of ionized atoms with a gas cloud of degenerate electrons.

of an ideal gas made up of pure neutrons and accounting for the general relativistic effects of these degenerate neutrons led to the Oppenheimer and Volkoff hydrostatic equilibrium equation,

$$\frac{dP}{dr} = -\frac{G\rho(r)\,m(r)}{r^2}\left(1+\frac{P(r)}{\rho(r)\,c^2}\right)\left(1+\frac{4\pi P(r)\,r^3}{m(r)\,c^2}\right)\left(1-\frac{2Gm(r)}{rc^2}\right)^{-1}, \quad (1)$$

where P, ρ, and m are all functions of the radius r and are defined as the pressure, density, and mass, respectively [4]. Since, the mass is a function of the radius, M is just the total mass enclosed by a radius R, as shown below:

$$M = \int_0^R 4\pi r^2 \rho(r)\,dr\ . \quad (2)$$

They concluded that with such high mass densities that are possible inside stars, general relativistic effects must be considered and a more precise equation of state should be used to describe the state of the matter (*i.e.* one that doesn't neglect the gravitational effects of neutrons) [5]. From this, the idea of a star collapsing under its own gravitational field to form a stable neutron star now seemed to be a possibility.

Oppenheimer and Volkoff argued that neutron stars should also have an upper mass limit not unlike that of the white dwarf stars, which have the Chandrasekhar limit. With this idea and Landau's model dealing with a cold degenerate Fermi gas of neutrons, one obtains a critical mass of $1.5M_\odot$ for a neutron star which is essentially the same limit as Chandrasekhar's limit for the white dwarfs. Oppenheimer and Volkoff, with their equations of state, came up with a neutron star critical mass of $0.7M_\odot$. One can also come up with a Chandrasekhar limit for a neutron star which is $5.73M_\odot$ [4]. The main reason for the difference between the $5.73M_\odot$ limit and the $0.7M_\odot$ limit is because of the relativity considerations that Oppenheimer and Volkoff employed.

Unfortunately, when the understanding of thermonuclear fusion came in, the idea of a neutron star receded into the background for the next thirty years, since it was thought that the main source of energy inside neutron stars was not degenerate neutron pressure (which wasn't a thought until the mid 1960's), but rather the neutron cores inside massive normal stars. We would like to point out at this time that for those thirty years, the idea of neutron stars did not actually fade away even though there was a drop in interest. However, during those thirty years, some work on topics related to neutron stars was continued by Wheeler in 1959, Misner and Sharp in 1964, McVittie in 1965, and May and White in 1965 and 1966. We borrowed heavily from May and White's work on gravitational collapse with general relativistic effects [6] for the mathematical approach that is used in Chapter II.

In recent times the idea of a neutron star is widely accepted because it allows us to understand the stellar objects known as Pulsars. However, the upper mass limit is still uncertain because of the uncertainty in the composition of the neutron stars which puts the accuracy of the equation of state in question. The upper mass limit is more of a range, ranging from 1.4 to $3.0M_\odot$ [4]. In closing this part of history, we ask a question. As a star collapses the internal mass density increases and depending on the star's initial mass it will become a white dwarf star up to $1.4M_\odot$ or a neutron star up to $\sim 3.0M_\odot$ or a black hole. What if there is another mass limit after the neutron star phase but before the black hole phase? A strange quark star?

Quark Stars Models

Introduction

Here we will look at three different models dealing with the astrophysics of quark matter. The first of these models is the non-interacting Fermi gas model which was used by Itoh in 1970 [7]. The second model is the asymptotic MIT bag model of hadrons using the effective quark-gluon coupling constant fixed to first order. The third model is the Chapline and Nauenberg's perturbational quantum chromodynamics (Q.C.D.) model to first order in the density dependent effective quark-gluon coupling constant [8,9]. These three models essentially differ in the definition of the equation of state that is being invoked.

The Non-Interacting Fermi Gas Model

In Itoh's letter, he investigates the hydrostatic equilibrium of a quark star using the Oppenheimer and Volkoff method, equations (1) and (2) [7,5]. He also assumes the conventional quark model at this time, but in the early 1970's the interaction between quarks was not well understood. So he neglected this interaction between the quarks, even though this is not permissible at densities near that of the characteristic nuclear density, $\rho_N = 2.8 \times 10^{14} \frac{g}{cm^3}$, since the quarks are considered to be strongly interacting with each other at these densities. Model I is known as the non-interacting Fermi gas model and assumes that a quark star exists with equal amounts of each flavor (u, d, and s quarks). The model also makes the assumption that all quark masses are equal to each other, m_q, which was assumed to be $10GeV/c^2$. The equations of state that make up Itoh's model, Model I, are

$$P = 7.92 \times 10^{36} m_q^4 f(\chi_q), \ \frac{dyn}{cm^2}, \tag{3}$$

$$\rho' = \frac{\epsilon'}{c^2} = 5.34 \times 10^{15} \frac{n}{m_q} + 8.81 \times 10^{15} m_q^4 g(\chi_q), \ \frac{g}{cm^3}, \tag{4}$$

$$f(\chi) = \left(2\chi^2 - 3\right)\left(\chi^2 + 1\right)^{\frac{1}{2}} + 3\ arcsinh(\chi), \tag{5}$$

$$g(\chi) = 8\chi^3\left[\left(\chi^2 + 1\right)^{\frac{1}{2}} - 1\right] - f(\chi), \tag{6}$$

and

$$\chi_q = \frac{p_{F,q}}{m_q c}, \tag{7}$$

where P is the pressure, ϵ' is the total energy density, m_q is the quark mass in units of GeV/c^2, n is the baryon number density in units of fm^{-3} where n is related to χ_q by $n = 13.2\ m_q^3\chi_q^3$, and $p_{F,q}$ is the quark Fermi momentum. Itoh found that the maximum mass for a quark star was several orders of magnitude lower then that of a neutron star, so he concluded that all stars more massive then the maximum mass for a neutron star will collapse since hydrostatic equilibrium is not a possibility by model I.

The Asymptotic MIT Bag Model

The asymptotic MIT bag model, model II, introduced in 1974 was successful in describing hadron spectroscopy* and recently has had a renewal with the birth of the hypothetical strange quark matter or quark stars. A similar model was used in 1976 on a study of a spherically symmetric, non-rotating, general relativistic neutron star looking for hydrostatic equilibrium solutions to a polytropic equation of state [10]. The equations of state for model II that were used by Øveräard and Østaard in 1991 are

$$P = \frac{1}{3}(\epsilon' - 4B), \tag{8}$$

$$\epsilon' = An^{\frac{3}{4}} + B, \tag{9}$$

and

$$A = \frac{9}{4}\left(\frac{3\pi^2}{K}\right)^{\frac{1}{3}}\left(1 + \frac{8}{3}\left(\frac{\alpha_c}{\pi}\right)\right)\hbar c, \tag{10}$$

where P, ϵ', n have the same definitions as in model I and B is the "bag pressure" constant otherwise known as the MIT bag model constant. These equations of state carry A to second order in the quark-gluon coupling constant g where α_c is defined to be $g^2/4\pi$ to first order, where K is equal to three since there are three flavors of quark that this model is considering (u, d, and s quarks).

Haensel *el al.*, Alcock *el al.*, and Farhi and Jaffe all considered the following equations of state for model II which are believed to be more accurate in describing the quark matter state [1–3]:

$$P(\mu) = -\sum_i \Omega_i(\mu) - B \tag{11}$$

and

$$\epsilon' = \sum_i (\Omega_i(\mu_i) + \mu_i n_i(\mu_i)) + B\ . \tag{12}$$

Here $\Omega_i(\mu_i)$ are the thermodynamic potentials and $n_i(\mu_i)$ are the number densities for each constituent $i = u, d, s, e$ and are both a function of the chemical potentials

*The topic of hadron spectroscopy will not be discussed in this thesis. See references: Øveräard and Østaard, Chodos *et al.* 1974a, 1974b, and DeGrand *et al.* 1975.

μ_i. B is still the MIT bag model constant. In Chapter II, we will discuss in more detail the variables Ω_i, μ_i, n_i, B, and the μ from equation (11). For the purpose of this thesis we will adopt model II with a variation of the above equations of state, eqs. (11) and (12).

The Perturbational Q.C.D. Model

Chapline and Nauenberg's perturbational Q.C.D. model [8], model III, makes up the third model that will be looked at. The big difference between model III and the MIT bag model is that model III includes the energy dependence of the strong coupling constant, α_c:

$$\alpha_c = \left[\frac{\pi}{\left(22 - \frac{4}{3}K\right)}\right]\left(\frac{1}{\ln(\chi)}\right) , \tag{13}$$

here $K = 3$ for total number of quark flavors being used (u, d, and s) and $\chi \equiv \frac{k_F}{\Lambda_F}$ is the ratio of the Fermi momentum, k_F, to a constant cut-off factor, Λ_F. The energy density and number density for this model are defined to be

$$\epsilon' = An^{\frac{4}{3}} \tag{14}$$

and

$$n = \frac{\Lambda_F^3 \chi^3}{\pi^2} \tag{15}$$

with A given by equation (10) from model II. With the above two equations, the total energy density and internal pressure can be rewritten as

$$\epsilon' = \frac{9\hbar c}{4\pi^2}\left(1 - \frac{4}{27 \ln \chi}\right) \Lambda_F^4 \chi^4 \tag{16}$$

and

$$P = n\frac{d\epsilon'}{dn} - \epsilon' = \frac{3\hbar c}{4\pi^2}\left[1 - \frac{4}{27 \ln \chi}\left(1 - \frac{1}{\ln \chi}\right)\right] \Lambda_F^4 \chi^4, \tag{17}$$

which are the equations of state this model.

With this, the pressure equation from model II, eq. (8), can by rewritten as

$$P = \frac{1}{3}[\epsilon' - 4B(\chi)] \tag{18}$$

and

$$B(\chi) = \frac{\hbar c}{12\pi^2 \ln^2(\chi)} \Lambda_F^4 \chi^4, \tag{19}$$

where $B(\chi)$ is the density dependent bag pressure which is analogous to the MIT bag constant B in model II. Equations (8) and (18) both take the masses of the quarks to be equal to zero; however, equation (11) in model II takes m_s to be non-zero, but allows $m_u = m_d = m_e = 0$.

Results From History

The Results here are organized into four tables by model type. Tables 1, 2, and 3 show some results that were obtained by using models I, II, and III, respectively. Table 4 contains typical neutron star data. These tables are just a summary of some of the major results throughout the history of stellar dynamics involving the stability of stars (mainly strange quark stars) and include values for the maximum mass and radius limits. All the tables are in standard units for this subject which are: the mass, M, in solar masses ($M_{\odot}$), the radius, R, in kilometers (km), and time in seconds (s), unless otherwise stated. The tables are constructed with three columns entitled Results, Initial Conditions ($IC's$) and Comments, and References. Other variables that are in the tables are the maximum moment of inertia, $I_{\max}$ (in $g\ cm^2$), the mass density, ρ (in g/cm^3), and the m_q which is the quark mass (in MeV/c^2) that is being used for that model.

TABLE I.

Results from using the non-interacting Fermi gas model.

Results	IC's and Comments	Reference
$M_{max} = 1.3 M_{\odot}$ $R_{max} = 17 km$ $I_{max} = 1.59 \times 10^{45} g\, cm^2$	$m_q = 400 MeV/c^2$	[11]
$M_{max} = 2 \times 10^{-3} M_{\odot}$ $R_{max} = 25 m$	$m_q = 10 GeV/c^2$ and reproduced Itoh's 1970 [7]	[11]

TABLE II.

Results from using the asymptotic MIT bag model.

Results	IC's and Comments	Reference
$M_{max} = 2.033\,(B_o/B)^{\frac{1}{2}}\,M_{\odot}$ $R_{max} = 11.09\,(B_o/B)^{\frac{1}{2}}\,km$ $I_{max} = 1.59 \times 10^{45}\,(B_o/B)^{\frac{3}{2}}\,g\,cm^2$ $\rho_{max} = 1.92 \times 10^{15}\,(B_o/B)\,g/cm^3$	$m_q = 279\frac{MeV}{c^2}$, $K = 3$, $m_u = m_d = 0$, with $B_o = 56\frac{MeV}{fm^3}$, and $\alpha_c = 0.549$	[11]
$M_{max} = 1.98 M_{\odot}$ $R_{max} = 10.79 km$ $I_{max} = 2.20 \times 10^{45} g\,cm^2$	same as above and for comparison he used OV eqs. (1) and (2)	[11]
$M_{max} = 1.964 M_{\odot}$ $R_{max} = 10.71 km$ $\rho_{max} = 2.057 \times 10^{15} g/cm^3$ $I_{max} = 2.159 \times 10^{45} g\,cm^2$	$M = 1.4 M_{\odot}$ and $B = 60\frac{MeV}{fm^3}$	[1]
$M_{max} = 1.75 M_{\odot}$ $R_{max} = 9.81 km$ $\rho_{max} = 2.45 \times 10^{15} g/cm^3$ $I_{max} = 4.58 \times 10^{45} g\,cm^2$	non-zero m_s 1^{st} order in quark-quark interactions	[1]
$M_{max} = 2.185 M_{\odot}$ $R_{max} = 12.06 km$ $I_{max} = 3.01 \times 10^{45} g\,cm^2$	his SQ2 model using $M = 1.4 M_{\odot}$ and scaling B	[1]
$\bar{M}_{max} = 2.00 M_{\odot}$ $R_{max} = 11.1 km$ $\rho_{max} = 1.91 \times 10^{15} g/cm^3$	his model used $m_i = 0$ and $B = 56\frac{MeV}{fm^3}$ he ignored quark masses	[12]

TABLE III.

Results from using the perturbational Q.C.D. model.

Results	IC's and Comments	Reference
$M_{max} = 1.32 M_{\odot}$ $R_{max} = 7.20 km$ $I_{max} = 0.67 \times 10^{45} g\, cm^2$	$\Lambda_F = 250$	[11]

TABLE IV.

Results for a typical "normal" neutron star.

Results	IC's and Comments	Reference
$M_{max} = 1.4 M_{\odot}$ $R_{max} = 11.91 km$ $I_{max} = 1.361 \times 10^{45} g\, cm^2$ $N_b = 1.841 \times 10^{57}$	$M = 1.4 M_{\odot}$ $N_b =$ Baryon Number	[11]

CHAPTER II

Mathematical Approach

Introduction

This chapter has three main parts. Part one contains the mathematical approach for the hydrodynamics of the fluid flow that is used in the Fortran program under the subroutine called Dynami. The second part contains the mathematical approach for the equations of state used to describe the quark matter. Part three describes the strange quark model with the appropriate equations of state for strange quark confinement and can be found throughout the Fortran program under the subroutines called Initial, Thermo_Potentials, and others. We have borrowed heavily from May and White for our discussion on the hydrodynamics of the fluid flow and from Haensel *et al.* for the quark matter properties (for more details see references [5,13,6,14], and [15]). To insure a bit of consistency in the notation, we **will not** be using standard units (*ie.* $c = \hbar = 1$). The main units that will be encountered are kilometers (km), solar masses ($M_\odot$), seconds (s), and mega-electronvolts (MeV which converts to $M_\odot \left(\frac{km}{s}\right)^2$).

Mathematical Background

For the rest of this section we will be employing the Einstein summation convention where we sum over two repeated Greek indices unless otherwise stated. We will also be using the standard interpretation regarding tensor indices, where raised indices are contravariant and the lowered indices are covariant. In the following subsections the reader will find that the notation changes where Greek character κ is concerned. The character κ will take on two roles, one as a tensor index and the other as a radial coordinate which carries the mass as a conserved quantity.

The Metric

The initial conditions for the setup of our quark star are very important and is discussed in the last section. We will assume that the star is perfectly spherical and of uniform density. The most general line element or metric with spherical symmetry that satisfies the above conditions is

$$ds^2 = a^2 \left(dx^0\right)^2 - b^2 \left(dx^1\right)^2 - R^2 \left(d\Omega\right)^2 + \eta dx^0 dx^1, \tag{20}$$

where

$$\left(d\Omega\right)^2 = d\theta^2 + \sin^2\theta d\phi^2 \tag{21}$$

and the metric coefficients $a, b, R,$ and η are all functions of the time coordinate, $x^0 \equiv ct$, and the radial coordinate, $x^1 \equiv \kappa$. We will set $\eta = 0$ for simplicity. Since we want to view the dynamics in proper time, we need to view the 4-velocity of the fluid, u^ν, in a comoving frame*. Therefore the fluid 4-velocity must take the form

$$u^\nu = \begin{cases} u^0, & \text{for } \nu = 0, \\ 0, & \text{for } \nu \neq 0, \end{cases} \tag{22}$$

and must satisfy

$$u^\nu u_\nu = g_{\nu\lambda} u^\lambda u^\nu = c^2, \tag{23}$$

the condition for any 4-velocity in a comoving spherically symmetric frame. When combining equations (20, 21) and (22), the metric takes the form of

$$ds^2 = a^2 c^2 dt^2 - b^2 d\kappa^2 - R^2 d\Omega^2 \ , \tag{24}$$

where a and b are now only functions of R and t. With equations (22) and (24), the 4-velocity of the fluid, u^ν, in a comoving frame takes the form of

$$u^\nu = \begin{cases} a^{-1} c, & \text{for } \nu = 0, \\ 0, & \text{for } \nu \neq 0. \end{cases} \tag{25}$$

*A comoving frame is best described as an observer in a locally inertial frame which is moving with the fluid at the instant the observer makes his measurements.

The Energy Momentum Tensor

From the above description of the metric, eq. (24), and of the fluid 4-velocity, eq. (25), in a comoving spherically symmetric frame, an energy momentum tensor for a perfect fluid or gas can be constructed (see ref: [13,16,17] and [18]). Thus, the general form for the energy momentum tensor for a perfect fluid is

$$T^{\mu}_{\nu} = \rho\omega g_{\nu\lambda} u^{\lambda} u^{\mu} - g^{\mu}_{\nu} P \ . \tag{26}$$

When taking the components of equation (26) one finds the following:

$$\begin{aligned} T^0_0 &= \rho c^2 \left(1 + \frac{\varepsilon}{c^2}\right) \ , \\ T^1_1 &= T^2_2 = T^3_3 = -P \ , \end{aligned} \tag{27}$$

and

$$T^{\mu}_{\nu} = 0 \quad \text{for } \mu \neq \nu \ ,$$

where P is the proper pressure, ω is the relativistic enthalpy or heat function, ρ is the proper rest mass density which is not the same as the ρ' earlier, and ε is defined as the proper specific internal energy per unit mass and should not be confused with ϵ' from before. These four variables $\rho, \varepsilon, \omega$, and P will be defined in more detail later on.

The Einstein Field Equations

Order to keep the following equations in a short form, we make the following notational change. The notation for the traditional partial derivative will been replaced by a subscript notation. Thus

$$\frac{\partial x}{\partial t} \equiv x_t \tag{28}$$

and

$$\frac{\partial x}{\partial \kappa} \equiv x_{\kappa} \ , \tag{29}$$

where the time, t, and radial coordinate, κ, will be the only two subscripts that denote partial derivatives.

The Einstein field equations are also known as the gravitational field equations and with the use of equations (24) and (27) the compact form of these field equations are given by the tensor relation,

$$-\frac{8\pi G}{c^4}T^{\mu}_{\nu} = R^{\mu}_{\nu} - \frac{1}{2}g^{\mu}_{\nu}R^{\lambda}_{\lambda} \ . \tag{30}$$

By introducing a mass function, $m(\kappa, t)$, to equation (30), [4,6] and [18], each field equation takes the form:

$$\begin{aligned}
\left(T^0_0\right) \quad 4\pi\rho G\left(1+\frac{\varepsilon}{c^2}\right)R^2R_\kappa &= \frac{c^2}{2}\left[R+\frac{RR_t^2}{a^2c^2}-\frac{RR_\kappa^2}{b^2}\right]_\kappa \\
&\equiv Gm_\kappa \ , && (31)\\
\left(T^1_1\right) \quad \frac{4\pi G}{c^2}PR^2R_t &= -\frac{c^2}{2}\left[R+\frac{RR_t^2}{a^2c^2}-\frac{RR_\kappa^2}{b^2}\right]_t \\
&\equiv -Gm_t \ , && (32)\\
\left(T^2_2=T^3_3\right) \quad 4\pi G\rho\omega R^3 &= c^2\left[R+\frac{RR_t^2}{a^2c^2}-\frac{RR_\kappa^2}{b^2}\right] \\
&\quad +\frac{R^3c}{ab}\left[\left(\frac{ca_\kappa}{b}\right)_\kappa-\left(\frac{b_t}{ac}\right)_t\right] \ , && (33)
\end{aligned}$$

and

$$\left(T^1_0=T^0_1\right) \qquad 0 = \frac{a_\kappa}{a}R_t+\frac{b_t}{b}R_\kappa-R_{\kappa t} \ . \tag{34}$$

We now define $m(\kappa, t)$ as the total mass enclosed in a sphere of radius κ:

$$m(\kappa,t) = 4\pi\int_0^\kappa \rho\left(1+\frac{\varepsilon}{c^2}\right)R^2R_\kappa \, d\kappa \ . \tag{35}$$

We now complete the above set of equations, (31 - 35), with the addition of the state equation for pressure, P, in general form

$$P = P(\epsilon') = P(\rho, \varepsilon) \ , \tag{36}$$

where the pressure is a function of the total internal energy density, T^0_0.

Conservation Laws

The constituents in the ensemble must always be conserved and we will be assuming that neither a constituent nor its mass can be created or destroyed. For this, we must introduce the conservation of baryon number, which requires

$$\Box_\nu\left(\rho u^\nu\right) = 0 \ , \tag{37}$$

where ρ, the baryonic mass density, is

$$\rho = \sum_{i=1}^{N} n_i m_i \tag{38}$$

with m_i as the mass and n_i as the number density of each baryonic species or constituent. The total number density for each species is given by

$$n = \sum_{i=1}^{N} n_i \ . \tag{39}$$

With the above assumption and equations (38) and (39) the mass density, ρ, becomes

$$\rho = n\overline{m} \ , \tag{40}$$

where $\overline{m}$ is a constant. Since we are working in a comoving frame, the conservation of baryon number transforms to

$$(\rho R^2 b)_t = 0 \ . \tag{41}$$

By carrying out the partial derivatives in both equation (41) and in the Einstein field equations, eqs. (31) - (34), one can find a new function, $f(\kappa)$, which one then uses to fix the scale of the radial coordinate κ by letting

$$f(\kappa) \equiv \rho R^2 b = \frac{1}{4\pi} \ . \tag{42}$$

Thus,

$$b = \frac{1}{4\pi\rho R^2} \ , \tag{43}$$

which is the square root of the second coefficient in the metric equation (20) and we now can describe $\overline{m}$ as

$$\overline{m} = \int_0^{\kappa} 4\pi\rho R^2 b \, d\kappa \ , \tag{44}$$

which states that $\overline{m}$ is the total rest mass enclosed in a sphere of "radius" κ much like equations (2) and (35). In equation (44), ρ is still the proper rest mass density and the rest of the integrand is the proper volume between a radius of κ and $\kappa + \Delta\kappa$. This guarantees mass conservation by carrying the mass as a coordinate which is frequently used in classical Lagrangian hydrodynamics.

To make sure that energy and momentum are also conserved for the comoving frame, the following relations must be satisfied:

$$\left[\rho\left(1+\frac{\varepsilon}{c^2}\right)\right]_t+\left(\frac{b_t}{b}+\frac{2R_t}{R}\right)\rho\omega=0 \tag{45}$$

and

$$\frac{a_\kappa}{a}+\frac{P_\kappa}{\rho\omega c^2}=0\ . \tag{46}$$

By substituting equations (34) and (41) into equation (45) one gets

$$\varepsilon_t=\frac{\rho_t P}{\rho^2} \tag{47}$$

from the first law of thermodynamics with no heat transfer (*i.e.* we ignore entropy generation).

With everything we have just done in mind, there is still one thing left untouched and that is the relativistic enthalpy or the heat function that is inside equation (26). The relativistic enthalpy is defined as

$$\omega=1+\frac{\varepsilon}{c^2}+\frac{P}{\rho c^2}\ . \tag{48}$$

We also need a relation analogous to the relativistic enthalpy, but in the comoving frame. One can do this by making the following definitions

$$\Gamma\equiv\frac{R_\kappa}{b}\ , \tag{49}$$

and

$$u\equiv\frac{R_t}{a}\ . \tag{50}$$

When used in combination with equations (31, 35, 43, 44, 49) and (50), one obtains an expression for Γ^2 as

$$\Gamma^2=1+\left(\frac{u}{c}\right)^2-\frac{2mG}{Rc^2}\ . \tag{51}$$

We now can obtain an equation of motion from equations (34, 46, 50) and (51) which makes equation (25) become

$$u_t=-a\left[4\pi R^2\frac{\Gamma}{\omega}P_\kappa+\frac{mG}{R^2}+\frac{4\pi G}{c^2}PR\right]\ . \tag{52}$$

This is the equation of motion for the system that was described at the beginning of this section and with the use of equations (34, 43) and (52), the mass conservation equation now becomes

$$\frac{(\rho R^2)_t}{\rho R^2} = -\frac{a u_\kappa}{R_\kappa} \; . \tag{53}$$

Summary

From the above outline of the mathematics used, a complete set of equations for spherically symmetric general relativistic hydrodynamics can be constructed which are relisted below:

$$\begin{aligned}
u_t &= -a\left[4\pi R^2 \frac{\Gamma}{\omega} P_\kappa + \frac{mG}{R^2} + \frac{4\pi G}{c^2} P R\right] \; , && (54)\\
R_t &= a u \; , && (55)\\
\frac{(\rho R^2)_t}{\rho R^2} &= -\frac{a u_\kappa}{R_\kappa} \; , && (56)\\
\varepsilon_t &= -P\left(\frac{1}{\rho}\right)_t \; , && (57)\\
\frac{(a\omega)_\kappa}{a\omega} &= \frac{\left[\varepsilon_\kappa + P\left(\frac{1}{\rho}\right)_\kappa\right]}{\omega c^2} \; , && (58)\\
m &= 4\pi \int_0^\kappa \rho\left(1 + \frac{\varepsilon}{c^2}\right) R^2 R_\kappa \, d\kappa \; , && (59)\\
\Gamma &= 4\pi \rho R^2 R_\kappa \; , && (60)\\
P &= P(\epsilon, \rho) \; , && (61)
\end{aligned}$$

and

$$\omega = 1 + \frac{\varepsilon}{c^2} + \frac{P}{\rho c^2} \; . \tag{62}$$

Now the task is to obtain solutions for the above list of equations. But we first need to obtain an expression for the state equation, eq. (61), that represents the matter that will be dealt with.

Equations Of State For Quark Matter

In Chapter I, we took a brief look at three types of models on astrophysical quark matter confinement: the non-interacting Fermi gas model, the asymptotic MIT bag model of hadron, and perturbational Q.C.D. model. This thesis will be looking at the stellar dynamics using a variation of model II. Before we use the numerical integration scheme, summarized in the last section, on model II, we must first recall that the T_0^0 element from energy momentum tensor, eq. (26) or (27), is defined as the total internal energy density:

$$T_0^0 \equiv \epsilon' = \rho' c^2 \ . \tag{63}$$

May and White break up this total internal energy density, T_0^0, into two terms,

$$T_0^0 \equiv \rho c^2 + \rho \varepsilon \ , \tag{64}$$

where ρ is the proper rest mass density and ε is the proper specific internal energy per unit mass as in equation (27). From here on we will redefine χ to be the total internal energy density given by either equation (63) or by (64) and by definition the two are equivalent:

$$\chi \equiv \rho' c^2 = \rho c^2 + \rho \varepsilon \ . \tag{65}$$

Now all we need is to find the proper expressions for P, ρ, and ε from the total internal energy density given by equation (12) from Haensel *et al.*, which is

$$\chi = \sum_i \left(\Omega_i\left(\mu_i\right) + \mu_i n_i\left(\mu_i\right)\right) + B \ , \tag{66}$$

where the chemical potential for each constituent $i = u, d, s$, and e is given by μ_i. The MIT bag model constant, B, is defined as the amount of pressure that is required to inflate a bag of three quarks in open vacuum. Where the fluid is considered to be made of an ideal Fermi fluid, or gas, comprised of the three main quark flavors and electrons (*ie.* the constituent u, d, s, and e).

In addition to the above equation, eq. (66), we will need the thermodynamic potential, $\Omega_i\left(\mu_i\right)$, as outlined by Haensel, Zdunlk, and Schaeffer [1]. The thermodynamic potentials for each quark flavor and for the electrons are given below to first order in the strong coupling constant $\alpha_c = g^2/4\pi \simeq 0.45$:

$$\begin{aligned}
\Omega_f\left(\mu_f\right) &= -\frac{\mu_f^4}{4\pi^2\left(\hbar c\right)^3}\left[1-\frac{2\alpha_c}{\pi}\right], \qquad \text{where } f = u, d \text{ and } m_f = 0, && (67)\\
\Omega_s\left(\mu_s\right) &= -\frac{1}{4\pi^2\left(\hbar c\right)^3}\left[\mu_s\sqrt{\mu_s^2-m_s^2c^4}\left(\mu_s^2-\frac{5}{2}m_s^2c^4\right)\right.\\
&\left.+\frac{3}{2}m_s^4c^8\ln\left(\frac{\mu_s+\sqrt{\mu_s^2-m_s^2c^4}}{m_sc^2}\right)-\frac{2\alpha_c}{\pi}\left[3\left(\mu_s\sqrt{\mu_s^2-m_s^2c^4}\right.\right.\right.\\
&\left.\left.-m_s^2c^4\ln\left(\frac{\mu_s+\sqrt{\mu_s^2-m_s^2c^4}}{m_sc^2}\right)\right)^2-2\left(\mu_s^2-m_s^2c^4\right)\right.\\
&-3m_s^4c^8\ln^2\left(\frac{m_sc^2}{\mu_s}\right)+6\ln\left(\frac{\tilde{\rho}}{\mu_s}\right)\left(\mu_sm_s^2c^4\sqrt{\mu_s^2-m_s^2c^4}\right.\\
&\left.\left.\left.-m_s^4c^8\ln\left(\frac{\mu_s+\sqrt{\mu_s^2-m_s^2c^4}}{m_sc^2}\right)\right)\right]\right], && (68)
\end{aligned}$$

and

$$\Omega_e\left(\mu_e\right) = -\frac{\mu_e^4}{12\pi^2\left(\hbar c\right)^3}, \qquad (69)$$

where $\tilde{\rho}$ is the renormalization point for massive quarks defined as $\tilde{\rho} = M_Nc^2/3 = 313MeV$, with M_Nc^2 defined as the nucleon rest energy. We can now find the total number density for any one of these constituent by

$$n_i\left(\mu_i\right) = -\frac{\partial\Omega_i}{\partial\mu_i}, \qquad (70)$$

thus number densities for each constituent $i = u, d, s$, and e are as follows:

$$\begin{aligned}
n_f\left(\mu_f\right) &= \frac{\mu_f^3}{\pi^2\left(\hbar c\right)^3}\left[1-\frac{2\alpha_c}{\pi}\right], \qquad \text{where } f = u, d \text{ and } m_f = 0, && (71)\\
n_s\left(\mu_s\right) &= \frac{\mu_s^2-m_s^2c^4}{\pi^2\left(\hbar c\right)^3}\left[\sqrt{\mu_s^2-m_s^2c^4}-\frac{2\alpha_c}{\pi}\right.\\
&\left.\times\left[\mu_s-\frac{3m_s^2c^4}{\sqrt{\mu_s^2-m_s^2c^4}}\ln\left(\frac{\mu_s+\sqrt{\mu_s^2-m_s^2c^4}}{\tilde{\rho}}\right)\right]\right], && (72)
\end{aligned}$$

and

$$n_e\left(\mu_e\right) = \frac{\mu_e^3}{3\pi^2\left(\hbar c\right)^3}. \qquad (73)$$

From the discussion on the mathematical background we know that baryon number must be conserved, thus the total number of the quarks must also be conserved throughout the baryon-quark phases transition. The presumed way that the baryonic matter converts to this more stable quark matter at stellar densities near the nucleon density, ρ_N, is by the following weak interactions:

$$d \rightarrow u + e + \bar{\nu}_e \ , \tag{74}$$

$$u + e \rightarrow d + \nu_e \ , \tag{75}$$

$$s \rightarrow u + e + \bar{\nu}_e \ , \tag{76}$$

$$u + e \rightarrow s + \nu_e \ , \tag{77}$$

and

$$s + u \leftrightarrow d + u \ . \tag{78}$$

From these weak interaction processes the matter maintains equilibrium between the quarks and the electrons*. When the weak interaction processes are combined with the condition of overall charge neutrality, we have

$$\frac{2}{3} n_u\left(\mu_u\right) = \frac{1}{3}\left[n_d\left(\mu_d\right) + n_s\left(\mu_s\right)\right] + n_e\left(\mu_e\right) \ . \tag{79}$$

This makes the following constraints on the chemical potentials giving us a single independent chemical potential for quark matter that we called μ:

$$\begin{array}{c} \mu_d = \mu_s \equiv \mu \\ \mu_d = \mu_u + \mu_e \equiv \mu \end{array} \ , \tag{80}$$

thus,

$$\sum_i \mu_i = \mu_u + \mu_d + \mu_s + \mu_e = 3\mu \ . \tag{81}$$

The pressure equation is then derived from equation (66) by the differential equation

$$P = n_B \frac{\partial \chi}{\partial n_B} - \chi \ , \tag{82}$$

where $n_B = \frac{1}{3}\left(n_u\left(\mu_u\right) + n_d\left(\mu_d\right) + n_s\left(\mu_s\right)\right)$ is the baryon number density and $\dfrac{\partial \chi}{\partial n_B}$ is called the Gibbs potential which is defined as

$$G = \frac{\partial \chi}{\partial n_B} = \sum_i \mu_i \frac{\partial n_i\left(\mu_i\right)}{\partial n_B} = 3\mu - 3\mu_e \ . \tag{83}$$

*Note that the first four of these processes contribute to some energy loss due to neutrino emission. So we accept the possibility that the star's dynamics in the computer program may lose some mass.

But in order to derive the pressure equation from equation (82), one must first rewrite the total internal energy density, eq. (66), in terms of μ, the single independent chemical potential. Therefore the terms $\Omega_i(\mu_i)$ and $n_i(\mu_i)$, as defined by equations (67 - 73), must be rewritten as functions of the single independent chemical potential μ. We can do this by defining two new functons f and ξ as functions of both μ and μ_i, therefore $\Omega_i(\mu_i)$ and $n_i(\mu_i)$ can be rewritten in the following way:

$$\Omega_i(\mu_i) = \Omega_i(\mu) + f_i(\mu, \mu_i) \tag{84}$$

and

$$n_i(\mu_i) = n_i(\mu) + \xi_i(\mu, \mu_i) \ . \tag{85}$$

By substituting these into equation (66) we get

$$\chi = \sum_i [\Omega_i(\mu) + \mu_i n_i(\mu_i) + f_i(\mu, \mu_i)] + B \ , \tag{86}$$

where

$$\sum_i \mu_i n_i = 3\mu n_B + \mu_e [n_u(\mu) - n_e(\mu) + \xi_u(\mu, \mu_u) - \xi_e(\mu, \mu_e)] \ . \tag{87}$$

Typically μ_e and $\xi_u(\mu, \mu_u)$ will be very small in comparison to the contributions from the down and strange quarks and in some cases one can let μ_e tend to zero. Now we can derive two equivalent expressions for the pressure by using equation (82). The first expression is of the form

$$P(\mu) = -\sum_i \Omega_i(\mu) - B \ , \tag{88}$$

while the second expression takes the form $P(\rho, \varepsilon, \mu_i, \mu)$, which is

$$P(\rho, \varepsilon, \mu_i, \mu) = \sum_i \mu_i n_i + \sum_i f_i(\mu, \mu_i) - \rho c^2 - \rho\varepsilon \ . \tag{89}$$

In order to complete this set of equations for the pressure we need an expression for this single independent chemical potential μ. The following expression for μ is a recursion relation meaning that you take the old value of μ and plug it in to get a new value of μ then iterate until you converge on the final value of μ:

$$\begin{aligned} \mu_{new} \ = \ & \Big\{\chi - \sum_i \Omega_i\left(\mu^{old}\right) - \sum_i f_i\left(\mu^{old}, \mu_i^{old}\right) - B - \mu_e^{old}\Big[n_u\left(\mu^{old}\right) \\ & -n_e\left(\mu^{old}\right) + \xi_u\left(\mu^{old}, \mu_u^{old}\right) - \xi_e\left(\mu^{old}, \mu_e^{old}\right)\Big]\Big\} / 3n_B^{old} \ , \end{aligned} \tag{90}$$

where χ is equal to equation (64) and is held constant, along with B, during a single iteration process.

Since we will be considering only a strange quark star we will use the above equations for the limiting case of $i = s$ as will be shown in the next section. Otherwise the above derivation gives the description for quark confinement of the quark matter and Appendix A contains these expressions in a form that can be integrated into the Fortran program of Appendix B.

The Strange Quark Star Model

The last section gave us a prescription for calcuating the internal pressures in quark matter. We will simplify this iterative process by considering only the possibility of having a stable Strange Quark Star. Witten [12] pointed out that the most stable matter at these high stellar densities, where $\rho_o > \rho_N$, is strange quark matter. Therefore we now assume that an isolated strange quark star can exist, such that *the star** is charge neutral meaning that it obeys equations (74 - 81). This assumtion tells us that the phase transitions from baryon to quark matter has occurred and, furthermore, the quarks in the star have converted into strange quarks by the weak interactions, eqs. (74 - 78). Which, in turn, lowers the internal Fermi energy of the system because the strange quark is the most massive constituent in quark matter and with this the strange quark degeneracy of *the star* will increase [2].

We then take the initial internal structure of our strange quark star to be a spherical distribution of a homogeneous and isotropic configuration of pure strange quarks in bags of three. Where the bags are not allowed to interact with each other, but the quarks in each bag can interact with the other two. The amount of pressure that is required to inflate such a bag of three quarks in open vacuum is called the MIT bag model constant B. With this description for quark confinement, we now can visualize *the star* as a star of pure strange quarks out to some radius R and surrounded by a soup of up quarks, down quarks, and electrons to maintain overall

*We refere to the words *the star* as a Strange Quark Star that is surrounded by a soup of up and down quarks and electrons that make the whole system charge neutral.

charge neutrality. The Fortran program that will be discussed in Chapter III will look at the stellar dynamics of this pure strange quark star. We constructed our star with 200 consecutive subsystems called zones. Each zone has a thickness Δr for which we must keep track of all the thermodynamic properties. By using equations (86) and (89 - 90) with $i = s$ for each subsystem or zone, along with the May and White numerical integration scheme equations (54 - 62), we can numerically obtain solutions for this set of equations where the expression for the state equation (61) is replaced by equation (89).

We now show the expressions for the $i = s$ case that we will use in the numerical program that calculates the thermodynamic properties. In this case the single independent chemical potential for strange quark matter, eq. (80), is just

$$\mu \equiv \mu_s \ . \tag{91}$$

The total internal energy density, eq. (86), for the system is given by

$$\chi \equiv \rho c^2 + \rho \varepsilon = \Omega_s(\mu) + \mu n_s(\mu) + B \ , \tag{92}$$

which give us the internal pressure from equation (89) for the system as

$$P(\rho, \varepsilon, \mu) = \mu n_s - \rho c^2 - \rho \varepsilon \ , \tag{93}$$

where $f_s(\mu, \mu_s)$ from equation (93) is equal to zero. $\Omega_s(\mu)$ and $n_s(\mu)$ are

$$\begin{aligned} \Omega_s(\mu) \ = \ & -\frac{1}{4\pi^2(\hbar c)^3}\biggl[\mu\sqrt{\mu^2 - m_s^2c^4}\left(\mu^2 - \frac{5}{2}m_s^2c^4\right) \\ & +\frac{3}{2}m_s^4c^8\ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_sc^2}\right) - \frac{2\alpha_c}{\pi}\biggl[3\biggl(\mu\sqrt{\mu^2 - m_s^2c^4} \\ & -m_s^2c^4\ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_sc^2}\right)\biggr)^2 - 2\left(\mu^2 - m_s^2c^4\right) \\ & -3m_s^4c^8\ln^2\left(\frac{mc^2}{\mu}\right) + 6\ln\left(\frac{\tilde{\rho}}{\mu}\right)\biggl(\mu m_s^2c^4\sqrt{\mu^2 - m_s^2c^4} \\ & -m_s^4c^8\ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_sc^2}\right)\biggr)\biggr]\biggr] \end{aligned} \tag{94}$$

and

$$\begin{aligned} n_s(\mu) &= \frac{\mu^2 - m_s^2c^4}{\pi^2(\hbar c)^3}\left[\sqrt{\mu^2 - m_s^2c^4} - \frac{2\alpha_c}{\pi}\right. \\ &\times \left.\left[\mu - \frac{3m_s^2c^4}{\sqrt{\mu^2 - m_s^2c^4}}\ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{\tilde{\rho}}\right)\right]\right] \;, \end{aligned} \tag{95}$$

where $\tilde{\rho}$ is still the renormalization point for massive quarks defined as $\tilde{\rho} = M_N c^2/3 = 313 MeV$ with $M_N c^2$ defined as the nucleon rest energy. Finally the recursion relation for μ, eq. (90), reduces to

$$\mu_{new}\left(\mu^{old}\right) = \frac{\rho c^2\left(1 + \frac{\varepsilon}{c^2}\right) - \Omega_s\left(\mu^{old}\right) - B}{n_s\left(\mu^{old}\right)} \;. \tag{96}$$

Thus μ_{new} is a function of μ^{old} in an iterative process using equation (94) and (95) for a given ρ and ε for each subsystem. ρ is the proper rest mass density which is initally homogeneous and isotropic and ε is the proper specific internal energy per unit mass. Appendix A also shows these expressions in a form that was integrated in the Fortran program of Appendix B.

We already stated that initially the star is completely uniform, meaning that ρ, ε, and μ are the same for every zone in the system initially. If we did not make this initial assumption we would need to know ρ, ε, and μ for all 200 zones and this would be very difficult to do.

So by setting the initial value for the mass, we know what the Schwarzschild radius, R_s, is for this mass and is defined as

$$R_s = \frac{2GM_T}{c^2} \;. \tag{97}$$

Therefore we can select an initial radius for this quark star to be some multiple of R_s (like $2R_s$) and then find the initial mass density of the star for this volume. We choose this initial set up method for two reasons: it gives us an initial radius not unlike that of a neutron star and it also gives us an initial density such that $\rho \sim \rho_N$. When setting the initial value for ε, the numerical integration scheme and the equations of state seem to impose restrictions for this value. For the case $\varepsilon \neq 0$ we found that μ may go negative or at the very least it may drive μ to be less than $m_s c^2$, the rest mass energy of the strange quark. So we choose the smallest initial ε , $\varepsilon = \varepsilon_{\min}$, which avoids this problem. Now that we have the initial values set for ρ and ε, we make a guess as to what the initial value for the chemical potential, μ, is and, by using the above recursion relation equation (96) along with equations (94 and 95), the true initial value of μ can be found. We use this μ to recalculate the initial radius because, for large ε, ε has a tendency to reduce the volume since equation (96) is trying to make $n_s(\mu)\,m_s$ equal to the mass density ρ. With this we can use the numerical integration scheme, eqs. (54 - 62), but replacing equation (61) with equation (93), where μ is given by equation (96) to calculate the numerical results seen in Chapter IV.

CHAPTER III

The Program

Introduction

As we have stated many times before, the stellar dynamics used in the computer program comes from the numerical integration scheme of May and White [6]. The program is in Fortran using the Microsoft Fortran 90 DOS version 5.1 compiler on a 486DX2-66 personal computer with 16 MBy of ROM. This thesis contains two Fortran programs. The strange quark star program is in Appendix B and a Fortran program using May and White's equations of state can be found in Appendix C.

Since Fortran can not use half integers for array assignments as seen in May and White's equations [6], we will adopt the following assignment configuration. All the shells have even j indices and all the zones have odd j indices where $j = -1, 0, 1, \cdots, JJ + 1$ with $j = 0$ defined as the origin of the star's coordinate system and $j = JJ$ defined as the outer most shell of the star at a radius of $r = R$. The regions between any two consecutive shells j to $j + 2$ are called the zones and have an initial thickness of Δr. Therefore we divide the star into 200 consecutive zone-shell sets and initially all the zones are of equal thickness Δr. Thus the whole star can be visualized like this

$$-1 \bullet_0 1 \;{}_{2}\Big)\, 3 \;{}_{4}\Big)\, 5 \;{}_{4}\Big)\, 7 \;{}_{8}\Big) \cdots\cdots\cdots JJ-1 \;{}_{JJ}\Big)\, JJ+1$$

where the central numbers $-1, 1, 3, 5, \cdots JJ - 1, JJ + 1$ are the zones, the lower numbers $2, 4, 6, 8, \cdots JJ$ are the shells and $JJ = 400$, which implies that there are 200 zones and 200 shells.

In order for us to use the May and White numerical integration scheme in our program, we first have to make an array assignment conversion since May and White use half integer indices in there equations. Thus the array assignments that we have set for the time stepping are all centered on $t3$ which we defined to be the most current time. Therefore we set $t1 \equiv t3 - \Delta t$ and $t5 \equiv t3 + \Delta t$ where Δt is the time step, either $t2$ or $t4$. The array assignments that we impose on both the shells and the zones are always shell centered. This means that $j \equiv j3$ is the current shell that is being looked at by the computer, where $j = 2, 4, 6, \cdots JJ$ by steps of 2. We now describe the array assignments for the zone and shell that lies above and below this central shell $j \equiv j3$ in the following manner. The shell below the central shell $j3$ is defined as $j1 \equiv j3 - 2$. The zone which lies between these two shells is defined as $j2 \equiv j3 - 1$. Likewise for the zone and shell above the centeral shell $j3$. The shell which lies above the centeral shell $j3$ is defined as $j5 \equiv j3 + 2$. The zone which lies between these two shells is defined as $j4 \equiv j3 + 1$. So when looking at the programs in Appendies B and C, the variable arrays will look something like

$$variable(t3, j2),\ variable(t2, j3),\ \cdots\cdots \text{ and so on.}$$

where $j3 \equiv j = 2, 4, 6, \cdots JJ$ by steps of 2. At the beginning of each program is a complete list of the nomenclature that is used throughout that program with the dimensions of each array used and the units of each variable. For the reader's convience, we have put the May and White equation numbers on the line above the respective Fortran equation line in each program.

Testing The Program

The Free-Fall Test

Our first test on the computer code is a gravitational free-fall collapse using May and White's equations of state and comparing our numerical results with their results. We perform the free-fall test using May and White's equations of state with the initial conditions of $\varepsilon = 0$ and $\omega = a = 1$ everywhere inside the star, implying that $P = Q = 0$ for all times (Q represents an artificial viscosity as described by May and White and is initially set to zero). Therefore the numerical integration scheme drastically simplifies in the free-fall limiting case with the pressure defined as

$$P = P(\rho, \varepsilon) = \frac{2}{3}\rho\varepsilon \ . \tag{98}$$

The equations (54 - 62) reduce to

$$u_t = -\frac{mG}{R^2} \ , \tag{99}$$

$$R_t = u \ , \tag{100}$$

$$\frac{(\rho R^2)_t}{\rho R^2} = -\frac{u_\kappa}{R_\kappa} \ , \tag{101}$$

$$\epsilon_t = 0 \ , \tag{102}$$

$$\frac{(\omega)_\kappa}{\omega} = 0 \ , \tag{103}$$

$$m = 4\pi \int_0^\kappa \rho R^2 R_\kappa \, d\kappa \ , \tag{104}$$

$$\Gamma = 4\pi\rho R^2 R_\kappa \ , \tag{105}$$

$$P = 0 \ , \tag{106}$$

and

$$\omega = 1 \ , \tag{107}$$

which are the Newtonian equations for gravitational free-fall. From this a theoretical solution can be formulated, which is

$$\frac{\pi}{2} - t\sqrt{\frac{8\pi G\rho_\circ}{3}} = \frac{\pi}{2} - \left[\sqrt{\zeta}\sqrt{1-\zeta} + \arcsin\sqrt{1-\zeta}\right] \ , \tag{108}$$

where we define ζ as

$$\zeta \equiv \frac{R(\kappa, t)}{R(\kappa, 0)} \tag{109}$$

and the initial mass density is

$$\rho_\circ = \frac{3\left(m(\kappa, t) - m(\kappa, 0)\right)}{4\pi R^3(\kappa, 0)} \ . \tag{110}$$

For a gravitational free-falling star, the characteristic collapse time, τ_c, is

$$\tau_c = \sqrt{\frac{3\pi}{32G\rho_\circ}} \ , \tag{111}$$

which is the total time it would take for a body like a star to collapse to a single point under its gravitational field if it lacked all internal energy and pressure.

In figure 1, we have plotted the theoretical solution to equations (108) against equation (109) along with our numerical results using May and White's Free-Fall equations and achieved outstanding accuracy in our numerical results, as can be seen from the relative %Error located at the top of figure 1. We found that the theoretical solution and the computer code are both very sensitive to the accuracy of π, as one can see from both the figure and in equation (108). We can also see that if the value of π used is not accurate enough, the theoretical solution will curve above the true position of this theoretical solution. Therefore we adopted a value of π equal to 3.14159265358979323846 which gave us the theoretical curve that is shown in figures 1, 2, and throughout the figures located in Chapter IV. May and White found it necessary to invoke an adjustment on all the times of the amount of $t = t + 0.48\Delta t^{0+1}$ to move their numerical results up to the theoretical solution, where Δt^{0+1} is equal to the initial $\Delta t \equiv t2 \equiv t4$ in our program. We believe that the main reason why we need not adjust our times is because we were using a more accurate value of π.

The Non-Free-Fall Test

The second test that we performed on the computer code was the gravitational non-free-fall collapse test using May and White's equations of state and comparing our numerical results with those of May and White. Here we take the initial conditions $\varepsilon \neq 0$ and $\omega \neq a \neq 1$ which imply that $P \neq Q \neq 0$, so we must use the pressure defined by equation (98). Therefore we must use the full expressions for equations (54 - 62).

May and White used a γ-law relationship for their non-free-fall computations with $\gamma = \frac{5}{3}$ which gives the pressure as $P = \frac{2}{3}\rho\varepsilon = \frac{2}{3}\chi$. They initially set the internal energy per unit mass to be $\varepsilon_\circ = 1.92 \times 10^{-5}c^2$ and the initial radius as $R_\circ = \dfrac{2M_TG}{6.2 \times 10^{-3}c^2} \simeq 1004R_s$ for an initial mass of $M_T = 21M_\odot$ where R_s is the Schwarzschild radius. The mass density is simply $\rho = \dfrac{3M_T}{4\pi R_\circ^3}$. Using this setup, we have plotted our computational results for several mass regions in the star. This means that we have kept track of several volumes that contains 100%, 85.7%, 42.2%,

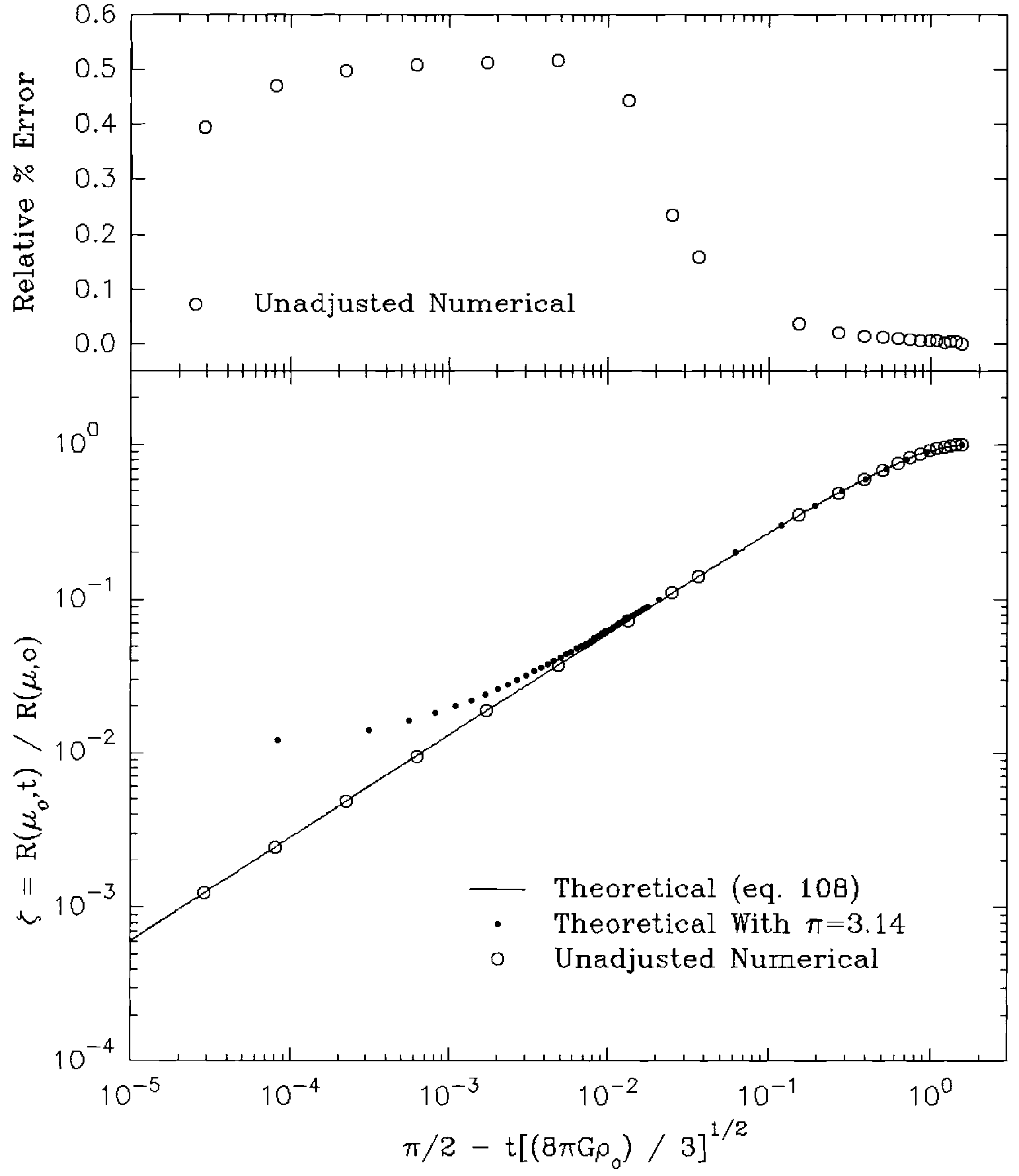

Figure 1. A free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $21M_\odot$ non-quark star where $\Delta t^{0+1} = 0$ for the adjustments on t. The relative error of the numerical data is shown above.

25.0%, and 10^{-4}% of the total mass of the star. These results can be seen in figure 2 along with the theoretical solution for Free-Fall, eq. (108). We found that our numerical results in figure 2 are very close to May and White's results [6].

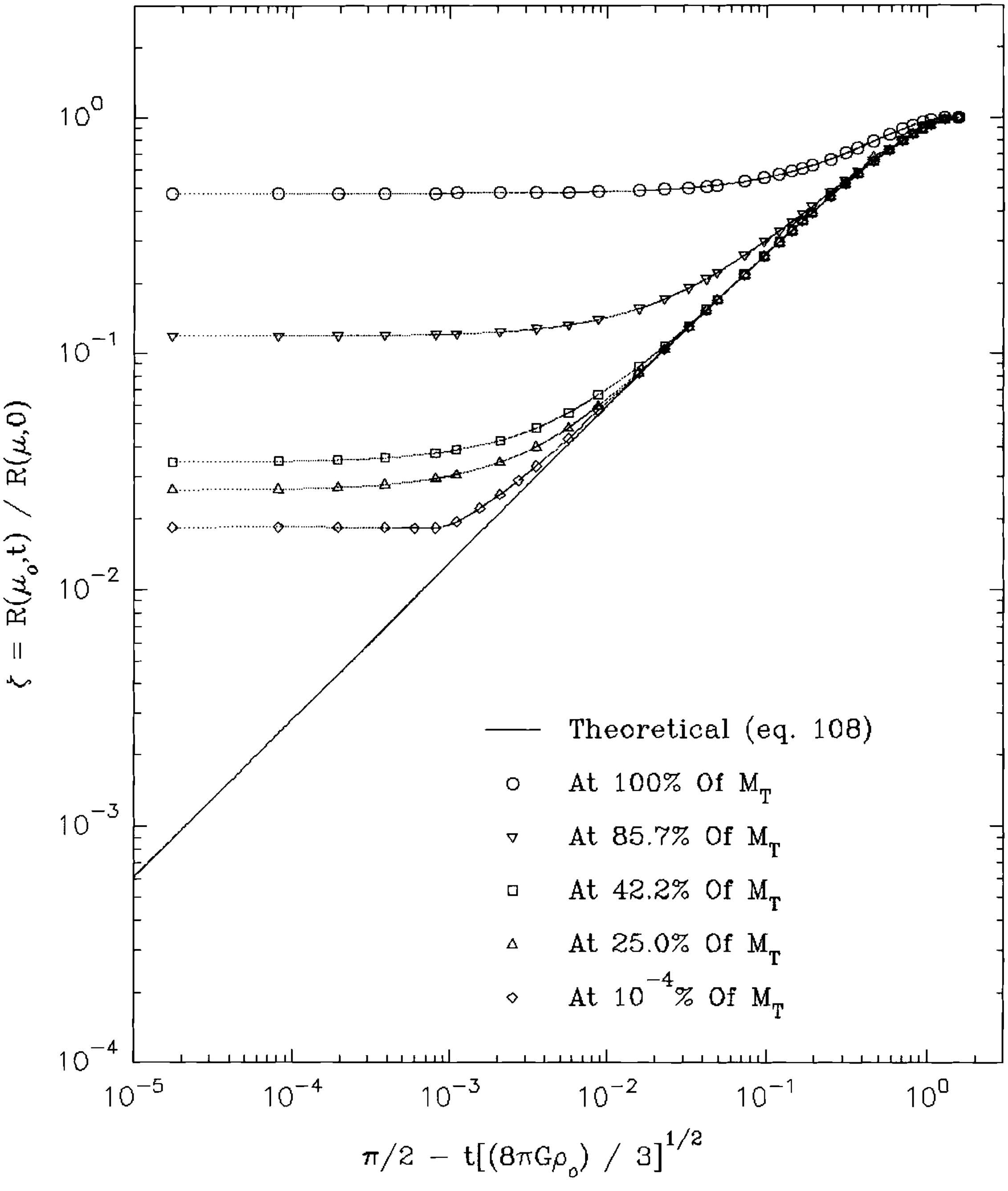

Figure 2. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $21M_\odot$ non-quark star where $\Delta t^{0+1} = 0$ for the adjustments on t.

CHAPTER IV

Results

Introduction

This chapter gives the results that were obtained with the Fortran program in Appendix B using the initial conditions for the strange quark star model (for the non-free-fall case) that was discussed in Chapter III. When looking at the strange quark star in free-fall, we find that its behavior is the same as in figure 1, since there are no equations of state that are being used in either case. This implies that there is no internal pressures or energies throughout either star. The only thing that is different between the two free-fall cases is that the initial density is much larger in the stange quark star which makes the free-fall collapes time much faster. We now take a look at the non-free-fall case for a stange quark star and at the end one will find our closing remarks.

A Non-Free-Fall Look Inside A Strange Quark Star

We looked for stability for the case where $B = 60MeV/fm^3$ and $\alpha_c = 0.45$ for $m_s = 150MeV/c^2$. We also took a brief look at a strange quark star with the same B and α_c but with $m_s = 200MeV/c^2$ and $m_s = 250MeV/c^2$. These results and results from $m_s = 150MeV/c^2$ are found in tables V - VII. When looking for stability we found that in most cases the star would oscillate as it either collapsed or expanded. These oscillations have amplitudes from a few tenths of a meter to several meters. We associate these oscillations with density shock waves that are propagating throughout the star.

A word of caution is needed when looking at the collapse or expansion of this type of star. Since the stars are so small and the internal densities are so great one finds that the time scales are very small, around a microsecond or less for each consecutive time step, because the dynamics force the time step to be this small. This is to be expected, for when we look at the characteristic collapse time, $\tau_c = \sqrt{\frac{3\pi}{32G\rho_\circ}}$, we find that the total collapse time gets shorter as the density increases. If one uses $\rho_\circ = \rho_N \equiv 1.4 \times 10^{-4} M_\odot / km^3$, one finds $\tau_c = 1.25895 \times 10^{-4} s$. This is the total time it would take a star to collapse to a point under free-fall conditions (recall that for a quark star we have $\rho_\circ > \rho_N$).

Tables V - VII show our results that were received from the strange quark star Fortran program of Appendix B for different strange quark masses and for different initial stellar masses. The initial setup values are on the left-hand side of these tables and the ending values after an elapsed time given by ETIME are shown on the right-hand side, where ρ_{20}/ρ_N is the density ratio in the 20^{th} zone of the star. In most cases our program can't go pass this "ETIME" for these initial star masses because the star evolves beyond the capability of this program with the equations of state that are being used.

TABLE V.

Results for a strange quark star with a strange quark mass of $m_s = 150 MeV/c^2$, strong coupling constant of $\alpha_c = 0.45$, and bag constant of $B = 60 MeV/fm^3$

M_{TOT} $(M_\odot)$	ε_{min} (c^2)	R_i (km)	$\bar{\rho}_i/\rho_N$	μ (MeV)	ETIME $(10^{-5}s)$	ρ_{20}/ρ_N	R_f (km)
1.70	0.90	10.042	1.507	493.1	10.39	5.83	11.857
1.80	0.75	10.633	1.460	477.6	10.999	25.73	11.985
1.90	0.65	11.220	1.388	463.2	10.278	22.17	12.103
2.00	0.50	11.814	1.379	449.7	9.661	25.38	12.18
2.10	0.40	12.405	1.340	437.0	9.299	28.66	12.266
2.20	0.25	12.995	1.367	425.0	8.625	27.13	12.34

When working with this program we found that ε was indeed bounded for workable solutions to our equations of state, eq. (92) and (93), in the Fortran program of Appendix B. We also found that ε is directly dependent on the choice of the initial mass for the star; as the initial mass increases, ε seems to decrease as can be seen in these tables. In general, ε seems to have the following range $(0.05c^2 \leq \varepsilon \leq \sim 1.0c^2)$. For $\varepsilon < 0.05c^2$ the program would crash due to the fact that ε would turn negative in the outer most region of the star for most initial masses from $(1.0M_\odot \leq M \leq 2.5M_\odot)$. We believe the reason for this is that the star is collapsing faster than the program can handle and this produces non-physical results, thus the star seems to implode when $\varepsilon < 0.05c^2$. For $\varepsilon > 1.0c^2$ the opposite seems to happen. The star has just too much internal energy and can not stay together so the star explodes. To avoid the ε problem we choose an ε which is the minimum value that could be assigned to it to make the star either expand or collapse steadily. This in turn makes the initial internal mass density, ρ, as large as we can get it (*i.e.*, $\rho_\circ > \rho_N$), because the density will decrease as ε_{min} increases eventually making $\rho_\circ < \rho_N$ which makes the quark matter equations of state invalid. With the density decreasing as the ε is increased, we find that initial radius of the star decreases. This seems to imply that there is a maximum value for ε in the initial setup of a strange quark star.

We plotted several sets of data concerning a strange quark star with a strange quark mass of $150MeV/c^2$ for initial stellar masses from $1.7M_\odot$ to $2.2M_\odot$ (table V). The results are shown in figures 3 - 16. When comparing a 1.8 $M_\odot$ strange quark star (figures 4 - 7) to a $2.1M_\odot$ strange quark star (figures 12 - 15), we see two different types of dynamics. One can see that the $2.1M_\odot$ strange quark star has a collapsed core and the rest of the star appears to be collapsing as well, but slowly with respect to the time scale. The $1.8M_\odot$ strange quark star does not have a collapsed core; in fact, the core seems to be slowly expanding and the star may find a point of stability which is unknown.

We found that a $1.7M_\odot$ strange quark star (figure 3) evolves in a similar manner to the $1.8M_\odot$ strange quark star (figure 4). Likewise a $1.9M_\odot$ strange quark star (figure 8) seems to evolve in a similar manner to the other two stars at $1.7M_\odot$ and

TABLE VI.

Results for a strange quark star with a strange quark mass of $m_s = 200MeV/c^2$, strong coupling constant of $\alpha_c = 0.45$, and bag constant of $B = 60MeV/fm^3$

M_{TOT} $(M_\odot)$	ε_{min} (c^2)	R_i (km)	$\bar{\rho}_i/\rho_N$	μ (MeV)	ETIME $(10^{-5}s)$	ρ_{20}/ρ_N	R_f (km)
1.70	0.85	10.040	1.550	496.1	11.12	22.47	11.62
1.80	0.80	10.633	1.419	480.9	9.770	3.597	11.78
1.90	0.63	11.224	1.406	466.8	10.02	26.33	11.823
2.00	0.45	11.814	1.426	453.6	9.015	23.90	11.911
2.10	0.35	12.405	1.390	441.2	8.755	30.05	11.98
2.20	0.20	12.996	1.420	429.6	8.167	29.14	12.06

TABLE VII.

Results for a strange quark star with a strange quark mass of $m_s = 250MeV/c^2$, strong coupling constant of $\alpha_c = 0.45$, and bag constant of $B = 60MeV/fm^3$

M_{TOT} $(M_\odot)$	ε_{min} (c^2)	R_i (km)	$\bar{\rho}_i/\rho_N$	μ (MeV)	ETIME $(10^{-5}s)$	ρ_{20}/ρ_N	R_f (km)
1.70	0.80	10.04	1.59	499.1	10.067	36.15	11.05
1.80	0.65	10.63	1.55	484.4	9.114	26.46	11.16
1.90	0.50	11.22	1.53	470.7	8.590	30.89	11.24
2.00	0.35	11.81	1.53	458.6	8.062	29.64	11.34
2.10	0.20	12.40	1.56	446.2	7.684	32.80	11.40
2.20	0.10	13.00	1.55	435.1	7.425	30.82	11.44

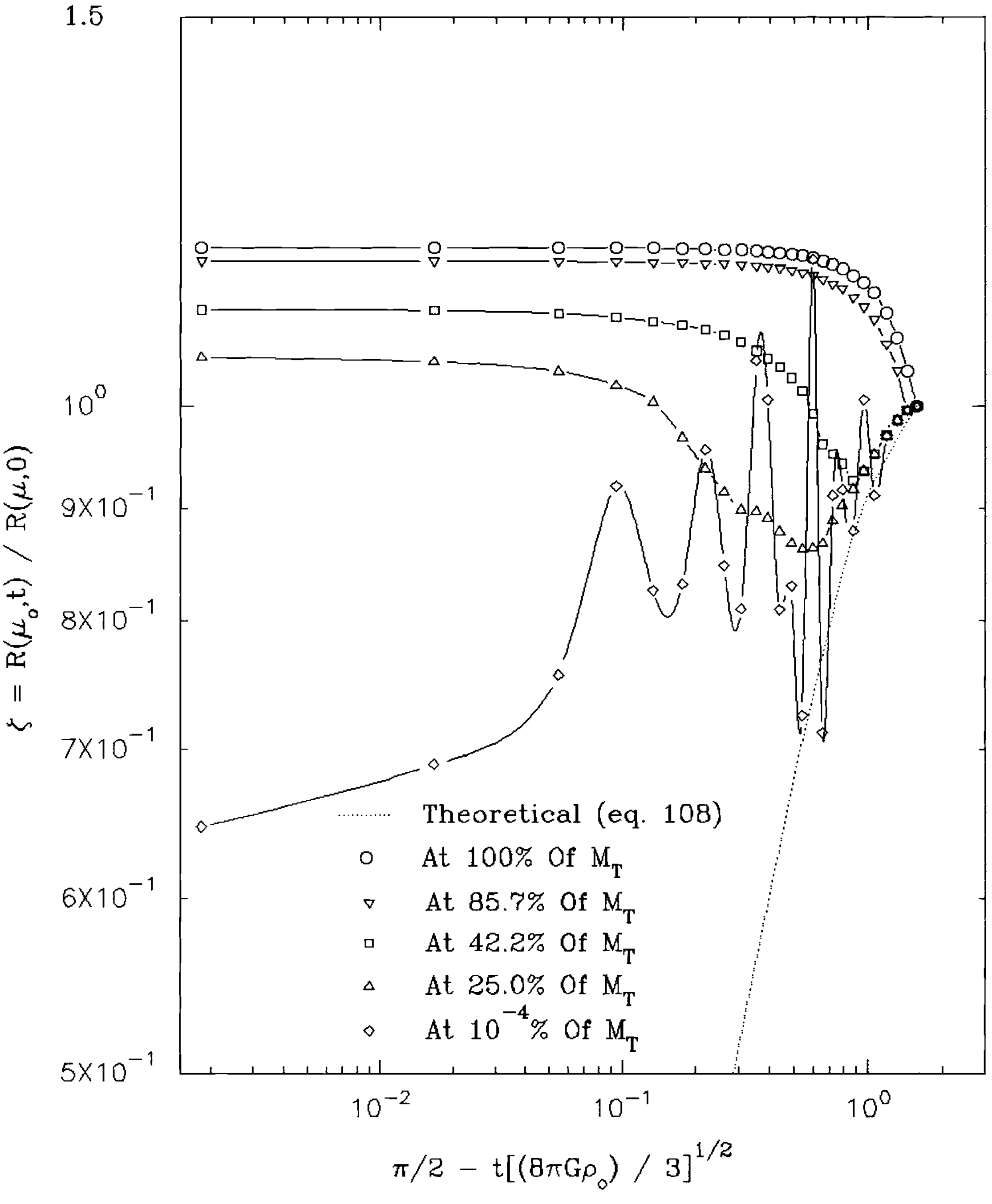

Figure 3. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t(8\pi G\rho_0/3)^{1/2}$, for a $1.7M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.9c^2$.

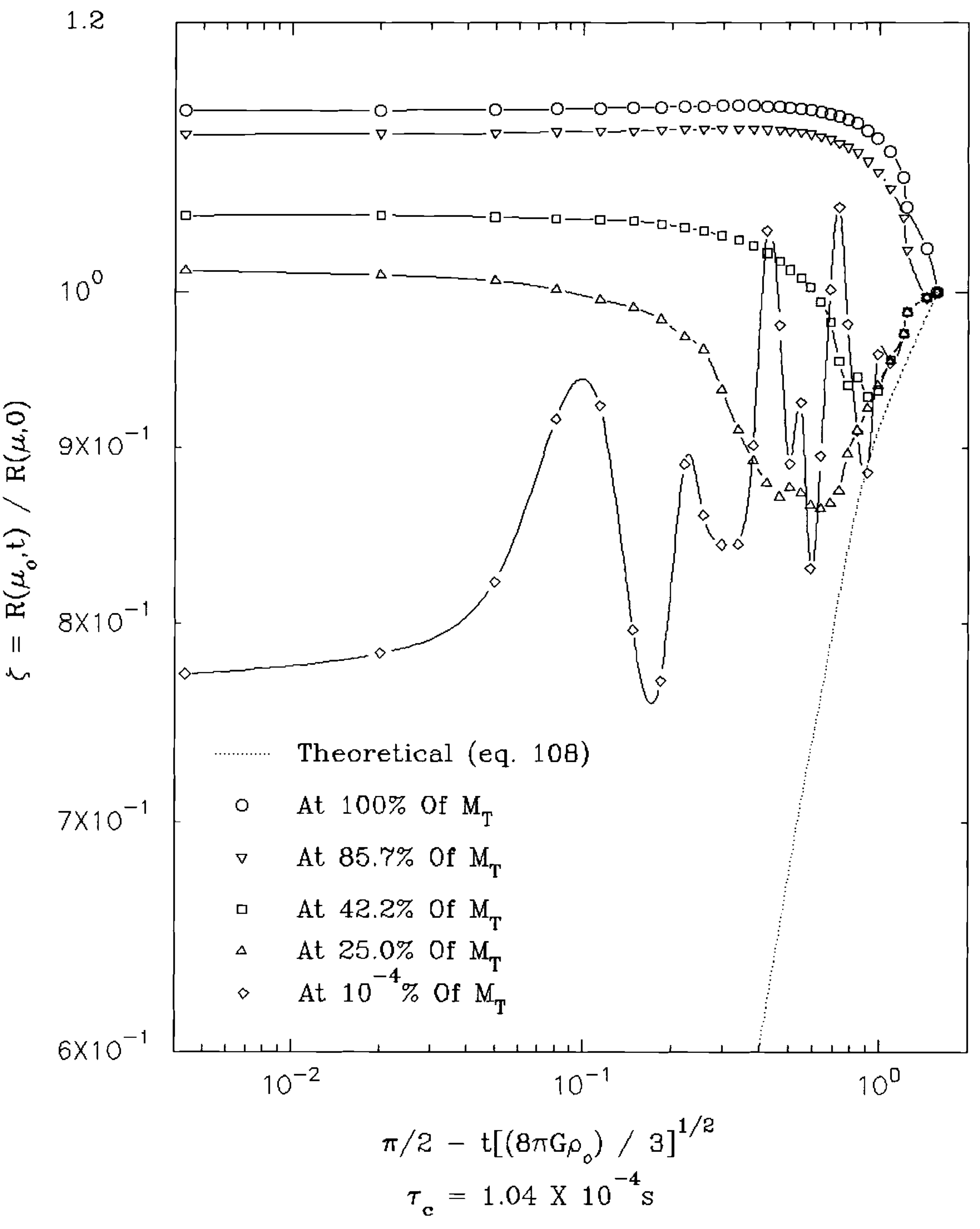

Figure 4. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $1.8M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.75c^2$.

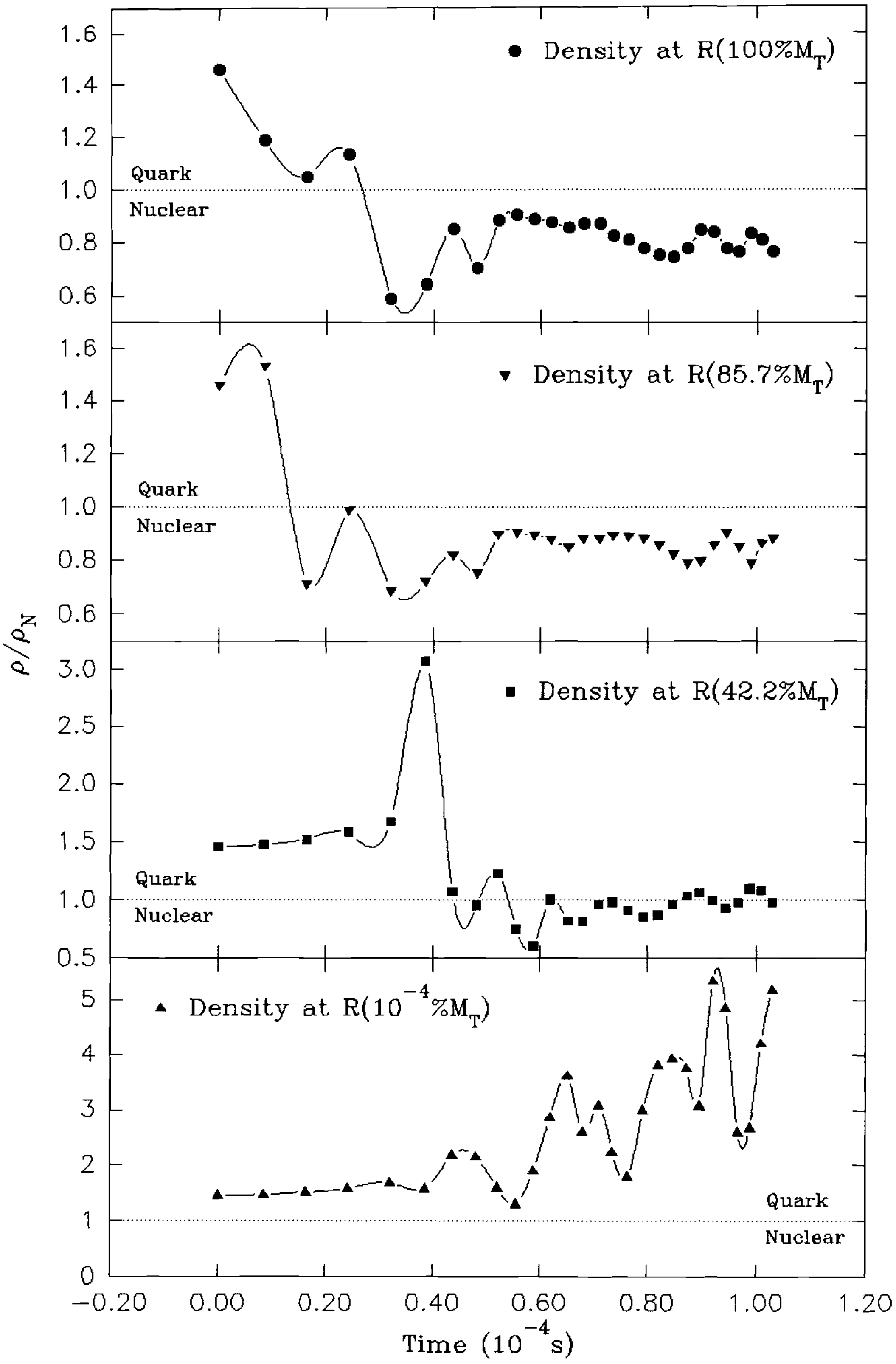

Figure 5. A non-free-fall plot of the density ratio ρ/ρ_N v.s. time for a $1.8M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.75c^2$.

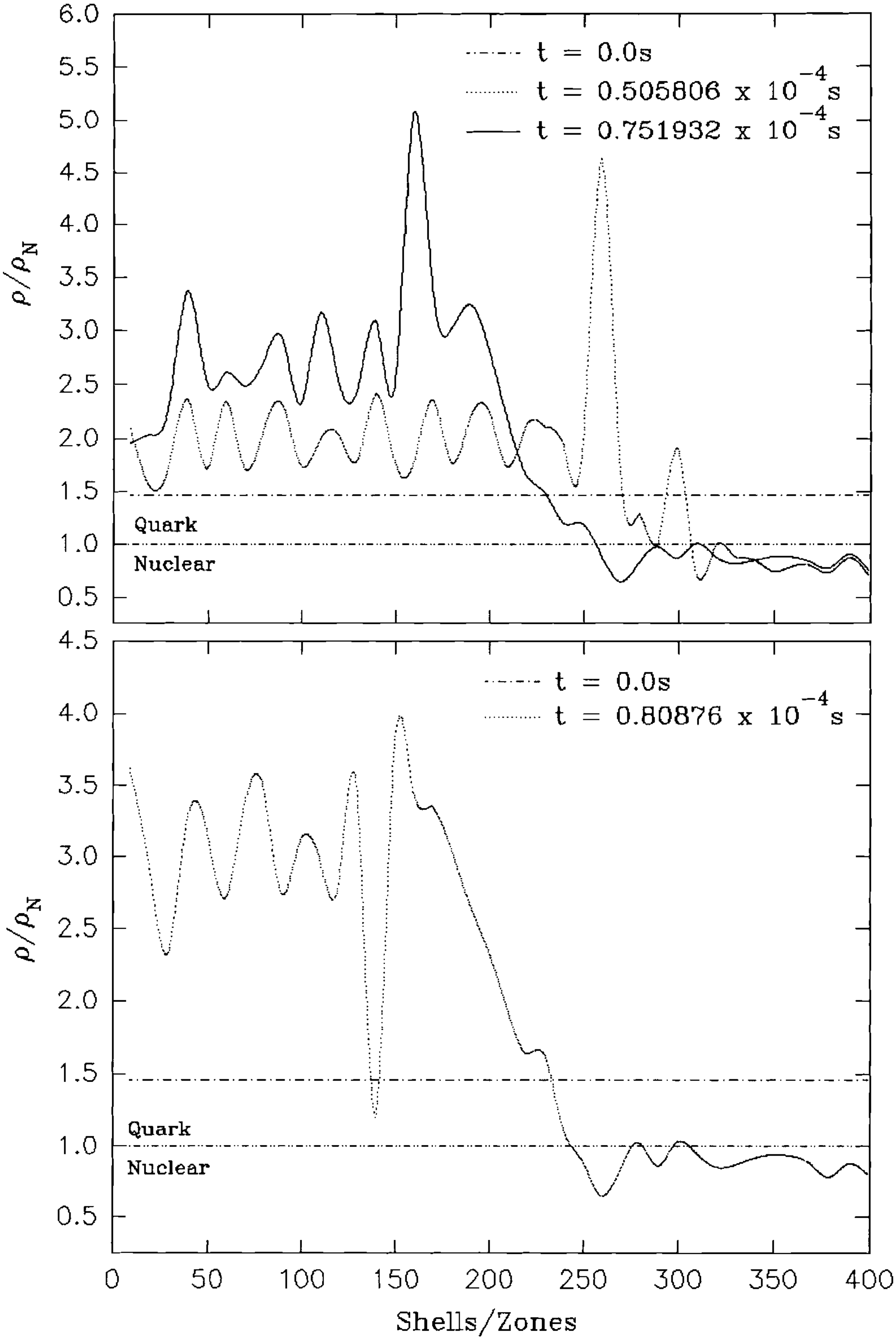

Figure 6. A non-free-fall plot of the density ratio ρ/ρ_N for specific times as a function of the zones for a $1.8M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.75c^2$.

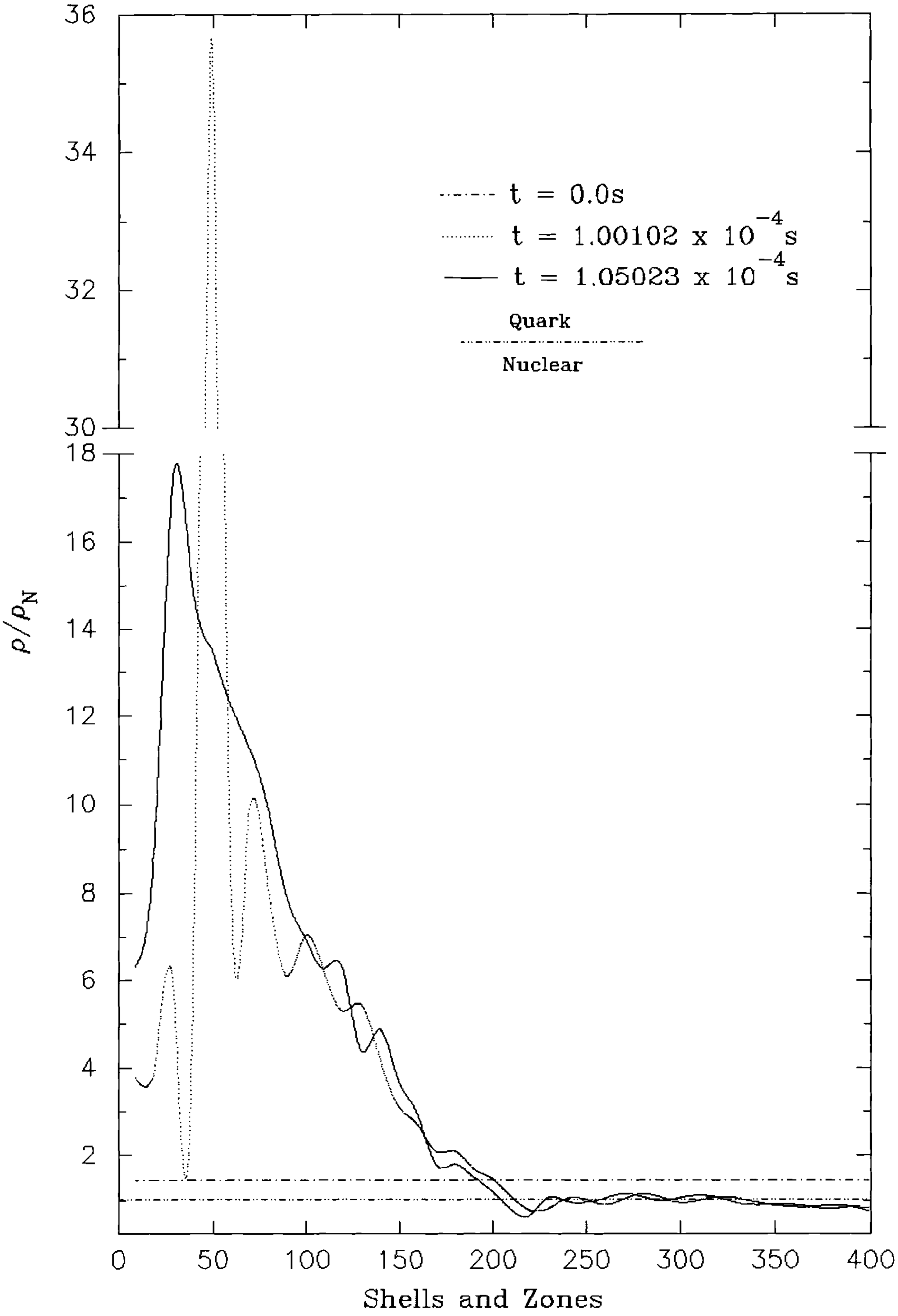

Figure 7. A non-free-fall plot of the density ratio ρ/ρ_N for specific times as a function of the zones for a $1.8M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.75c^2$.

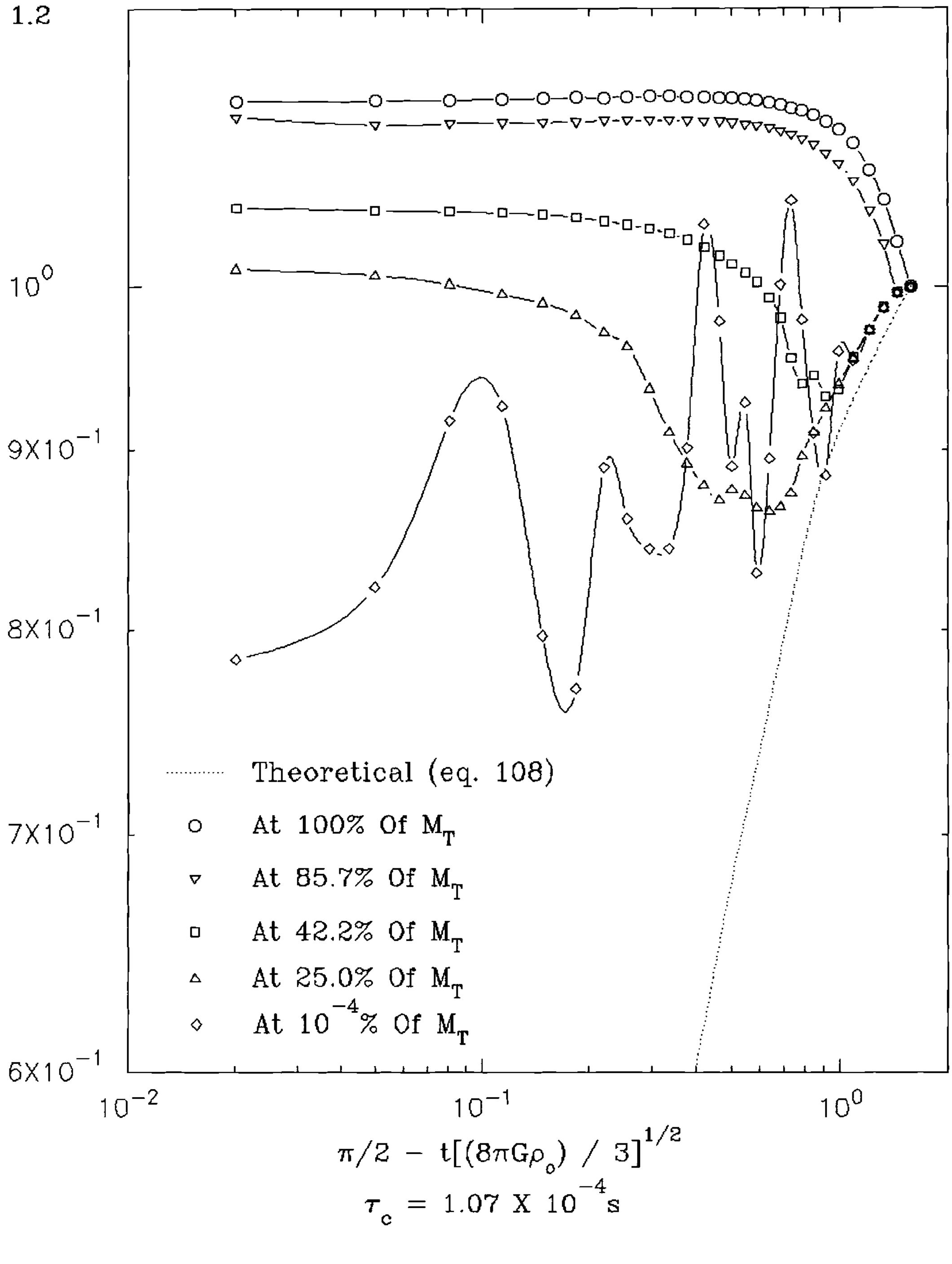

Figure 8. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $1.9M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.65c^2$.

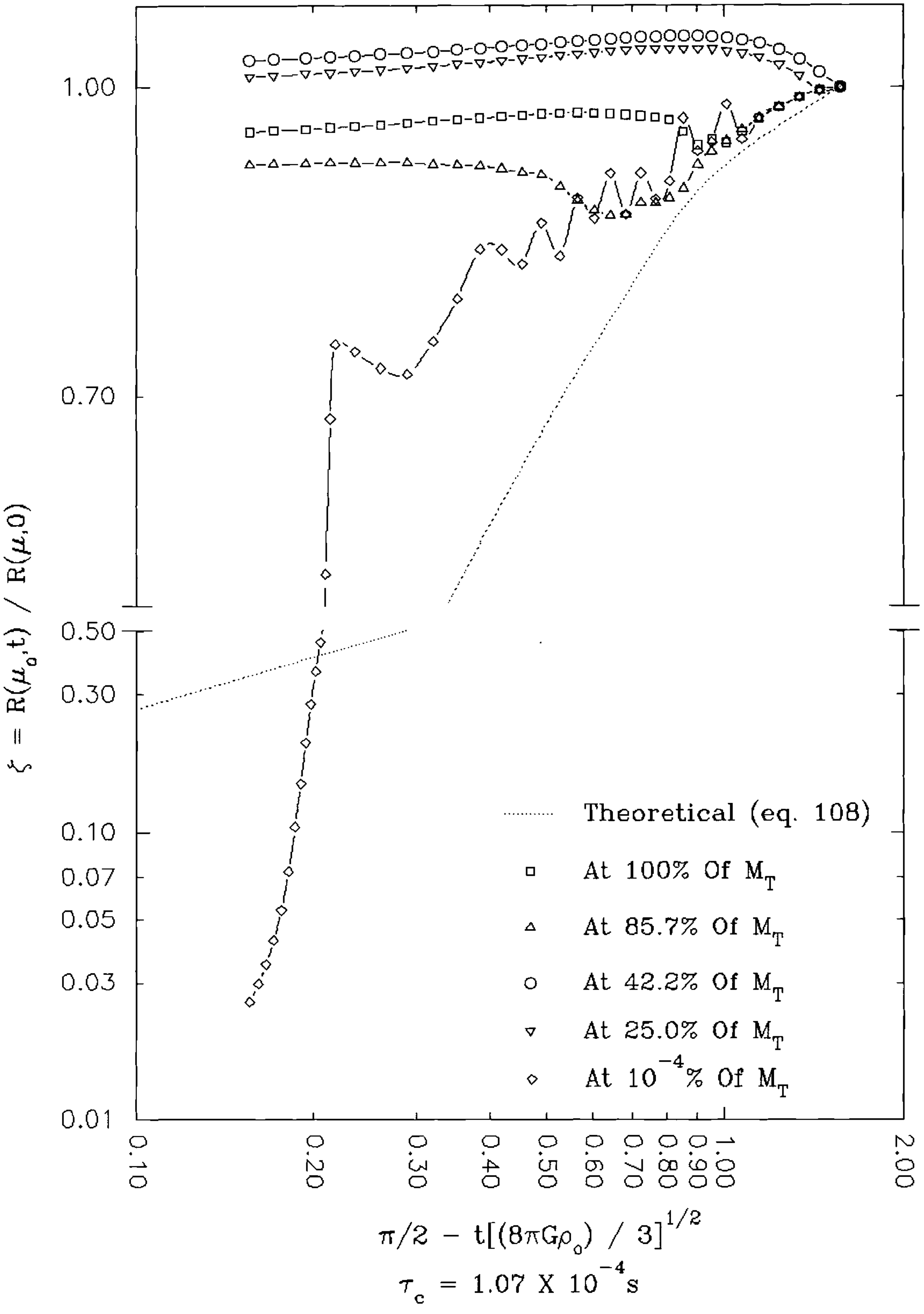

Figure 9. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $2.0M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.5c^2$.

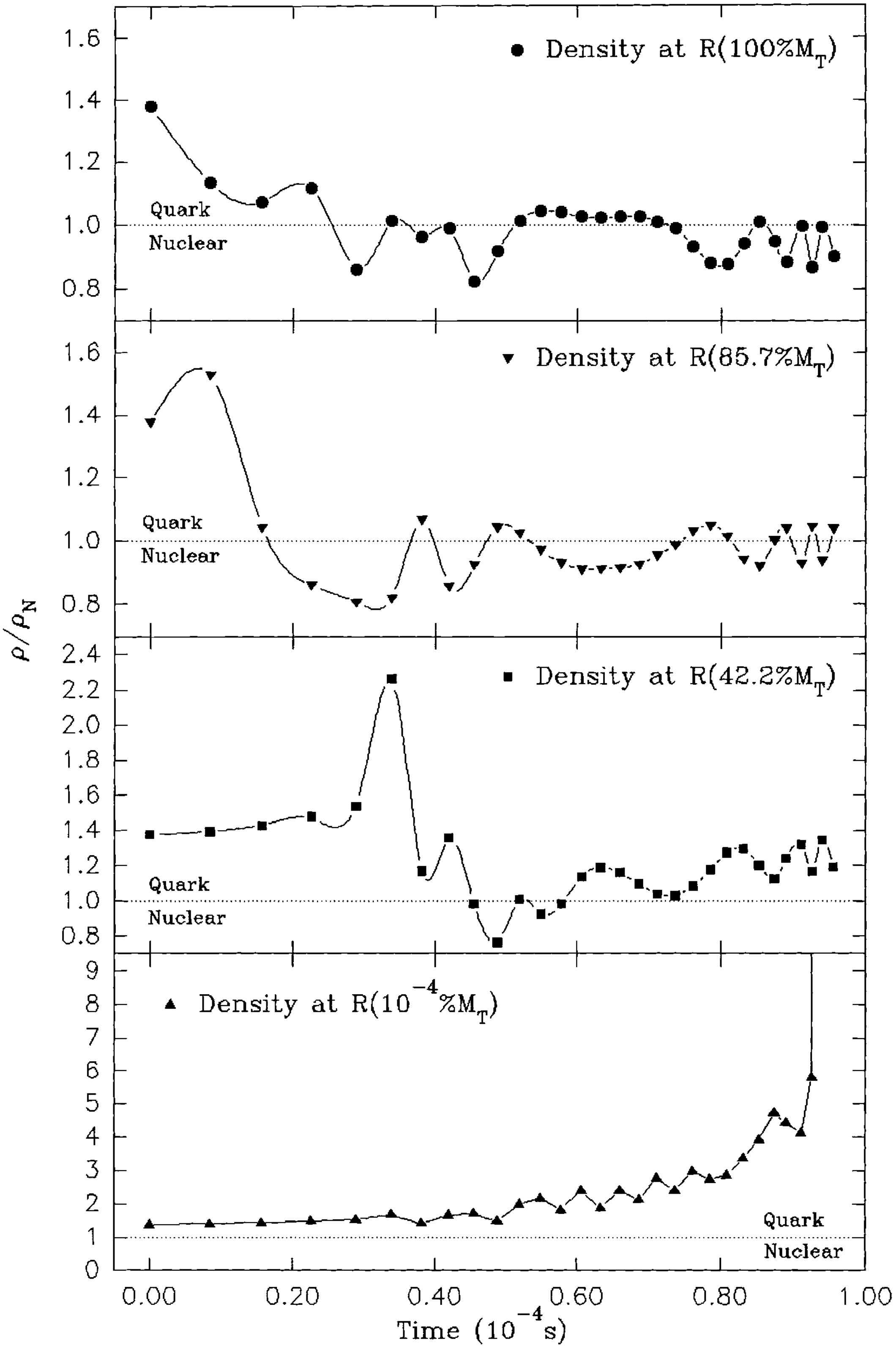

Figure 10. A non-free-fall plot of the density ratio ρ/ρ_N v.s. time for a $2.0M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.5c^2$.

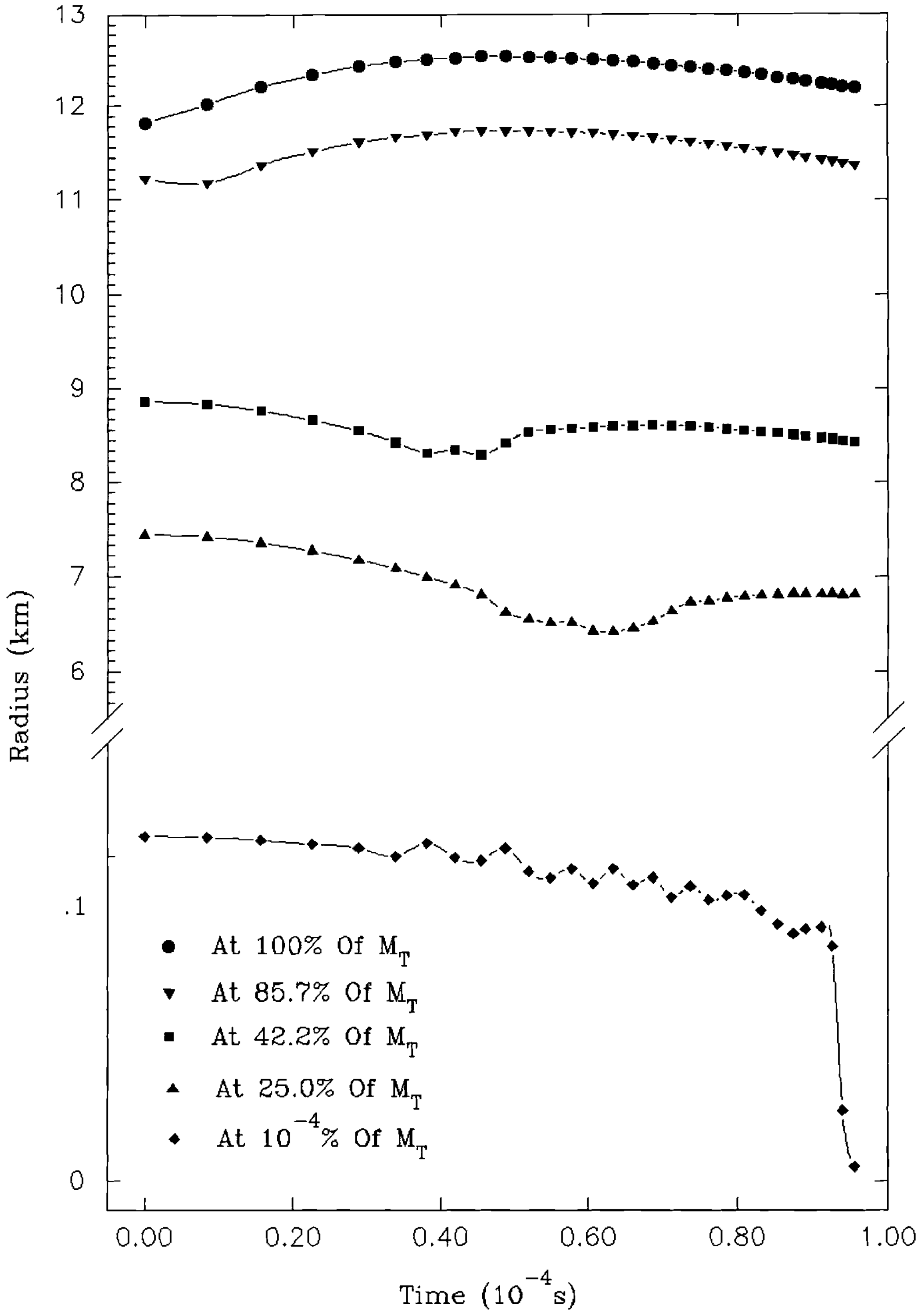

Figure 11. A non-free-fall plot of radius v.s. time for a $2.0M_{\odot}$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.5c^2$.

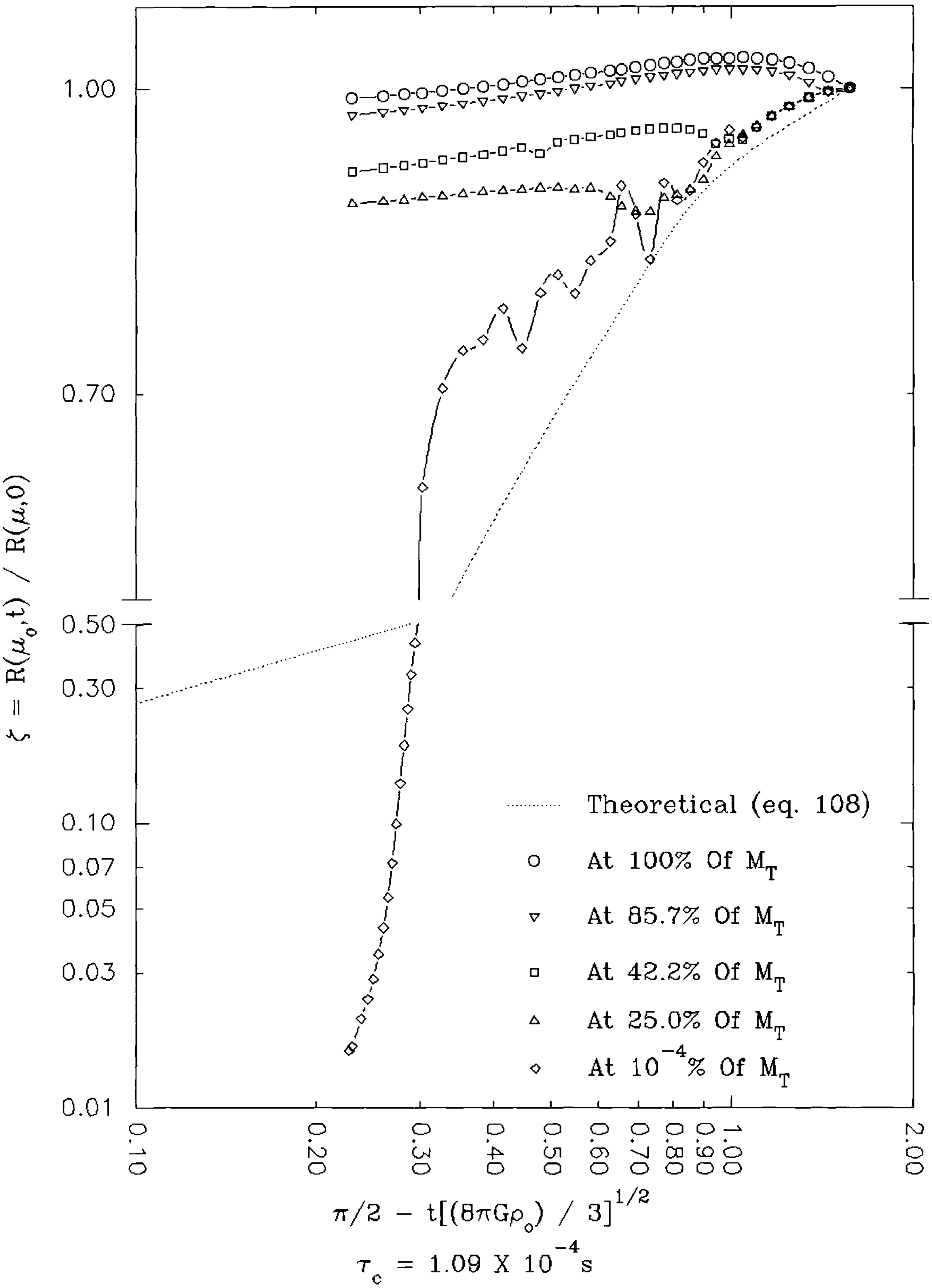

Figure 12. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $2.1M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.4c^2$.

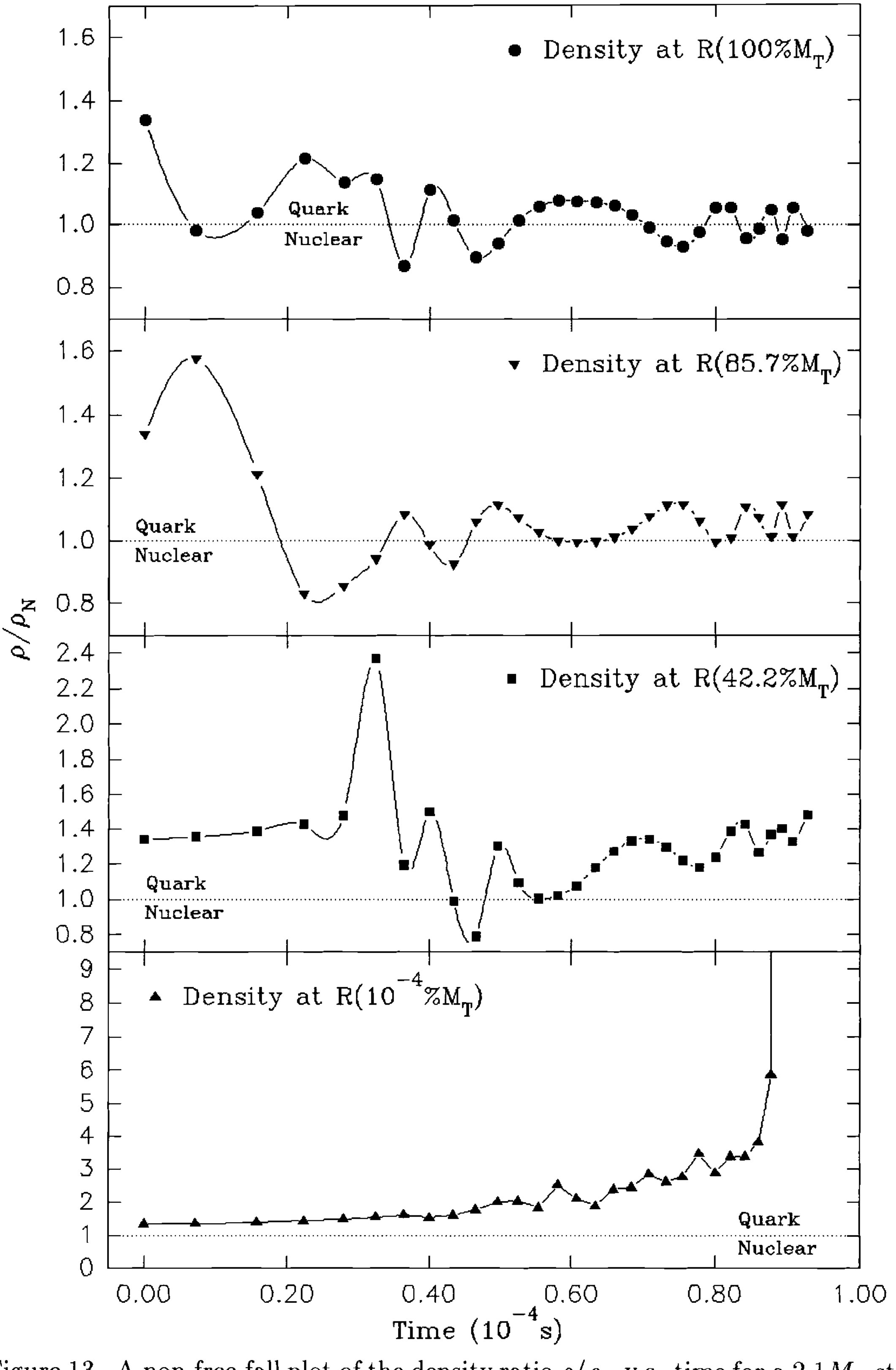

Figure 13. A non-free-fall plot of the density ratio ρ/ρ_N v.s. time for a $2.1M_{\odot}$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.4c^2$.

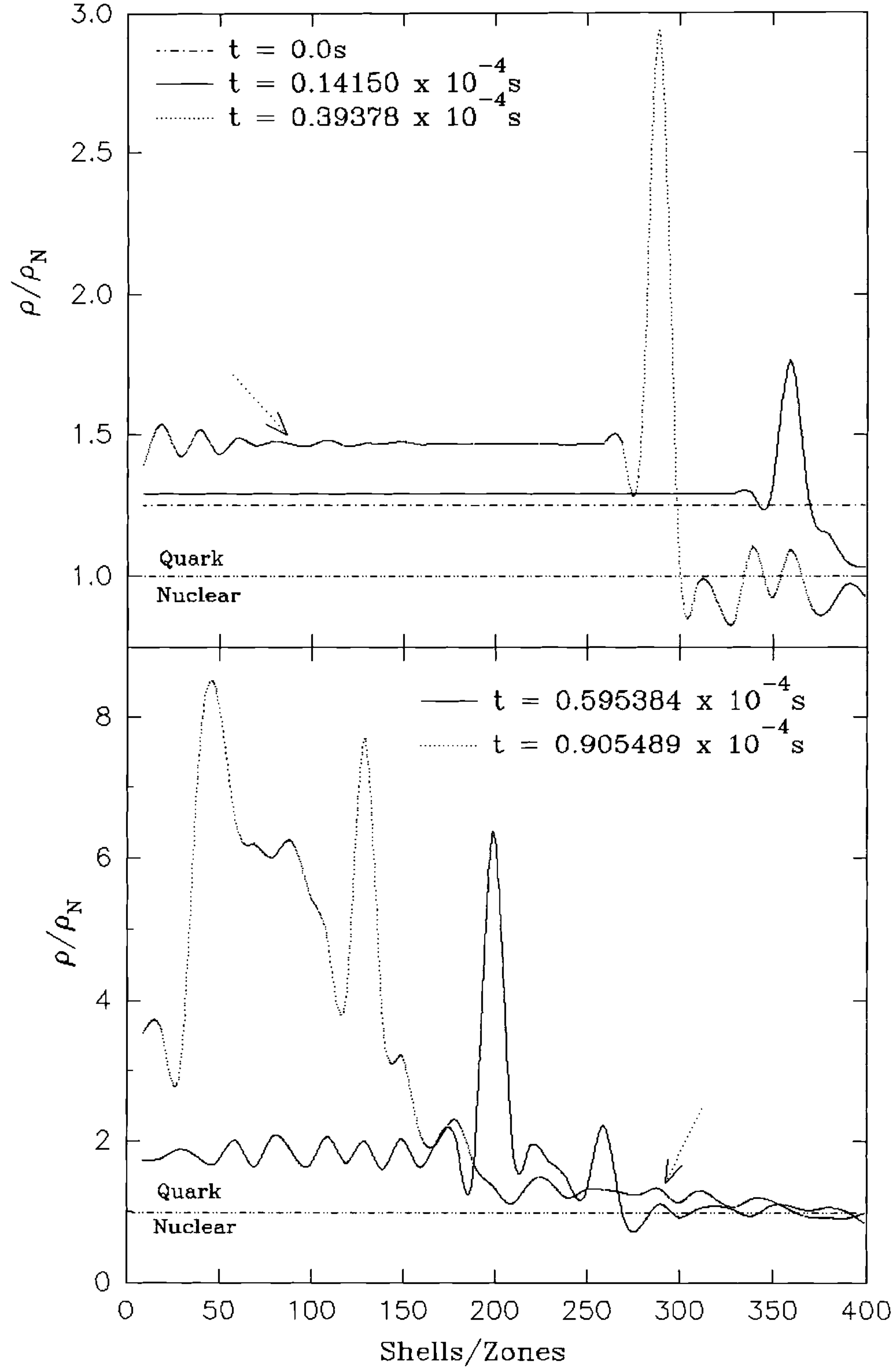

Figure 14. A non-free-fall plot of the density ratio ρ/ρ_N for specific times as a function of the zones for a $2.1M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.4c^2$.

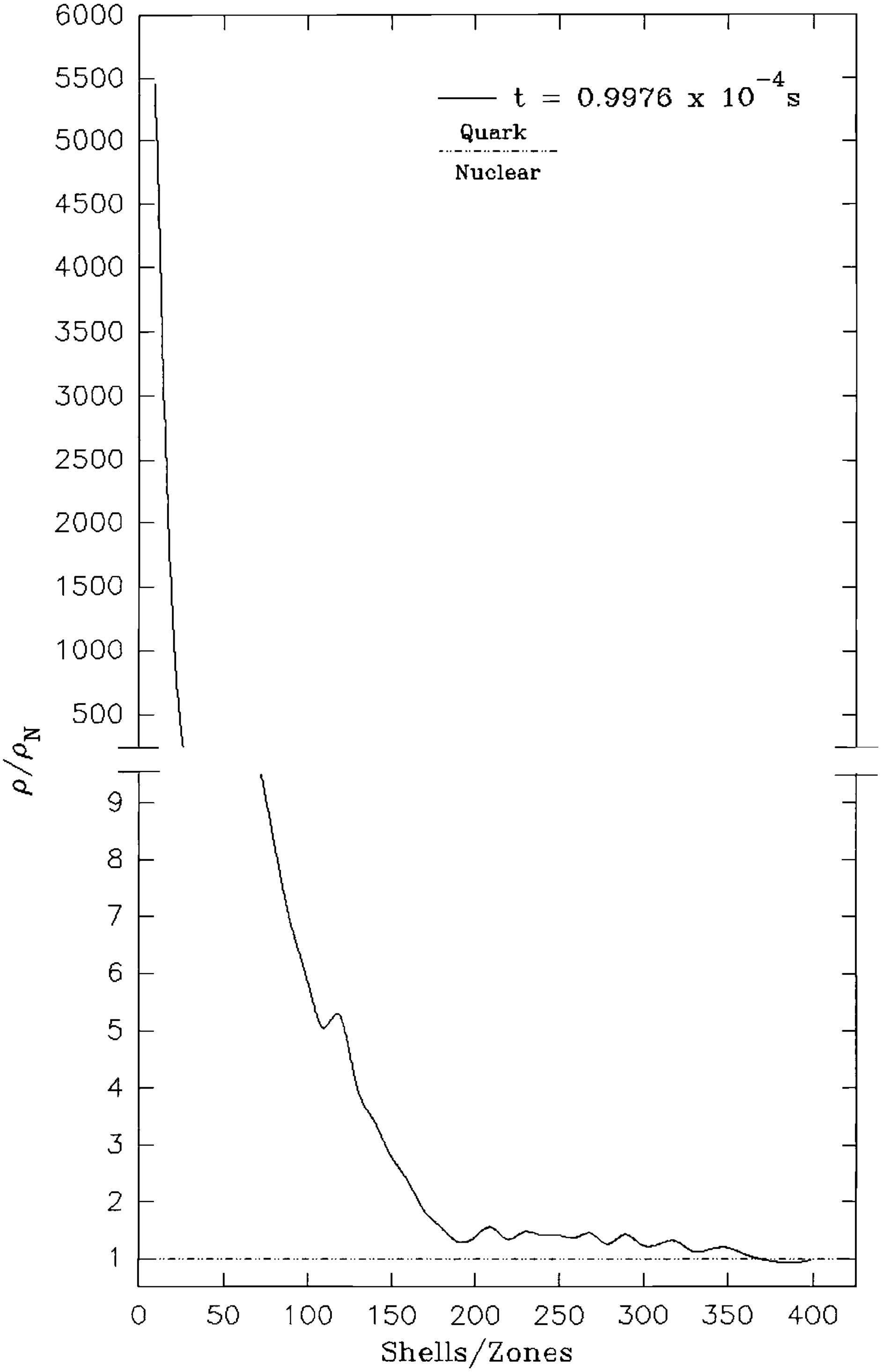

Figure 15. A non-free-fall plot of the density ratio ρ/ρ_N for specific times as a function of the zones for a $2.1M_{\odot}$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.4c^2$.

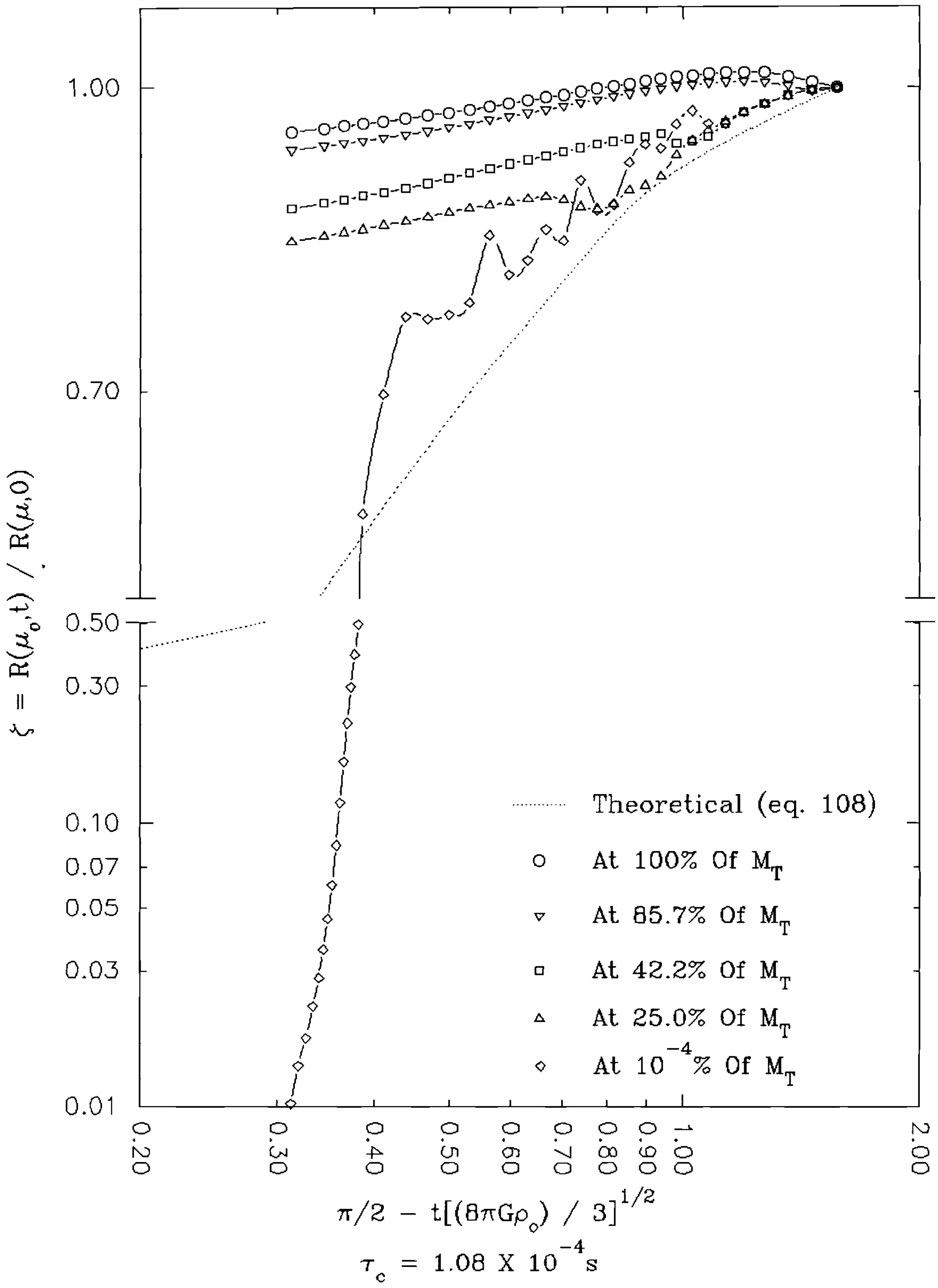

Figure 16. A non-free-fall plot of the radius ratio, ζ, v.s. a dimensionless scaled time, $\pi/2 - t\,(8\pi G\rho_0/3)^{1/2}$, for a $2.2M_\odot$ strange quark star with a strange quark mass, $m_s = 150MeV/c^2$, and minimum internal energy, $\varepsilon_{min} = 0.25c^2$.

$1.8M_\odot$. However, when we look at a $2.0M_\odot$ strange quark star (figures 9 - 11) and a $2.2M_\odot$ strange quark star (figure 16), we see that they both evolve in a similar manner to the $2.1M_\odot$ strange quark star: they all have a collapsed core.

Therefore from these plots for a strange quark star with a strange quark mass of $150MeV/c^2$, bag constant of $B = 60MeV/fm^3$, and strong coupling constant of $\alpha_c = 0.45$, we find that there may be such a thing as a strange quark star and from these results we suspect that a maximum mass limit for a stable strange quark star could lie between these two masses, $1.8M_\odot < M_{\text{max}} < 2.0M_\odot$. However, we say this with a bit of caution because the strange quark star is still hypothetical and this method for finding stability may be questionable.

Closing Remarks

We have seen that our Fortran program has been a very well means for doing numerical work on stellar dynamics. When we tested the root program that is in Appendix C, we found that we matched May and White's results with excellent accuracy as we showed in Chapter III.

One may wonder how well our strange quark star model compares to the predecessors in table II. We found from the table that the average $\overline{M}_{\text{max}} = 1.98M_\odot$ and if we take the center point of our M_{max} range we have $M_{\text{max}} = 1.9M_\odot$. If we also average the radii from table V, we obtain $\overline{R}_{\text{max}} = 10.93km$; whereas we got $R_{\text{max}} = 11.22km$ for $M_{\text{max}} = 1.9M_\odot$ as seen in table V. These results are only as good as are our program and we should be able to get better results by having more precise equations of state for quark confinement and a better understanding of the coupling between quarks and between quarks and gluons. We should also pay more attention to the strong density shock waves that are possible throughout such stars, as discussed by May and White [13,6]. Despite these problems, we still believe and agree with Witten [12] that the most stable matter at these high densities is strange quark matter. We suspect that a star composed of such matter can not exist for an initial stellar mass greater than $2.1M_\odot$ and from our study we seem to find a maximum mass range for a strange quark lying some where between $1.8M_\odot$ and $2.0M_\odot$. We should also point out that these strange quark stars are possibly just the core of a slightly larger star like a neutron star. Since we have assumed in our initial setup of this problem that "*the star*" must satisfy charge neutrality, then there must be some form of a crust or an envelope of charge neutralizing matter around the strange quark star. It may also be possible to have a massive neutron star which gains mass by some means of accretion from a companion star to form, momentarily, a stable strange quark star before proceeding to a black hole. The bad news is that this strange quark mass range of $1.8M_{\odot}$ to $2.0M_{\odot}$ lies within the mass range of neutron stars, which extends from $1.4M_{\odot}$ to $3.0M_{\odot}$. So one may conclude that neutron stars and strange quark stars are essentially the same type of star. Maybe strange quark stars are just neutron star with a big percentage of strange quarks inside. If this is so, then we may need to redefine the term "strange quark star" to be a neutron star with a certain percentage of the star being composed of strange quarks. Thus we need a dual model, one that takes into account both the neutron and the quark matter equations of state, and allows the baryon-quark phase transition to take place when needed. Our model could be improved by looking at other $\varepsilon > \varepsilon_{\text{min}}$ so that the initial internal mass density is closer to $\rho_{_N}$, which may lessen the initial shock wave. The most important thing that should be considered is changing the initial conditions of the strange quark star to include general quark matter and the appropriate baryon-quark and quark-strange quark phase transitions.

BIBLIOGRAPHY

1. P. Haensel, *et al.*, "Strange Quark Stars", *Astron. Astrophys.* **160** (1986) 121.

2. C. Alcock, *et al.*, "Strange Stars", *Ap. J.* **310** (1986) 261.

3. E. Farhi and R.L. Jaffe, "Strange Matter", *Phys. Rev. D* **Vol.30** #11 (1984) 2379.

4. S. L. Shapiro and S. A. Teukolsky, Black Holes, White Dwarfs, and Neutron Stars, John Wiley and Sons, (1983).

5. J. R. Oppenheimer and G. M. Volkoff, "On Massive Neutron Cores", *Phys. Rev.* **55** (1939) 374.

6. M. M. May and R. H. White, "Stellar Dynamics and Gravitational Collapse", Methods in Comp. Phys. **Vol. 7** (1967) 219.

7. N. Itoh, "Hydrostatic Equilibrium of Hypothetical Quark Stars", Prog. Theor. Phys. **Vol. 44** #1 (1970) 291.

8. G. Chapline and M. Nauenberg, "Asymptotic Freedom and the Baryon-Quark Phase Transition", *Phys. Rev. D* **Vol.16** # 2 (1977) 450.

9. G. Chapline and M. Nauenberg, "Phase Transition From Baryon to Quark Matter", *Nature* **Vol. 264** #18 (1976) 235.

10. K. Brecher and G. Caporaso, "Obese 'Neutron' Stars", *Nature* **Vol. 259** (1976) 377.

11. T. Øveräard and E. Østaard , "Mass, Radius, and Moment of Inertia for Hypothetical Quark Stars", *Astron. Astrophys.* **243** (1991) 412.

12. E. Witten, "Cosmic Separation of Phases", *Phys. Rev. D* **141** #2 (1984) 272.

13. M. M. May and R. H. White, "Hydrodynamic Calculations of General-Relativistic Collapse", *Phys. Rev.* **141** #4 (1966) 1232.

14. G. C. McVittie, "An Example of Gravitational Collapse in General Relativity", *Ap. J.* **143** (1966) 682.

15. C. W. Misner and D. H. Sharp, "Relativistic Equations for Adiabatic, Spherically Symmetric Gravitational Collapse", *Phys. Rev.* **136** #2B (1964) B571.

16. S. Weinberg, Gravitation and Cosmology, John Wiley and Sons, (1972).

17. R. C. Tolman, Relativity, Thermodynamics, and Cosmology, Oxford, (1934).

18. L. D. Landau and E. M. Lifshitz, The Classical Theory of Fields, Pergamon, (1994).

19. A. Goyal and S. Dutta, "Flipped Neutrino Emissivity From Strange Matter", *Phys. Rev. D* **Vol.49** #8 (1994) 3910.

20. B. V. Martemyanov, "How Large Can The Crust of a Strange Star Be?", *Phys. Rev. D* **Vol.49** #8 (1994) 4293.

21. M. L. Olesen and J. Madsen, "Nucleation of Quark Matter Bubbles in Neutron Stars", *Phys. Rev. D* **Vol.49** #6 (1994) 2698.

22. Z. Dai, T. Lu, and Q. Peng, "The Birth of Strange Matter From Two-Flavor Quark Matter in a Neutron Star", Phys. Lett. B **319** (1993) 199.

23. J. Madsen, "Bulk Viscosity of Strange Quark Matter, Damping of Quark Star Vibration, and the Maximum Rotation Rate of Pulsars", *Phys. Rev. D* **Vol.46** #8 (1992) 3290.

24. S. Chakrabarty, "Equation of State of Strange Quark Matter and Strange Star", *Phys. Rev. D* **Vol.43** #2 (1991) 627.

25. W. B. Fechner and P. C. Joss, "Quark Star With 'Realistic' Equations of State", *Nature* **Vol. 274** (1978) 347.

26. B. Freedman and L. McLerran, "Quark Star Phenomenology", *Phys. Rev. D* **Vol.17** #4 (1978) 1109.

27. B. Freedman and L. McLerran, "... III. The Ground-State Energy of a Relativistic Quark Gas", *Phys. Rev. D* **Vol.16** #4 (1977) 1169.

28. K. Brecher, "Quark Bags or Interacting Neutrons: A "Neutron" Star Test", *Ap. J.* **215** (1977) L17.

29. J. R. Oppenheimer and H. Snyder, "On Continued Gravitational Contraction", *Phys. Rev.* **56** (1939) 455.

30. J. J. Carroll, A Theoretical Study of Gravitational Collapse for a Quark Star, Oklahoma State University Master Thesis, (Dec. 1984)

31. S. Chandrasekhar, An Introduction to the Study of Stellar Structure, Dover, (1967).

32. D. Flamm and F. Schoberl, Introduction to the Quark Model of Elementary Particles, **Vol.1**, Gordon and Breach Sci., (1982).

APPENDICES

APPENDIX A

THE EQUATIONS FOR QUARK MATTER IN A PROGRAMMABLE FORM AND THE CORRECTED MAY AND WHITE EQUATIONS

Program Setup.

We have used the following procedure to calculate the thermodynamic properties for our strange quark stars. But first, we set the following initial conditions for the whole star:

M_T	the mass of the star,
m_s	the strange quark mass,
α_c	the strong coupling constant,
B	the bag pressure constant, and
$\varepsilon_{\min}$	the initial energy per unit mass.

With these initial conditions, we can calculate an initial radius, $R_\circ$, for the star by setting

$$R_\circ = \frac{2M_T G}{c^2} \; .$$

This gives us an initial mass density

$$\rho_\circ = \frac{3M_T}{4\pi R_\circ^3} \; ,$$

which is comparable to nuclear density.

Now we need to know the total strange quark number density for the above volume (*ie.* $n_s = \rho_\circ m_s$), but first we need the masses of the other two quarks which are given by the following relations:

$$m_u = \frac{m_s}{35.0} ,$$

$$m_d = \frac{3.6 m_s}{70.0} ,$$

and the mass of an electron, which is given by $m_e = 0.511 MeV/c^2$. We must now establish the conditions for the star to be totally electrically neutral. This is done by using equations (68, 72, 79, 80) and (96) in the following way. We first make a guess for the initial chemical potential for the whole system to use in equation (80), we choose $\mu \equiv \mu_s = 500 MeV$. We choose this value because μ_s must always be larger than $m_s c^2$ in equations (68) and (72) to return real values. In order to establish the electrically neutral condition, we must iterate these equations to converge on the true chemical potential for the whole system. Therefore, we calculate $\Omega_s(\mu)$ and $n_s(\mu)$ with the guessed value for μ, where $\Omega_s(\mu)$ and $n_s(\mu)$ are given by equations (68) and (72) restated below:

$$\begin{aligned}
\Omega_s(\mu) &= -\frac{1}{4\pi^2(\hbar c)^3}\left[\mu\sqrt{\mu^2 - m_s^2c^4}\left(\mu^2 - \frac{5}{2}m_s^2c^4\right)\right. \\
&\quad + \frac{3}{2}m_s^4c^8 \ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_s c^2}\right) - \frac{2\alpha_c}{\pi}\left[3\left(\mu\sqrt{\mu^2 - m_s^2c^4}\right.\right. \\
&\quad \left. - m_s^2c^4 \ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_s c^2}\right)\right)^2 - 2\left(\mu^2 - m_s^2c^4\right) \\
&\quad - 3m_s^4c^8 \ln^2\left(\frac{mc^2}{\mu}\right) + 6\ln\left(\frac{\tilde{\rho}}{\mu}\right)\left(\mu m_s^2c^4\sqrt{\mu^2 - m_s^2c^4}\right. \\
&\quad \left.\left.\left. - m_s^4c^8 \ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{m_s c^2}\right)\right)\right]\right]
\end{aligned} \tag{112}$$

and

$$\begin{aligned}
n_s(\mu) &= \frac{\mu^2 - m_s^2c^4}{\pi^2(\hbar c)^3}\left[\sqrt{\mu^2 - m_s^2c^4} - \frac{2\alpha_c}{\pi}\right. \\
&\quad \left. \times \left[\mu - \frac{3m_s^2c^4}{\sqrt{\mu^2 - m_s^2c^4}} \ln\left(\frac{\mu + \sqrt{\mu^2 - m_s^2c^4}}{\tilde{\rho}}\right)\right]\right] .
\end{aligned} \tag{113}$$

Then we take these values and use them in the recursion relation for μ, eq. (90), where ρ and ε for are initially constant for the whole system. The recursion relation for μ is restated below

$$\mu_{new}\left(\mu^{old}\right) = \frac{\rho c^2\left(1+\frac{\varepsilon}{c^2}\right) - \Omega_s\left(\mu^{old}\right) - B}{n_s\left(\mu^{old}\right)} \ . \tag{114}$$

Thus μ_{new} is then plugged back in to equations (68) and (72) and the process repeats until μ_{new} and μ^{old} converge. Once this is done, the star is now charge neutral and we can now section the star into 200 zones of equal thickness, Δr, and of constant μ, $\rho_{\circ}$, ε, $\Omega_s(\mu)$ and $n_s(\mu)$* where we have to recalculate $\rho_{\circ}$ for each zone by solving equation (114) for ρ. This ensures that μ, $\rho_{\circ}$, and ε are correct because for large ε's the radius decreases since the recursion relation for μ is trying to make $n_s(\mu)\, m_s$ equal to the rest mass density ρ. Therefore, the new initial radius is

$$R_{\circ} = \left[\frac{3M_T}{4\pi\left(\rho c^2 + \rho\varepsilon\right)}\right]^{\frac{1}{3}} \ , \tag{115}$$

where ρ is the proper rest mass density which is initally homogeneous and isotropic, and ε is the proper specific internal energy per unit mass and is initally constant for each zone.

At the end of Chapter II, we had made the assumption that our star is just a strange quark star (*ie.* where $i = s$). Therefore f^i_{j-1} and ξ^i_{j-1} from equations (84) and (85) are zero and reduce to

$$\Omega^i_{j-1}\left(\mu^i_{j-1}\right) = \Omega^s_{j-1}\left(\mu_{j-1}\right) \tag{116}$$

and

$$n^i_{j-1}\left(\mu^i_{j-1}\right) = n^s_{j-1}\left(\mu_{j-1}\right) \tag{117}$$

where $j = 2, 4, 6, \cdots JJ$ are the shells outside the zones $j-1$. From this the energy density per unit mass, eq. (92), takes the form of

$$\chi^s_{j-1} = \Omega^s_{j-1}\left(\mu_{j-1}\right) + \mu_{j-1} n^s_{j-1}\left(\mu_{j-1}\right) + B \ , \tag{118}$$

*Recall that in Chapter III we constructed the Fortran program with 200 consecutive sets of shells and zones, each zones have a thickness of Δr. Therefore the quark star is subdivided into 200 subsystems for which we must keep track of all the thermodynamical properties.

for any time. The recursion relation for the next chemical potential, eq. (114), for each zone now becomes

$$\mu_{j-1}^{n+1}\left(\mu_{j-1}^{n}\right) = \frac{\rho_{j-1}^{n}c^2 + \rho_{j-1}^{n}\varepsilon_{j-1}^{n} - \Omega_{j-1}^{s,n}\left(\mu_{j-1}^{n}\right) - B}{n_{j-1}^{s,n}\left(\mu_{j-1}^{n}\right)} , \tag{119}$$

where n is the most current time and ρ and ε are constant for the iteration and μ_{j-1}^{n+1} is calculated zone by zone throughout the star. The pressure is a function of ρ_{j-1}, ε_{j-1} and μ_{j-1}^{s} for each subsystem at any time n

$$P_{j-1}\left(\rho_{j-1}, \varepsilon_{j-1}, \mu_{j-1}^{s}\right) = \mu_{j-1}^{s} n_{j-1}^{s} - \rho_{j-1}c^2 - \rho_{j-1}\varepsilon_{j-1} \tag{120}$$

where ρ_{j-1} is expressed by equation (119) for any time n and takes the following form

$$\rho_{j-1} = \frac{\Omega_{j-1}^{s}\left(\mu_{j-1}^{s}\right) + \mu_{j-1}^{s} n_{j-1}^{s} + B}{\varepsilon_{j-1} + c^2} . \tag{121}$$

We now can iterate through the May and White numerical integration scheme [6] outlined in Chapter II to find the next pressures, internal energy densities for each new volume. We now can recalculate the next chemical potential, μ_{j-1}^{s}, for each of the subsystems. Then we start the whole procedure all over for the next iteration of the May and White numerical integration scheme.

Quark Matter.

If we were to use a model with a homogeneous and isotropic distribution of up, down, and strange quarks along with electrons to form a quark star. Then we should use the explicit form of the setup as described in the section entitled "Equations Of State For Quark Matter" from Chapter II. We have listed here the equations in a programmable form as seen in equations (118-121), where the j indices define the shell or zone, the i indices indicate the constituent, $(i = u, d, s, e)$, and the n indices indicate the time or time step (See Chapter III for more detail about the indices j and n).

The energy density per unit mass, eq. (66), for any time n, will transform into

$$\chi_{j-1} \equiv \rho_{j-1}c^2 + \rho_{j-1}\varepsilon_{j-1} , \tag{122}$$

which is the same as

$$\chi_{j-1} = \sum_i \left(\Omega^i_{j-1}\left(\mu^i_{j-1}\right) + \mu^i_{j-1} n^i_{j-1}\left(\mu^i_{j-1}\right)\right) + B \,. \tag{123}$$

From equation (123), $\sum_i \mu^i_{j-1} n^i_{j-1}\left(\mu^i_{j-1}\right)$ for any time n can now be expressed as

$$\begin{aligned}\sum_i \mu^i_{j-1} n^i_{j-1}\left(\mu^i_{j-1}\right) &= 3\mu_{j-1} n^B_{j-1} + \mu^e_{j-1}\left[n^u_{j-1}\left(\mu_{j-1}\right) - n^e_{j-1}\left(\mu_{j-1}\right)\right. \\ &\quad \left. + \xi^u_{j-1}\left(\mu_{j-1}, \mu^u_{j-1}\right) - \xi^e_{j-1}\left(\mu_{j-1}, \mu^e_{j-1}\right)\right] \,,\end{aligned} \tag{124}$$

where n^B_{j-1} are the baryonic number densities for each zone. We want $P(\chi, \mu^i, \mu)$ not just $P(\mu)$, so equations (84) and (85) for any time n becomes

$$\Omega^i_{j-1}\left(\mu^i_{j-1}\right) = \Omega^i_{j-1}\left(\mu_{j-1}\right) + f^i_{j-1}\left(\mu_{j-1}, \mu^i_{j-1}\right) \tag{125}$$

and

$$n^i_{j-1}\left(\mu^i_{j-1}\right) = n^i_{j-1}\left(\mu_{j-1}\right) + \xi^i_{j-1}\left(\mu_{j-1}, \mu^i_{j-1}\right) \,. \tag{126}$$

The pressure equation, eq. (89), which is a function of χ, μ^i, and μ for each subsystem for any central time, n, can be expressed by

$$P^n_{j-1}\left(\chi^n_{j-1}, \mu^{i,n}_{j-1}, \mu^n_{j-1}\right) = \sum_i \mu^{i,n}_{j-1} n^{i,n}_{j-1}\left(\mu^{i,n}_{j-1}\right) + \sum_i f^{i,n}_{j-1}\left(\mu^n_{j-1}, \mu^{i,n}_{j-1}\right) - \chi^n_{j-1} \,, \tag{127}$$

where μ^n_{j-1} is the chemical potential for the next time step and is calculated by the recursion relation, eq. (90), which can be written in the following manner:

$$\begin{aligned}\mu^n_{j-1} &= \left\{\chi^n_{j-1} - \sum_i \Omega^{i,n}_{j-1}\left(\mu^{i,n}_{j-1}\right) - B - \mu^{e,n}_{j-1}\left[n^{u,n}_{j-1}\left(\mu^n_{j-1}\right)\right.\right. \\ &\quad \left.\left. - n^{e,n}_{j-1}\left(\mu^n_{j-1}\right) + \xi^{u,n}_{j-1}\left(\mu^n_{j-1}, \mu^{u,n}_{j-1}\right) - \xi^{e,n}_{j-1}\left(\mu^n_{j-1}, \mu^{e,n}_{j-1}\right)\right]\right\} / 3n^{B,n}_{j-1}.\end{aligned} \tag{128}$$

Once the $\left(\mu^n_{j-1}\right)'s$ have been calculated for each zone, $j-1$, the $\left(\mu^{i,n}_{j-1}\right)'s$ can be calculated from equation (80) which is re-expressed as

$$\begin{aligned}\mu^n_{j-1} &\equiv \mu^{s,n}_{j-1} = \mu^{d,n}_{j-1} \\ \mu^n_{j-1} &\equiv \mu^{u,n}_{j-1} + \mu^{e,n}_{j-1}\end{aligned} \,. \tag{129}$$

Now we can iterate through the May and White numerical integration scheme [6] outlined in Chapter II to find the next pressures and internal energy densities for each new volume. We now can recalculate for the next chemical potentials μ^s_{j-1} for each of the subsystems. Then we start the whole procedure all over for the next iteration of the May and White numerical integration scheme.

Corrections.

When working with May and White's numerical integration scheme, we found a few typographical errors. The corrected form of these equations are shown with our definition of the indices j and n.

May and White's equation 107 should read

$$\Gamma^0_{j-1} = +\sqrt{1 + \frac{1}{2}\left[\left(\frac{u^1_j}{c}\right)^2 + \left(\frac{u^1_{j-2}}{c}\right)^2\right] - \frac{2G}{c^2}\frac{1}{2}\left[\frac{m^0_j}{R^0_j} + \frac{m^0_{j-2}}{R^0_{j-2}}\right]}\ . \tag{130}$$

May and White's equation 133 should read

$$\begin{aligned}\left(\frac{u_\mu}{R_\mu}\right)^{n+1}_{j-1} \quad = \quad & \left\{4\pi c^2 \rho^n_{j-1}\left(R^2\right)^n_{j-1}\frac{\Gamma^n_j - \Gamma^n_{j-2}}{\Delta\mu_{j-1}} + 4\pi G\rho^n_{j-1}\left(1 + \frac{\varepsilon^n_{j-1}}{c^2}\right)\left[\sqrt{(R^2)^n_{j-1}}\right]\right. \\ & \left. -\frac{G}{2}\left[\frac{m^n_j}{\left(R^n_j\right)^2} + \frac{m^n_{j-2}}{\left(R^n_{j-2}\right)^2}\right]\right\}\frac{2}{\left(u^{n+1}_j + u^{n+1}_{j+2}\right)}\ .\end{aligned} \tag{131}$$

May and White's equation 143 should read

$$f \equiv \left[\frac{\varepsilon^{n+2}_{j+1} - \varepsilon^{n+2}_{j-1} + (P+Q)^{n+1}_j\left[\left(\rho^{n+2}_{j+1}\right)^{-1} - \left(\rho^{n+2}_{j-1}\right)^{-1}\right]}{w^{n+2}_j c^2}\right]\ . \tag{132}$$

May and White's equation 149 should read

$$\Delta m^{n+2}_{j-1} = \Gamma^{n+2}_{j-1}\left(1 + \frac{\varepsilon^{n+2}_{j-1}}{c^2}\right)\Delta\mu_{j-1}\ . \tag{133}$$

APPENDIX B

NON-FREE-FALL PROGRAM:
FOR OUR STRANGE QUARK STAR MODEL

```
C     THIS 'LOGIN.FOR' FILE DECLARES THE ARRAYS, VARIABLES,
C     COMMON BLOCKS, AND THE PARAMETERS USED THOUGHOUT THE
C     THE PROGRAM 'QS__NFF.FOR' AND IS CALLED UP IN EACH
C     SUBROUTINE.
*
      DOUBLE PRECISION R(5,-1:401), delR(5,-1:401),
     !RSQ(5,-1:401),rho(5,-1:401),epsil(5,-1:401),U(5,-1:401),
     !UR(5,-1:401),PQ(5,-1:401),M(5,-1:401),delM(5,-1:401),
     !delMU(-1:401),P(5,-1:401),Q(5,-1:401),W(5,-1:401),
     !gamma(5,-1:401),A(5,-1:401),delT(6)
      DOUBLE PRECISION Irho(5),IRad(5),X1(5,0:200),
     !Y1(5,0:200),X2(5,0:200),Y2(5,0:200),X3(5,0:200),
     !Y3(5,0:200),deltTr(0:200),deltTe(0:200),deltTc(0:200),
     !deltTMU(0:200)
      DOUBLE PRECISION ND(3),OMEGA(4,-1:401),
     !NUMDEN(4,-1:401),Vol(-1:401), Numb(3,-1:401),
     !massE(4), MU(5,-1:401,5), B, alphac, BARNUM
      DOUBLE PRECISION ETIME, ADJ, Mtot, radius, dr, SRAD,
     !PI, G, C, CSQ, fourPI, hb, HBC, Msun, toJ, CONST, rhoo
      INTEGER T, t1, t2, t3, t4, t5, t6
      INTEGER N, DD, NUM(0:200), JJ
      COMMON /MASSES/ delM, delMU
      COMMON /TIMES/ delT, ETIME, ADJ
      COMMON /MandR/ M, R
      COMMON /WandA/ W, A
      COMMON /PandQ/ P, Q
      COMMON /URPQ/ UR, PQ
      COMMON /VandN/ Vol, Numb
      COMMON /GAMMAU/ gamma, U
      COMMON /RRSQ/ delR, RSQ
      COMMON /THERMO/ rho, epsil
      COMMON /NUMBER/ N, JJ, DD, NUM
      COMMON /OTHERS/ Mtot, radius, dr, Irho, Irad, SRAD
      COMMON /TIMERS/ deltTr, deltTe, deltTc, deltTMU
      COMMON /PLOTDAT/ X1, Y1, X2, Y2, X3, Y3
      COMMON /SQPa/ OMEGA, NUMDEN, ND
      COMMON /SQPb/ alphac, MU, massE, B
      COMMON /BAYRON/ BARNUM
      PARAMETER(T = 3,t1=T-2,t2=T-1,t3=T,t4=T+1,t5=T+2,
     !t6=T+3)
      PARAMETER(PI=3.14159265358979323846D0, G=1.32733D11)
      PARAMETER(fourPI = 4.D0 * PI, C = 2.998D5, CSQ = C**2)
      PARAMETER(Msun = 1.99D30, toJ = 1.60219D-19)
      PARAMETER(hb = 6.582D-22, HBC = (hb * C)**3,
     !rhoo = 313.D0)
      PARAMETER(CONST = - 1.D0 / (fourPI * PI * HBC))
*
C      rhoo IS 1/3 OF THE NUCLEON REST ENERGY IN MeV
C      UNITS ARE;  {hb -> h_bar -> MeV * s}
C                  {Msun -> Solar Mass -> kg}
C                 {toJ -> Mev to Joule -> kg/km^3 * (km/s)^2}
C                  {C -> speed of light -> km/s}
*
```

```
*
C  Non-Free-Fall Fortran Program
C  Strange Quark Star equations of state:
C  P = MU(I) * NUMDEN(I) - Xi                    !Msun/km/s^2
C Xi = OMEGA(I) + MU(I) * NUMDEN(I) + B          !Msun/km/s^2
C  I = u, d, s, e ---> only using I = s here and
C Xi = rho * CSQ + epsil * rho
C
C Program QS__NFF.FOR VERSION: I-Working-Unadjusted
C Also uses program 'QS__LOG.FOR'
C============================================================
C Time Array                    |  Shell Array
C (1 to 6 centered on t3=3)     | (0 to 401 centered on j3=j)
C                               |
C t^n-1   Last current time     |j-1  Shell below central shell
C dt^n-.5 Step from t^n to      |j-.5 Fictitious zone, j-1 to j
C                      t^n-1
C t^n     Current Time          |j    Central Shell
C dt^n+.5 Step from t^n to      |j+.5 Fictitious zone, j to j+1
C                      t^n+1
C t^n+1   Next current time     |j+1 Shell above central shell
C
C      *** (time,shell)==(t3, j4)=(n, j+.5) and so on. ***
C============================================================
C                      Array Assignment
C         |  M. & W.  | Code            | M. & W.   | Code
C_________|___________|_________________|___________|__________
C     t1 = n - 1  -> t1 = t-2     | j1 = j - 1  -> j1 = j-2
C     t2 = n - .5 -> t2 = t-1     | j2 = j - .5 -> j2 = j-1
C     t3 = n      -> t3 = t       | j3 = j      -> j3 = j
C     t4 = n + .5 -> t4 = t+1     | j4 = j + .5 -> j4 = j+1
C     t5 = n + 1  -> t5 = t+2     | j5 = j + 1  -> j5 = j+2
C                                 |
C============================================================
C
C  MAIN UNITS ARE; {Mass in Msun, Radius in km,
C                                         Time in seconds}
C  WE DEFINE {a Joule to be a kg*km^2/s^2}
C            {a Newton to be a kg*km/s^2}
C
C There is no preferred caricature, all variables are
C defined as either Real or Integer, and the defaults aren't
C used in this progam. Also there is no difference in using
C upper or lower case caricatures. The only reason why both
C upper and lower case are used is that it make reading the
C code a bit easier.
C
C        PROGRAM'S NOMENCLATURE/ARRAY LIST WITH UNITS:
C
C R(5,-1:401).....R is the radial coordinate for each shell
C                   j3 in units of (km).
C delR(5,-1:401)..del-R is the change in radius from shell
C                   j1 to j3 in units of (km).
C RSQ(5,-1:401)...RSQ is the radial coordinate squared for
```

```
C                    each shell j3 in units of (km^2).
C M(5,-1:401).....M is the total mass enclosed by the Jth
C                    shell in units of (Msun).
C delM(5,-1:401)..delM is the mass enclosed between shells
C                    j1 and j3 or for the j2 zone in(Msun).
C delMU(-1:401)...del-MU is the total rest mass in the j2
C                    zone corresponding to a radius j3 in
C                    units of (Msun).
C rho(5,-1:401)...rho is the proper rest mass density for
C                    each zone at every time step in units
C                    of (Msun/km^3).
C epsil(5,-1:401).epsilon is the total internal energy per
C                    unit mass for each zone at each time in
C                    unit of (km/s)^2.
C P(5,-1:401).....P is the internal pressure of the star
C                    through zone j2 at a time t1, t3, or ¦
C                    of in units of (Msun/km^3*(km/s)^2).
C Q(5,-1:401).....Q is the artificial bulk viscosity term
C                    added to P. It has the same units as P.
C PQ(5,-1:401)....P+Q is pressure plus artificial viscosity
C                    for shell j3 at t3 in (Msun/s^2 km).
C W(5,-1:401).....W relativistic enthalpy or heat function,
C                    dimensionless.
C gamma(5,-1:401).gamma is a dimensionless quantity related
C                    to mass and the square root of the 2nd
C                    metric coefficient (b).
C U(5,-1:401).....U is the 1st component of the fluid
C                    4-velocity in units of (km/s).
C UR(5,-1:401)....dU/dR is the rate of change in the 1st
C                    component of the fluid 4-velocity (U)
C                    w.r.t. the radius (R) in (vel/km)=1/s.
C A(5,-1:401).....A is the square root of the first metric
C                    coefficient (a) and is dimensionless
C                    (a=Ac in units of km/s).
C delT(6).........del-T is the time array in units of sec
C deltTe(0:200)...del-T(epsilon) is the time step with a 2%
C                    change in epsilon in units of s.
C deltTr(0:200)...del-T(rho) is the time step with a 2%
C                    change in rho in units of s.
C CS..............c(s) is the speed of sound, velocity of
C                    sound in the fluid, in units of km/s.
C deltTc(0:200)...del-T(c) is the time step with a 20%
C                    change in the inverse of CS in units of
C                    seconds
C F...............F is the functional part of the recursion
C                    relations in A and P.
C X#(5,0:200).....these arrays store data to be sent to
C                    output files.
C Y#(5,0:200).....these arrays store data to be sent to
C                    output files.
C IRad(5).........store the initial radii at certain %mass
C                    levels.
C Irho(5).........store the initial densities at certain
C                    %mass levels.
```

```
C OMEGA(4,-1:401).are the thermodynamic potentials per
C                   volume in units of MeV/km^3.
C MU(5,-1:401,5)...are the chemical potentials in MeV.
C NUMDEN(4,-1:401).are the number densities in 1/km^3.
C radius...........the calculated initial radius in km.
C dr...............the initial sizes of each zone in km.
C rhoavg...........the mean density of all zones in units of
C                   density.
C epsavg...........the mean internal energy per unit mass in
C                   units of (km/s)^2.
C       COUNTERS:
C
C J counts the number of shells and zones by twos,
C                                         J = 0,2,4,...,JJ
C N counts the number of iterations, N = 0,1,2,...,N
C DD is an output counter, DD = 1,2..DD, then starts over.
C
C============================================================
C       CONSTANTS AND CONVERSIONS:
C
C    C = 2.998E5 km, speed of light, where CSQ = C*C
C   PI = 3.14159265358979323846D0, where fourPI = (4 * pi)
C   The universal gravitational constant,
C                        G = 1.32733D11 km/(Msun*s^2)
C Msun = 1.99D30 kg, the conversion from kg to solar masses
C  toJ = 1.60219D-19, the conversion from MeV to
C                                      kg*(km/s)^2 = Joule
C   hb = 6.582D-22 MeV*s
C  HBC = (hb * C)**3
C
C============================================================
C        BC's AND IC's
C
C JJ = 400 the total number of shells and zones.
C Mtot -> the intitial mass used in Msun.
C massE(3) -> the intitial rest mass energy of the strange
C              quark in units of MeV/c^2.
C MU(3) -> the intitial chemical potential of the strange
C              quark in units of MeV.
C alphac = 0.45, is the strong coupling constant in the
C              quark-gluon interaction to first order in
C              (g^2/4PI), dimensionless.
C   B = 57 Mev/fm^3, the bag pressure constant.
C  dt = 1.0D-7 s, the intitial time step.
C ADJ = ?? * dt s, adjustments to time for errors in the
C                  integrated time.
C
C============================================================
C
C        THE BODY OF THE PROGRAM
C
      INCLUDE 'QS__LOG.FOR'
      DOUBLE PRECISION FF
      OPEN (UNIT = 1, FILE= 'QS__DAT1.DAT', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE= 'QS__DAT2.DAT', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE= 'QS__DAT3.DAT', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE= 'QS__DAT4.DAT', STATUS = 'OLD')
      OPEN (UNIT = 5, FILE= 'QS__DAT5.DAT', STATUS = 'OLD')
      OPEN (UNIT = 6, FILE= 'QS__DAT6.DAT', STATUS = 'OLD')
      PRINT*, '*****---> I`M WORKING <---*****'
      N=0                          !SET ITERATION COUNTER.
      DD=1                         !SET OUTPUT COUNTER.
      ff=99.D0                     !SET ff GREATER THEN ZERO,
C                                   IF ff < 0 THEN PROGRAM
      CALL INITIAL
      CALL PLOTS(ff)
      N=N+1                        !INCREMENT OUTPUT COUNTER.
  119 CONTINUE
*(119)
      delT(t3) = (delT(t4) + delT(t2))/2.D0
      ETIME = ETIME + delT(t3)
      CALL DYNAMI
      CALL NEXT_MU
      CALL NSTEP
      CALL UPDATE
      CALL OUTONE(ff)
      IF (N .EQ. 2500) THEN
         PRINT*, '**** 1/4 th. DONE ****'
      ELSE IF (N .EQ. 5000) THEN
         PRINT*, '**** 1/2 DONE ****'
      ELSE IF (N .EQ. 7500) THEN
         PRINT*, '**** 3/4 th DONE ****'
      ENDIF
      IF (R(3,JJ) .LE. SRAD) THEN
         PRINT*, 'YES! WE HAVE A BLACK HOLE.'
         GOTO 33
      ENDIF
      IF (N .LE. 10000) THEN
         N = N + 1                 !INCREMENT ITERATION COUNTER.
         DD = DD + 1               !INCREMENT OUTPUT COUNTER.
         GOTO 119
      ENDIF
   33 CONTINUE
   22 CALL OUTTWO
      CALL OUTPUT
      STOP
      END
*
*============================================================
*
      SUBROUTINE INITIAL
      INCLUDE 'QS__LOG.FOR'
      DOUBLE PRECISION diff, percdiff, del, number, dt, tau,
     !rhoavg2,rhoavg, epsavg, newMU, oldMU, radtwo,
     !newOMEGA,ES,avgpress
      INTEGER J, j1, j2, j3, j4, j5
      INTEGER K, I
*
*    INITIALIZING ALL ARRAYS.
*
      JJ = 400            !TOTAL NUMBER OF SHELLS AND ZONES
      DO 10 I = 1,5
         DO 20 J = -1,JJ+1,1
            R(I,J) = 0.D0
            delR(I,J) = 0.D0
            rho(I,J) = 0.D0
            U(I,J) = 0.D0
            RSQ(I,J) = 0.D0
            M(I,J) = 0.D0
            delM(I,J) = 0.D0
              IF (I .EQ. 1) delMU(J) = 0.D0
            gamma(I,J) = 1.D0
            UR(I,J) = 0.D0
            PQ(I,J) = 0.D0
            epsil(I,J) = 0.D0
            P(I,J) = 0.D0
            Q(I,J) = 0.D0
            W(I,J) = 1.D0
            A(I,J) = 1.D0
              DO k=1,5
                MU(I,J,k) = 0.D0
              END DO
            IF (I .LE. 4) THEN
              NUMDEN(I,J) = 0.D0
              OMEGA(I,J) = 0.D0
            END IF
   20    CONTINUE
         delT(I) = 0.D0
         Irho(I) = 0.D0
         IRad(I) = 0.D0
         DO 25 K = 0,200
            X1(I,K) = 0.D0
            Y1(I,K) = 0.D0
            X2(I,K) = 0.D0
            Y2(I,K) = 0.D0
            X3(I,K) = 0.D0
            Y3(I,K) = 0.D0
            IF (I .EQ. 1) THEN
               NUM(K) = 0
               deltTe(K) = 0.D0
               deltTr(K) = 0.D0
               deltTc(K) = 0.D0
               deltTMU(K) = 0.D0
               ENDIF
   25    CONTINUE
   10 CONTINUE
          delT(6) = 0.D0
*
C        ALL ARRAY ELEMENTS THAT ARE NOT RESTATED
C          ARE AT THE ABOVE DEFAULT VALUES.
************************************************************
C        INITIAL AND BOUNDARY CONDITIONS.
C        The values set here are up to the programmer.
*
      Mtot = 2.0D0       !INITIAL MASS, in Msun
      massE(3) = 150.0D0 !STRANGE QUARK REST MASS IN MeV/CSQ
                         !THE REST OF THE QUARK MASSES ARE
C                          DETERMINED BY THERE MASS RATIOS.
     SRAD = 2.D0 * Mtot * G / CSQ
     RADIUS = SRAD * 2.D0!INITIAL radius uncalibrated in MU.
      alphac = 0.45D0         !THE STRONG COUPLING
      B = 60.D0*(1.D+18)**3 !THE BAG PRESSURE CONSTANT IN
C                           !MeV'S PER km^3
      P(3,JJ+1) = 0.D0        !Outside pressure
      rhoavg2 = 0.d0
C
      epsavg = .5D0*CSQ    !initial guess which will change.
      rhoavg = 3.D0 * Mtot / (4.D0 * PI * RADIUS**3)
                            !in Msum/km^3
      oldMU = 500.D0       !initial guess which will change.
************************************************************
*
C        CALCULATING THE OTHER QUARK MASS
*
      massE(1) = massE(3)/35.D0
      massE(2) = massE(3)*3.6D0/70.D0
      massE(4) = 0.511D0
*
************************************************************
C        MU VARIABLE FOR THE STRANGE QUARK STAR.
C
*
      k=0
      diff = 1.D0
      ES = massE(3)
      del = 0.d0
  90  CONTINUE
      oldMU = oldMU + del
*STRANGE
        newOMEGA = CONST * (oldMU * DSQRT(oldMU**2
     !- ES**2) * (oldMU**2 - 5.D0 * ES**2 / 2.D0) + 3.D0 /
     !2.D0 * ES**4 * DLOG((oldMU + DSQRT(oldMU**2 -
     !ES**2)) / ES) - 2.D0 * alphac / PI * (3.D0 * (oldMU *
     !DSQRT(oldMU**2 - ES**2) - ES**2 * DLOG((oldMU +
     !DSQRT(oldMU**2 - ES**2)) / ES))**2 - 2.D0 *
     !(oldMU**2 - ES**2)**2 - 3.D0 * ES**4 * (DLOG(ES /
     !oldMU))**2 + 6.D0 * DLOG(rhoo / oldMU) *
     !(oldMU * ES**2 * DSQRT(oldMU**2 - ES**2) - ES**4
     !* DLOG((oldMU + DSQRT(oldMU**2 - ES**2)) / ES))))
*
      NUMBER = (oldMU**2 - ES**2) / (PI**2 * HBC) *
     !(DSQRT(oldMU**2 - ES**2) - 2.D0 * alphac / PI *
     !(oldMU - 3.D0 * ES**2 / DSQRT(oldMU**2 - ES**2) *
     !DLOG((oldMU + DSQRT(oldMU**2 - ES**2)) / rhoo)))
*
      newMU = ((rhoavg * (CSQ+epsavg) * Msun / toJ) - B -
     !NEWOMEGA) / NUMBER
      diff = (newMU - oldMU)
      percdiff = DABS(diff / ((newMU + oldMU) / 2.D0))
      IF ((diff.EQ.0.D0).OR.(percdiff .LT. 1.D-12)) GOTO 99
      del = diff/4.d0
      k = k+1
      GOTO 90
  99  CONTINUE
*
************************************************************
C    SETTING THE INITIAL CHEMICAL POTENTIALS FOR UP, DOWN,
C                AND ELECTRON IN MeV'S
*
      DO 11 J = 2, JJ, 2  !UP AND ELECTRONS ARE COUPLED BY
         j2 = J-1            ! MU(5)=MU(1)+MU(4) AND MUST BE
         MU(t3,j2,3) = newMU      ! MU_calibrated.
         MU(t3,j2,2) = MU(t3,j2,3)
         MU(t3,j2,1) = MU(t3,j2,3)/2.D0
         MU(t3,j2,4) = MU(t3,j2,3)/2.D0
  11  CONTINUE
      CALL THERMO_POTENTIALS   !AT THIS POINT "THE STAR" IS
      CALL MU_calibrate        !TOTALLY ELECTRICALLY NEUTRAL
C                               AND THE MU'S, NUMDEN'S, AND
C                                   OMEGA'S ARE CALCULATED.
************************************************************
C        BOUNDARY CONDITIONS FOR THE "STRANGE" QUARK STAR
*
      DO 55 I = 1,4
         DO 56 J = 2, JJ, 2
            j2=J-1
            j4=j+1
            IF (I .EQ. 3) GOTO 56
            MU(t3,j2,I) = 0.D0
            NUMDEN(I,j2) = 0.D0
            OMEGA(I,j2) = 0.D0
  56     CONTINUE
  55  CONTINUE
************************************************************
      DO 15 J = 2, JJ, 2
         j2 = J - 1
*(102)
         epsil(t3, j2) = epsavg
         call density(rho(t3,j2), epsil(t3,j2), J)
         CALL PRESSURE(P(t3,j2),epsil(t3,j2), rho(t3,j2), J)
         avgpress = avgpress + P(t3,j2)
         rhoavg2 = rhoavg2 + rho(t3, j2)
  15  CONTINUE
      avgpress = 2.D0 * avgpress / DREAL(JJ)
      rhoavg2 = 2.D0 * rhoavg2 / DREAL(JJ)
      RADtwo = (3.D0 Mtot/(fourPI * rhoavg2 * (1.D0 + epsavg
     !/ CSQ)))**(1.D0 / 3.D0)
      dr = 2.D0 * RADtwo / DREAL(JJ)
      DO 16 J = 2, JJ, 2
         j1 = J - 2
         j2 = J - 1
         j3 = J
         delR(t3,j2) = dr
*(103)
         R(t3,j3) = R(t3,j1) + delR(t3,j2)
         Vol(j2) = fourPI*(R(t3,j1)**3 - R(t3,j3)**3) / 3.D0
         Numb(3,j2) = Vol(j2) * NUMDEN(3,j2)
*
C  THE NUMBER OF PARTICLES ARE CONSTANT IN TIME FOR
C  EACH ZONE.
*
   16 CONTINUE
*(102)
      dt = 1.D-7                    !SET INITIAL TIME STEP HERE.
      ADJ = 0.D0 * dt               !SET THE TIME STEP ADJUSTMENT.
      delT(1) = dt
      delT(2) = dt
      delT(4) = dt
      ETIME = 0.D0
************************************************************
C        NEED TO SET ASIDE SOME INITIAL CONDITIONS FOR THE
C                    NON-FREE FALL PLOTS.
C        The shell which contains ___% of total initial mass
*
      IRad(1) = radius       ! 100%
      IRad(2) = R(3,380)     ! 85.7%
      IRad(3) = R(3,300)     ! 42.2%
      IRad(4) = R(3,252)     ! 25.0%
      IRad(5) = R(3,4)       ! 1.0E-4%
      Irho(1) = rho(3,JJ-1)  ! zone corresponding to 100%
      Irho(2) = rho(3,379)   ! zone corresponding to 85.7%
      Irho(3) = rho(3,299)   ! zone corresponding to 42.2%
      Irho(4) = rho(3,251)   ! zone corresponding to 25.0%
      Irho(5) = rho(3,3)     ! zone corresponding to 1.0E-4%
************************************************************
```

```
C        HEADER FOR QS__DAT1.DAT
*
      tau = DSQRT(3.D0 * PI / (32.D0 * G * rho(T3, JJ-1)))
C   tau -> Is the total time it would take for this star to
C               collapse under the Free-Fall conditions.
*
      WRITE(1,*)
      WRITE(1,*) '           TIME IS ADJUSTED BY  ',ADJ
      WRITE(1,*)
      WRITE(1,*) '    dt = ',dt,'      tau = ',tau
      WRITE(1,*) '   RAD = ',RADIUS,'     SRAD = ',SRAD
      WRITE(1,*) '  Mtot = ',Mtot,'   SQmass = ',massE(3)
      WRITE(1,*) 'alphac = ',alphac,'        B = ',B
      WRITE(1,*) 'rhoavg = ',rhoavg,'   epsavg = ',epsavg
      WRITE(1,*) 'rhoav2 = ',rhoavg2,' avgpress = ',avgpress
      WRITE(1,*) '         ',number/numden(3,1),'     RADb =
',RADtwo
      WRITE(1,*)
************************************************************
C        NOW THE SETUP FOR T = 0
*
      DO 60 J = 2, JJ, 2
         j1=J-2
         j2=J-1
         j3=J
         j4=j+1
         j5=j+2
*(104)
         IF (J .EQ. 2) THEN
            RSQ(t3,j2) = R(t3,j3)**2 / 3.D0
         ELSE IF (j .EQ. JJ) THEN
            RSQ(t3,j4) = R(t3,j3)**2 / 3.D0
            RSQ(t3,j2) = (R(t3,j1)**2 + R(t3,j1) * R(t3,j3)
     !+ R(t3,j3)**2) / 3.D0
         ELSE
            RSQ(t3,j2) = (R(t3,j1)**2 + R(t3,j1) * R(t3,j3)
     !+ R(t3,j3)**2) / 3.D0
         ENDIF
*(105 & 106)
         delM(t3,j2) = fourPI * rho(t3,j2) * (1.D0 +
     !epsil(t3,j2) / CSQ) * RSQ(t3,j2) * delR(t3,j2)
         M(t3,j3) = M(t3,j1) + delM(t3,j2)
*(107)
         IF (J .EQ. 2) THEN
          gamma(t3,j2)=DSQRT(1.D0-G/CSQ*M(t3,j3) / R(t3,j3))
         ELSE IF (J .EQ. JJ) THEN
          gamma(t3,j4)=DSQRT(1.D0-G/CSQ*M(t3,j3) / R(t3,j3))
          gamma(t3,j2)=DSQRT(1.D0-G/CSQ*(M(t3,j3) / R(t3,j3)
     !+ M(t3,j1) / R(t3,j1)))
         ELSE
          gamma(t3,j2)=DSQRT(1.D0-G/CSQ*(M(t3,j3) / R(t3,j3)
     !+ M(t3,j1) / R(t3,j1)))
         ENDIF
*(108)
```

```
         delMU(j2) = fourPI * rho(t3,j2) * RSQ(t3,j2) *
     !delR(t3,j2) / gamma(t3,j2)
*(110 & 112)
         W(t3,j2)=(epsil(t3,j2) + P(t3,j2)/rho(t3,j2)) / CSQ
     !+ 1.D0
   60 CONTINUE
      DO 70 J = 2, JJ, 2
         j1=j-2
         j2=J-1
         j3=J
         j4=J+1
*(109)
         IF (J .EQ. 2) delMU(j1) = delMU(j2) / 2.D0
         delMU(j3) = (delMU(j2) + delMU(j4)) / 2.D0
*(113)
         IF (J .EQ. 2) W(t3,j1) = (W(t3,j2) * delMU(j2)) /
     !(2.D0 * delMU(j1))
         W(t3,j3) = (W(t3,j2) * delMU(j2) + W(t3,j4) *
     !delMU(j4)) / (2.D0 * delMU(j3))
*(114)
         P(t3,j3) = (P(t3,j4) * delMU(j2) + P(t3,j2) *
     !delMU(j4)) / (2.D0 * delMU(j3))
   70 CONTINUE
      A(t3,JJ) = 1.d0/W(t3,JJ-1)
      A(t3,JJ-1) = 1.d0/W(t3,JJ-1)
      DO 80 J = JJ-2, 2, -2
         j2 = J - 1
         j3 = J
         j4 = J + 1
         IF (J .EQ. 0) THEN
         F = (epsil(t3,j4)-epsil(t3,j2) + P(t3,j3) * (1.D0 /
     !rho(t3,j4))) / (W(t3,j3) * CSQ)
         ELSEIF (J .EQ. JJ)  THEN
         F = (epsil(t3,j4)-epsil(t3,j2) + P(t3,j3) * (-1.D0/
     !rho(t3,j2))) / (W(t3,j3) * CSQ)
         ELSE
         F = (epsil(t3,j4)-epsil(t3,j2) + Q(t3,j3) * (1.D0 /
     !rho(t3,j4) - 1.D0 / rho(t3,j2))) / (W(t3,j3) * CSQ)
         ENDIF
*(142 & 144)
         A(t3,j2) = A(t3,j4)*(W(t3,j4)/ W(t3,j2)) * DEXP(-F)
         A(t3,j3) = (A(t3,j2)*W(t3,j2)+A(t3,j4)* W(t3,j4)) /
     !(2.D0 * W(t3,j3))
   80 CONTINUE
      CALL OUTPUT
      RETURN
      END
*
*===========================================================
*
```

```
      SUBROUTINE DYNAMI
      INCLUDE 'QS__LOG.FOR'
      DOUBLE PRECISION F
      INTEGER J, j1, j2, j3, j4, j5
      CHARACTER ITERAT*1
      ITERAT = 'Y'
*
  100 DO 10 J = 2, JJ, 2
         j1=J-2
         j2=J-1
         j3=J
         j4=J+1
         j5=J+2
         IF (ITERAT .EQ. 'N') GOTO 127
*(120)
      PQ(t3,j3) = ((P(t3,j4) + Q(t2,j4)) * delMU(j2) +
     !(P(t3,j2)+Q(t2,j2)) * delMU(j4)) / (2.D0 * delMU(j3))
*(121)
      IF (J.EQ.2) W(t3,j1) = (W(t3,j2) * delMU(j2)) / (2.D0
     !delMU(j1))
      W(t3,j3)=(W(t3,j2) * delMU(j2) + W(t3,j4) * delMU(j4))
     !/ (2.D0 * delMU(j3))
*(122)
      IF (J .EQ. 2) THEN
      gamma(t3,j1)=(gamma(t3,j2)*delMU(j2))/(2.D0*
     !delMU(j1))
      else If (J .EQ. JJ) then
      gamma(t3,j3)=(gamma(t3,j2)*delMU(j2))/(2.D0 *
     !delMU(j3))
      ELSE
      gamma(t3,j3)=(gamma(t3,j2) * delMU(j2) + gamma(t3,j4)
     !* delMU(j4)) / (2.D0 * delMU(j3))
      END IF
*(123)
      U(t4,j3) = U(t2,j3) - delT(t3) * A(t3,j3) * (fourPI *
     !R(t3,j3)**2*gamma(t3,j3)/W(t3,j3)*(P(t3,j4) + Q(t2,j4)
     !- P(t3,j2) - Q(t2,j2)) / delMU(j3) + G * M(t3,j3) /
     !R(t3,j3)**2 + fourPI * G * PQ(t3,j3) * R(t3,j3) / CSQ)
*(124 & 125)
      IF (J .EQ. 2) THEN
       A(t4,j1)=A(t3,j1) + delT(t4) * (A(t3,j1) - A(t1,j1))
     !/ (2.D0 * delT(t2))
       A(t4,-1)=A(t3,-1) + delT(t4) * (A(t3,-1) - A(t1,-1))
     !/ (2.D0 * delT(t2))
      ELSE IF (J .EQ. JJ) THEN
       A(t4,JJ+1) = A(t3,JJ+1) + delT(t4) * (A(t3,JJ+1) -
     !A(t1,JJ+1)) / (2.D0 * delT(t2))
      END IF
      A(t4,j2)=A(t3,j2) + delT(t4) * (A(t3,j2) - A(t1,j2)) /
     !(2.D0 * delT(t2))
      A(t4,j3)=A(t3,j3) + delT(t4) * (A(t3,j3) - A(t1,j3)) /
     !(2.D0 * delT(t2))
*(126)
      epsil(t4,j2) = epsil(t3,j2) + delT(t4) * (epsil(t3,j2)
     !- epsil(t1,j2)) / (2.D0 * delT(t2))
  127 CONTINUE
*(127)
      delR(t5,j2) = delR(t3,j2) + delT(t4) * (A(t4,j3) *
     !U(t4,j3) - A(t4,j1) * U(t4,j1))
*(128, 129, & 130)
      R(t5,j3) = R(t5,j1) + delR(t5,j2)
      delR(t4,j2) = (delR(t5,j2) + delR(t3,j2)) / 2.D0
      R(t4,j3) = (R(t5,j3) + R(t3,j3)) / 2.D0
*(131)
      IF (J .EQ. 2) THEN
         RSQ(t5,j2) = R(t5,j3)**2 / 3.D0
      ELSE IF (J .EQ. JJ) THEN
         RSQ(t5,j4) = R(t5,j3)**2 / 3.D0
         RSQ(t5,j2) = (R(t5,j1)**2 + R(t5,j1) * R(t5,j3) +
     !R(t5,j3)**2) / 3.D0
      ELSE
         RSQ(t5,j2) = (R(t5,j1)**2 + R(t5,j1) * R(t5,j3) +
     !R(t5,j3)**2) / 3.D0
      ENDIF
*(133)
      IF ((DABS(gamma(t3,j2)) .LT. 1.D-3) .OR. (delR(t4,j2)
     !.EQ. 0.D0)) THEN
        IF (J .EQ. 2) THEN
       UR(t4,j2) = (fourPI * CSQ * rho(t3,j2) * RSQ(t3,j2) *
     !(gamma(t3,j3)-gamma(t3,j1)) / delMU(j2) + fourPI * G *
     !rho(t3,j2)*(1.D0+ epsil(t3,j2)/CSQ)* DSQRT(RSQ(t3,j2))
     !- G * M(t3,j3) / R(t3,j3)**2 / 2.D0) * 2.D0 / U(t4,j3)
     ELSE
       UR(t4,j2) = (fourPI * CSQ * rho(t3,j2) * RSQ(t3,j2) *
     !(gamma(t3,j3)-gamma(t3,j1)) / delMU(j2) + fourPI * G *
     !rho(t3,j2)*(1.D0+epsil(t3,j2)/CSQ) * DSQRT(RSQ(t3,j2))
     !- G * (M(t3,j3)/R(t3,j3)**2 + M(t3,j1) / R(t3,j1)**2)
     !/ 2.D0) * 2.D0 / (U(t4,j3) + U(t4,j1))
     END IF
*(132)
      ELSE IF ((DABS(gamma(t3,j2)) .GE. 1.D-3) .AND.
     !(delR(t4,j2).NE. 0.D0)) THEN
        UR(t4,j2) = (U(t4,j3) - U(t4,j1)) / delR(t4,j2)
      END IF
*(135)
      F = A(t4,j2) * UR(t4,j2) * delT(t4)
*(135 & 136)
      rho(t5,j2)=rho(t3,j2)*(RSQ(t3,j2)/RSQ(t5,j2))*
     !DEXP(-F)
      rho(t4,j2) = (rho(t5,j2) + rho(t3,j2)) / 2.D0
*(136 & 137)
      IF (rho(t5,j2) .LT. rho(t3,j2)) THEN
         Q(t4,j2) = 0.D0
      ELSE IF (rho(t5,j2) .GT. rho(t3,j2)) THEN
         Q(t4,j2) = 2.D0*rho(t4,j2)* (R(t4,j3)**2*U(t4,j3) -
     !R(t4,j1)**2*U(t4,j1))**2/(gamma(t3,j2)*RSQ(t3,j2)**2)
      END IF
*(138)
```

```
      CALL PRESSURE(P(t4,j2), epsil(t4,j2), rho(t4,j2), J)
*(139)
      epsil(t5,j2)=epsil(t3,j2)-(P(t4,j2)+Q(t4,j2))*(1.D0 /
     !rho(t5,j2) - 1.D0 / rho(t3,j2))
*(138)
      CALL PRESSURE(P(t5,j2), epsil(t5,j2), rho(t5,j2), J)
*(140)
      W(t5,j2)=1.D0 + (epsil(t5,j2) + (P(t5,j2) + Q(t4,j2))
     !/ rho(t5,j2)) / CSQ
   10 CONTINUE
      DO 25 J = 2, JJ, 2
         j1=j-2
         j2=J-1
         j3=J
         j4=J+1
*(120)
      PQ(t5,j3) = ((P(t5,j4)+Q(t4,j4))*delMU(j2)+(P(t5,j2) +
     !Q(t4,j2)) * delMU(j4)) / (2.D0 * delMU(j3))
*(121)
      IF (J .EQ. 2) W(t5,j1) = W(t5,j2) * delMU(j2) /
     !(2.D0 * delMU(j1))
      W(t5,j3) = (W(t5,j2) * delMU(j2)+W(t5,j4) * delMU(j4))
     !/ (2.D0 * delMU(j3))
   25 CONTINUE
      A(t5,jj) = 1.d0/W(t5,jj-1)
      A(t5,jj-1) = A(t5,jj)
      DO 30 J = JJ-2, 2, -2
         j2 = J - 1
         j3 = J
         j4 = J + 1
*(143)
         IF (J .EQ. 0) THEN
         F = (epsil(t5,j4)-epsil(t5,j2) + PQ(t5,j3) * (1.D0
     !/ rho(t5,j4))) / (W(t5,j3) * CSQ)
         ELSEIF (J .EQ. JJ)  THEN
         F = (epsil(t5,j4)-epsil(t5,j2) + PQ(t5,j3) * (-1.D0
     !/ rho(t5,j2))) / (W(t5,j3) * CSQ)
         ELSE
         F = (epsil(t5,j4)-epsil(t5,j2) + PQ(t5,j3) * (1.D0
     !/ rho(t5,j4) - 1.D0 / rho(t5,j2))) / (W(t5,j3) * CSQ)
         ENDIF
*(142 & 144)
         A(t5,j2) = A(t5,j4)*(W(t5,j4)/ W(t5,j2)) * DEXP(-F)
         A(t5,j3) = (A(t5,j2) * W(t5,j2) + A(t5,j4) *
     !W(t5,j4)) / (2.D0 * W(t5,j3))
   30 CONTINUE
      IF (ITERAT .EQ. 'Y') THEN
*(145, 146, & 147)
         DO 50 J = 2, JJ, 2
            j2 = J-1
            j3 = J
            epsil(t4,j2)=(epsil(t5,j2)+ epsil(t3,j2)) / 2.D0
            A(t4,j2) = (A(t5,j2) + A(t3,j2)) / 2.D0
            A(t4,j3) = (A(t5,j3) + A(t3,j3)) / 2.D0
            IF (J .EQ. 2) THEN
               A(t4,j1) = (A(t5,j1) + A(t3,j1)) / 2.D0
               A(t4,-1) = (A(t5,-1) + A(t3,-1)) / 2.D0
            ELSEIF (J .EQ. JJ) THEN
               A(t4,JJ+1) = (A(t5,JJ+1) + A(t3,JJ+1)) / 2.D0
            ENDIF
   50    CONTINUE
         ITERAT = 'N'            !AT THIS POINT WE RETURN TO
         GOTO 100                !  EQ. 127 FOR ONE MORE CYCLE.
      END IF
      DO 60 J = 2, JJ, 2
      j1=J-2
      j2=J-1
      j3=J
*(148)
         gamma(t5,j2) = fourPI * rho(t5,j2) * RSQ(t5,j2) *
     !delR(t5,j2) / delMU(j2)
*(149)
         delM(t5,j2)=gamma(t5,j2)*(1.D0+ epsil(t5,j2) / CSQ)
     !* delMU(j2)
*(150)
         M(t5,j3) = M(t5,j1) + delM(t5,j2)
   60 CONTINUE
      RETURN
      END
*
*==============================================================
*
      SUBROUTINE NSTEP
      DOUBLE PRECISION HOLD1, HOLD2, HOLD4, HOLD5, HOLD6
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2, K
*
      DO 10 J = 2, JJ, 2
         j2 = J-1
         k = DINT(DREAL(J)/2.D0)
         deltTe(K) = 0.D0
         deltTr(K) = 0.D0
*         deltTc(K) = 0.D0
         deltTMU(K) = 0.D0
*(152)
         deltTe(k) = 1.D-1 * epsil(t5,j2) * delT(t4) /
     !DABS(epsil(t5,j2) - epsil(t3,j2))
*(153)
         deltTr(k) = 2.D-2 * rho(t5,j2) * delT(t4) /
     !DABS(rho(t5,j2)- rho(t3,j2))
*
C WE HAVE TAKEN OUT EQUATION 154 AND REPLACED IT WITH THE
C CHANGE IN MU(t5,j2,3). WE DO THIS BECAUSE deltTc(k) FORCES
C THE NEXT TIME STEP TO BE TO SMALL.
C
*(154)
*       Cs = DSQRT(epsil(t5,j2)/ (1.D0 + epsil(t5,j2) / CSQ))
*       deltTc(k)=2.D-1*delMU(j2)/(rho(t5,j2) * RSQ(t5,j2) *
*     !A(t5,j2) * Cs)
*(MU)
        deltTMU(k) = 2.D-2 * MU(t5,j2,3) * delT(t4) /
     !DABS(MU(t5,j2,3) - MU(t3,j2,3))
   10 CONTINUE
*
C     delT(t6) IS ,INPUT, =  MAX{delT AS SPECIFIED BY INPUT}
C        sorts to find the minimum
*
      HOLD1 = deltTe(1)
      HOLD2 = deltTr(1)
*      HOLD3 = deltTc(1)
      HOLD4 = deltTMU(1)
      DO 20 k = 2, DREAL(JJ)/2.D0
         HOLD1 = DMIN1(HOLD1, deltTe(k))
         HOLD2 = DMIN1(HOLD2, deltTr(k))
*         HOLD3 = DMIN1(HOLD3, deltTc(k))
         HOLD4 = DMIN1(HOLD4, deltTMU(k))
   20 CONTINUE
         HOLD5 = delT(t1)
         HOLD6 = 1.2D0 * delT(t4)
*(151)
      delT(t6) = DMIN1(HOLD1, HOLD2, HOLD4, HOLD5, HOLD6)
      delT(t2) = delT(t4)
      delT(t4) = delT(t6)
      RETURN
      END
*
*==========================================================
*
      SUBROUTINE THERMO_POTENTIALS
      DOUBLE PRECISION ES
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2
*
C THIS SUBROUTINE CALCULATES THE THERMODYNAMIC POTENTIALS
C PER UNIT VOLUME AS A FUNCTION OF THE CHEMICAL POTENTIAL
C WHICH HAVE UNITS OF MeV'S. WE ALSO CALCULATE THE
C CORRESPONDING NUMBER DENSITIES FOR QUARK MATTER.
*
C THE UNITS FOR THERMODYNAMIC POTENTIALS PER UNIT VOLUME
C ARE {MeV/km^3}, AND THE UNITS FOR THE NUMBER DENSITIES
C ARE {1/km^3}.
*
      ES = massE(3)
      DO 12 J = 2, JJ, 2
        j2=J-1
*UP
        OMEGA(1,j2) = MU(t3,j2,1)**4*(1.D0-2.D0*alphac / PI)
     !* CONST
        NUMDEN(1,j2) = MU(t3,j2,1)**3 * ( - 4.D0 * CONST) *
     !(1.D0 - 2.D0 * alphac / PI)
*DOWN
        OMEGA(2,j2) = MU(t3,j2,2)**4*(1.D0-2.D0*alphac / PI)
     !* CONST
        NUMDEN(2,j2) = MU(t3,j2,2)**3 * ( - 4.D0 * CONST) *
     !(1.D0 - 2.D0 * alphac / PI)
*STRANGE
        OMEGA(3,j2)=CONST*(MU(t3,j2,3)* DSQRT(MU(t3,j2,3)**2
     !- ES**2)*(MU(t3,j2,3)**2 - 5.D0 * ES**2 / 2.D0)+3.D0 /
     !2.D0*ES**4 * DLOG((MU(t3,j2,3)+DSQRT(MU(t3,j2,3)**2 -
     !ES**2))/ES)-2.D0* alphac / PI * (3.D0 * (MU(t3,j2,3) *
     !DSQRT(MU(t3,j2,3)**2-ES**2)-ES**2*DLOG((MU(t3,j2,3) +
     !DSQRT(MU(t3,j2,3)**2 - ES**2)) / ES))**2 - 2.D0 *
     !(MU(t3,j2,3)**2-ES**2)**2 - 3.D0 * ES**4 * (DLOG(ES /
     !MU(t3,j2,3)))**2 + 6.D0 * DLOG(rhoo / MU(t3,j2,3)) *
     !(MU(t3,j2,3)*ES**2*DSQRT(MU(t3,j2,3)**2-ES**2)- ES**4
     !*DLOG((MU(t3,j2,3)+DSQRT(MU(t3,j2,3)**2-ES**2))/ES))))
*
        NUMDEN(3,j2) = (MU(t3,j2,3)**2-ES**2)/(PI**2*HBC) *
     !(DSQRT(MU(t3,j2,3)**2 - ES**2) - 2.D0 * alphac / PI *
     !(MU(t3,j2,3)-3.D0*ES**2/DSQRT(MU(t3,j2,3)**2-ES**2) *
     !DLOG((MU(t3,j2,3)+DSQRT(MU(t3,j2,3)**2-ES**2))/rhoo)))
*ELECTRON
        OMEGA(4,j2) = CONST * MU(t3,j2,4)**4 / 3.D0
        NUMDEN(4,j2) = MU(t3,j2,4)**3 / (3.D0 * PI**2 * HBC)
  12  CONTINUE
      RETURN
      END
*
*===========================================================
*
      SUBROUTINE MU_calibrate
      DOUBLE PRECISION diff, percdiff, del, newND1, newMU1,
     !tofm
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2
      PARAMETER (tofm=1.d18)
      k=0
  10  CONTINUE
*
C        CONDITION FOR CHARGE NEUTRALITY
*
      ND(1) = NUMDEN(4,1)+(NUMDEN(2,1) + NUMDEN(3,1)) / 3.D0
      ND(2) = 2.D0 * NUMDEN(1,1) / 3.D0
*
C        THE BARYON NUMBER DENSITY IS
*
      ND(3) = (NUMDEN(1,1)+NUMDEN(2,1) + NUMDEN(3,1)) / 3.D0
      diff = (ND(1) - ND(2))
      percdiff = DABS(diff / ((ND(1) + ND(2)) / 2.D0))
      IF ((diff.EQ.0.D0) .OR. (percdiff.LE.1.D-12)) GOTO 99
      newND1 = DABS((ND(2) + diff/2.d0) * 3.D0/2.D0)
      newMU1 = ((newND1 * PI**2 * HBC) / (1.D0 - 2.D0 *
     !alphac / PI))**(1.D0/3.D0)
      del = DABS(newMU1 - MU(t3,1,1))
*
C    This is only good for an initial setup where the Quark
```

```
C           Star is homogeneous.
*
      DO 13 J=2,JJ,2
         j2=J-1
         IF ((ND(2) .GT. ND(1))) THEN
            MU(t3,j2,4) = MU(t3,j2,4) + del
            MU(t3,j2,1) = MU(t3,j2,1) - del
         ELSE IF (ND(2) .LT. ND(1)) THEN
            MU(t3,j2,4) = MU(t3,j2,4) - del
            MU(t3,j2,1) = MU(t3,j2,1) + del
         ENDIF
  13  CONTINUE
      CALL THERMO_POTENTIALS
      k=k+1
      GOTO 10
  99  CONTINUE
*
C   PRINTS OUT THE MAIN HEADER FOR OUTPUT FILE QS__DAT1.DAT
*
       WRITE(1,*) 'It took ',k,' iterations to balance the
     !following relations'
       WRITE(1,*)
       WRITE(1,140)
       WRITE(1,145) massE(1), massE(2), massE(3), massE(4)
       WRITE(1,*)
       WRITE(1,100)
       WRITE(1,105) MU(t3,1,1), MU(t3,1,2), MU(t3,1,3),
     !MU(t3,1,4)
       WRITE(1,*)
       WRITE(1,110)
       WRITE(1,115) NUMDEN(1,1), NUMDEN(2,1),
     !NUMDEN(3,1), NUMDEN(4,1), NUMDEN(4,1)
       WRITE(1,*)
       WRITE(1,120)
       WRITE(1,125) OMEGA(1,1), OMEGA(2,1), OMEGA(3,1),
     !OMEGA(4,1)
       WRITE(1,*)
       WRITE(1,130)
  100 FORMAT('The balanced chemical potentials for charge
     !neutrality in units of MeV')
  105 FORMAT('MU_u =',1D24.16,1X, '    MU_d =', 1D24.16, 1X,
     !'    MU_s =', 1D24.16, 1X, '    MU_e =', 1D24.16)
  110 FORMAT('The balanced number densities for charge
     !neutrality in units of 1/fm^3')
  115 FORMAT('NUMDEN_u =',1D24.16,1X, 'NUMDEN_d =',1D24.16,
     !1X,'NUMDEN_s =', 1D24.16,1X, 'NUMDEN_e =',1D24.16, 1X,
     !'B_NUMDEN =', 1D24.16)
  120 FORMAT('The balanced thermodynamic potentials for
     !charge neutrality in units of MeV/km^3')
  125 FORMAT('OMEGA_u =',1D24.16,1X, ' OMEGA_d =', 1D24.16,0
     !1X,' OMEGA_s =', 1D24.16, 1X, ' OMEGA_e =', 1D24.16)
  140 FORMAT('The balanced masses for each quark in units of
     !MeV')
  145 FORMAT('  massE_u =', 1D24.16, 1X, '  massE_d =',
     !1D24.16, 1X,
     !'  massE_s =', 1D24.16, 1X, '  massE_e =', 1D24.16)
  130 FORMAT(120('='))
      RETURN
      END
*
*=============================================================
*

      SUBROUTINE density(rho, epsil, J)
       DOUBLE PRECISION rho, epsil, OMEGA(4,-1:401), Xi,
     !NUMDEN(4,-1:401), ND(3), massE(4), B, alphac,
     !MU(5,-1:401,5),toJ, Msun, C,CSQ
      INTEGER J, T, t3
      COMMON /SQPa/ OMEGA, NUMDEN, ND
      COMMON /SQPb/ alphac, MU, massE, B
      PARAMETER (T = 3, t3=T)
      PARAMETER (Msun = 1.99D30, toJ = 1.60219D-19)
      PARAMETER (C = 2.998D5, CSQ = C**2)
*
C        SPECIFIC INTERNAL ENERGY PER UNIT MASS
*
      Xi = OMEGA(3,J-1) + MU(t3,J-1,3) * NUMDEN(3,J-1)
*
C        The total initial energy density is
      Xi = Xi + B                        !in MeV/km^3
      Xi = Xi * toJ / Msun               !in Msun/km^3*(km/s)^2
C        The total initial energy per unit mass is
      rho = Xi / (epsil + CSQ)
      RETURN
      END
*
*=============================================================
*
      SUBROUTINE PRESSURE(P, epsil, rho, J)
      DOUBLE PRECISION P, epsil, rho, SUM, OMEGA(4,-1:401),
     !NUMDEN(4,-1:401), ND(3), massE(4), B, alphac,
     !MU(5,-1:401,5), Msun, toJ,c,csq
      INTEGER J, T, t3
      COMMON /SQPa/ OMEGA, NUMDEN, ND
      COMMON /SQPb/ alphac, MU, massE, B
      PARAMETER (T = 3, t3=T)
      PARAMETER (Msun = 1.99D30, toJ = 1.60219D-19)
      PARAMETER (C = 2.998D5, CSQ = C**2)
*
      SUM = MU(t3,J-1,3) * NUMDEN(3,J-1) * toJ / Msun
      P = -(rho * CSQ + rho * epsil - SUM)   !Msun/km/s^2
*
C   ONE CAN ALSO USE "P = -OMEGA(3,J-1)-B" AS GIVEN BELOW.
C     P = -(OMEGA(3,J-1)+B) * toJ/Msun       !Msun/km/s^2
      RETURN
      END
*
=============================================================
*
      SUBROUTINE NEXT_MU
      DOUBLE PRECISION oldMU,newMU,diff, percdiff, del, ES,
     !NUMBER, newOMEGA
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2
      INTEGER K
*
      ES = massE(3)
      DO 11 j = 2, JJ, 2
         j2=j-1
         diff = 1.D0
         del = 0.d0
         k=0
         oldMU = MU(t3,j2,3)
   90 CONTINUE
      oldMU = oldMU + del
*STRANGE
        newOMEGA = CONST * (oldMU * DSQRT(oldMU**2
     !- ES**2) * (oldMU**2 - 5.D0 * ES**2 / 2.D0) + 3.D0 /
     !2.D0 * ES**4 * DLOG((oldMU + DSQRT(oldMU**2 -
     !ES**2)) / ES) - 2.D0 * alphac / PI * (3.D0 * (oldMU *
     !DSQRT(oldMU**2 - ES**2) - ES**2 * DLOG((oldMU +
     !DSQRT(oldMU**2 - ES**2)) / ES))**2 - 2.D0 *
     !(oldMU**2 - ES**2)**2 - 3.D0 * ES**4 * (DLOG(ES /
     !oldMU))**2 + 6.D0 * DLOG(rhoo / oldMU) *
     !(oldMU * ES**2 * DSQRT(oldMU**2 - ES**2) - ES**4
     !* DLOG((oldMU + DSQRT(oldMU**2 - ES**2)) / ES))))
*
      NUMBER = (oldMU**2 - ES**2) / (PI**2 * HBC) *
     !(DSQRT(oldMU**2 - ES**2) - 2.D0 * alphac / PI *
     !(oldMU - 3.D0 * ES**2 / DSQRT(oldMU**2 - ES**2) *
     !DLOG((oldMU + DSQRT(oldMU**2 - ES**2)) / rhoo)))
*
      newMU = ((rho(t4,j2)*(CSQ+epsil(t4,j2)) * Msun / toJ)
     !- B - newOMEGA) / NUMBER
      diff = (newMU - oldMU)
      percdiff = DABS(diff / ((newMU + oldMU) / 2.D0))
      IF ((diff.EQ.0.D0).OR.(percdiff .LT. 1.D-12)) GOTO 99
      del = diff/4.d0
      k = k+1
      if (k.GE.500) stop
      GOTO 90
   99 CONTINUE
       MU(t5,j2,3) = newMU
       OMEGA(3,j2) = newOMEGA
       NUMDEN(3,j2) = number
   11 CONTINUE
      RETURN
      END
*
*=========================================================
*
```

```
      SUBROUTINE UPDATE
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2, j3, K
*
      DO 10 J = 2, JJ, 2
      j2=J-1
      j3=J
      delR(t1,j2) = delR(t3,j2)
      delR(t3,j2) = delR(t5,j2)
      delR(t5,j2) = 0.D0
      delR(t2,j2) = delR(t4,j2)
      delR(t4,j2) = 0.D0
      R(t1,j3) = R(t3,j3)
      R(t3,j3) = R(t5,j3)
      R(t5,j3) = 0.D0
      R(t2,j3) = R(t4,j3)
      R(t4,j3) = 0.D0
      rho(t1,j2) = rho(t3,j2)
      rho(t3,j2) = rho(t5,j2)
      rho(t5,j2) = 0.D0
      rho(t2,j2) = rho(t4,j2)
      rho(t4,j2) = 0.D0
      RSQ(t1,j2) = RSQ(t3,j2)
      RSQ(t3,j2) = RSQ(t5,j2)
      RSQ(t5,j2) = 0.D0
      epsil(t1,j2) = epsil(t3,j2)
      epsil(t3,j2) = epsil(t5,j2)
      epsil(t5,j2) = 0.D0
      epsil(t2,j2) = epsil(t4,j2)
      epsil(t4,j2) = 0.D0
      gamma(t1,j2) = gamma(t3,j2)
      gamma(t3,j2) = gamma(t5,j2)
      gamma(t5,j2) = 1.D0
      gamma(t1,j3) = gamma(t3,j3)
      gamma(t3,j3) = gamma(t5,j3)
      gamma(t5,j3) = 1.D0
      U(t2,j3) = U(t4,j3)
      U(t4,j3) = 0.D0
      delM(t1,j2) = delM(t3,j2)
      delM(t3,j2) = delM(t5,j2)
      delM(t5,j2) = 0.D0
      M(t1,j3) = M(t3,j3)
      M(t3,j3) = M(t5,j3)
      M(t5,j3) = 0.D0
      P(t1,j2) = P(t3,j2)
      P(t3,j2) = P(t5,j2)
      P(t5,j2) = 0.D0
      P(t1,j3) = P(t3,j3)
      P(t3,j3) = P(t5,j3)
      P(t5,j3) = 0.D0
      Q(t2,j2) = Q(t4,j2)
      Q(t4,j2) = 0.D0
      UR(t2,j2) = 0.D0
      W(t1,j2) = W(t3,j2)
      W(t3,j2) = W(t5,j2)
      W(t5,j2) = 1.D0
      W(t1,j3) = W(t3,j3)
      W(t3,j3) = W(t5,j3)
      W(t5,j3) = 1.D0
      A(t1,j2) = A(t3,j2)
      A(t3,j2) = A(t5,j2)
      A(t5,j2) = 1.D0
      A(t1,j3) = A(t3,j3)
      A(t3,j3) = A(t5,j3)
      A(t5,j3) = 1.D0
         DO k=1,4
            MU(t3,j2,k) = MU(t5,j2,k)
            MU(t5,j2,k) = 0.d0
         END DO
   10 CONTINUE
*
C        RESET THE B.C.'S AND I.C.'S FOR ALL TIMES
*
      MU(t3,0,3) = 0.d0
      MU(t3,JJ+1,3) = 0.d0
      delMU(JJ+1) = 0.D0
      delMU(-1) = 0.D0
      U(t2,0) = 0.D0
      U(t3,0) = 0.D0
      U(t4,0) = 0.D0
      M(t1,0) = 0.D0
      M(t3,0) = 0.D0
      M(t5,0) = 0.D0
      R(t1,0) = 0.D0
      R(t2,0) = 0.D0
      R(t3,0) = 0.D0
      R(t4,0) = 0.D0
      R(t5,0) = 0.D0
      P(t5,JJ+1) = 0.D0
      P(t4,JJ+1) = 0.D0
      RETURN
      END
*
*===========================================================
C    THIS OUTPUT SUBROUTINE GOES TO DATAFILE 'QS__DAT1.DAT'
*
      SUBROUTINE OUTPUT
      INCLUDE 'QS__LOG.FOR'
      INTEGER J, j2, j3, j4, I
*
*      if ((N .NE. 0) .AND. (N .LT. 15000)) GOTO 11
      IF (N .LE. 2) I = 2
      IF (N .GT. 2) I = 40
      WRITE(1,100)
      WRITE(1,200) N, ETIME + ADJ, delT(t4)
      DO 50 J = 0, JJ, I
        j2=J-1
        j3=J
      j4=J+1
      IF (N .EQ. 0) THEN
      pres = P(t3,j3)
      ELSE
      pres = PQ(t3,j3)
      ENDIF
      dens = rho(t3,j2) / 1.4D-4
                        !dens/nuclear dens(Dimensionless)
    WRITE (1, 300) J
    WRITE (1, 600) P(t3,j2),RSQ(t3,j2),delMU(j2),delMU(j3)
    WRITE (1, 601) Pres
    WRITE (1, 700) delM(t3,j2), delR(t3,j2), dens,
   !gamma(t3,j2)
    Write (1, 800) M(t3,j3),R(t3,j3),epsil(t3,j2),U(t2,j3)
    Write (1, 900) A(t3,j2),A(t3,j3),W(t3,j2),W(t3,j3)
    Write (1, 400) MU(t3,j2,1), MU(t3,j2,2), MU(t3,j2,3),
   !MU(t3,j2,4)
    Write (1, 500) NUMDEN(1,j2), NUMDEN(2,j2),
   !NUMDEN(3,j2),NUMDEN(4,j2)
    Write (1, 550) OMEGA(1,j2), OMEGA(2,j2), OMEGA(3,j2),
   !OMEGA(4,j2)
    IF (J .EQ. JJ) THEN
    Pres = P(t3,j4)               !Msun/km^3 (km/s)^2
    dens = rho(t3,j4) / 1.4D-4    !(Dimensionless)
    WRITE (1, *)
    WRITE (1, 610) Pres, RSQ(t3,j4), delMU(j4)
    WRITE (1, 710) delM(t3,j4), delR(t3,j4), dens,
   !gamma(t3,j4)
    Write (1, 810) epsil(t3,j4)
    Write (1, 910) A(t3,j4), W(t3,j4)
    Write (1, 400) MU(t3,j4,1), MU(t3,j4,1), MU(t3,j4,3),
   !MU(t3,j4,4)
    Write (1, 500) NUMDEN(1,j4), NUMDEN(2,j4),
   !NUMDEN(3,j4),NUMDEN(4,j4)
    Write (1, 550) OMEGA(1,j4), OMEGA(2,j4), OMEGA(3,j4),
   !OMEGA(4,j4)
    ENDIF
 50 CONTINUE
    avg=0.d0
100 FORMAT(120('='))
200 FORMAT('    N =', I10, 7X, 'T33 =', 1D24.16, 6X,
   !'dT4 =', 1D24.16)
300 FORMAT('    J =', I10)
601 FORMAT('  P33 =', 1D24.16)
600 FORMAT('  P32 =', 1D24.16, 3X, 'RSQ32 =', 1D24.16, 3X,
   !'  dtMU2 =', 1D24.16, 3X, 'dtMU3 =', 1D24.16)
610 FORMAT('  P34 =', 1D24.16, 3X, 'RSQ34 =', 1D24.16, 3X,
   !'  dtMU4 =', 1D24.16, 3X, '          ', 24X)
700 FORMAT('dtM32 =', 1D24.16, 3X, 'dtR32 =', 1D24.16, 3X,
   !'  rho32 =', 1D24.16, 3X, 'Gam32 =', 1D24.16)
710 FORMAT('dtM34 =', 1D24.16, 3X, 'dtR34 =', 1D24.16, 3X,
   !'  rho34 =', 1D24.16, 3X, 'Gam34 =', 1D24.16)
800 FORMAT('  M33 =', 1D24.16, 3X, '  R33 =', 1D24.16, 3X,
   !'epsil32 =', 1D24.16, 3X, '  U23 =', 1D24.16)
```

```
  810 FORMAT('        ', 24X, 3X, '        ', 24X, 3X,
     !'epsil34 =', 1D24.16, 3X, '        ', 24X)
  900 FORMAT('  A32 =', 1D24.16, 3X, '  A33 =', 1D24.16, 3X,
     !'    W32 =', 1D24.16, 3X, '  W33 =', 1D24.16)
  910 FORMAT('  A34 =', 1D24.16, 3X, '        ', 24X, 3X,
     !'    W34 =', 1D24.16, 3X, '        ', 24X)
  400 FORMAT('MU(u) =', 1D24.16, 3X, 'MU(d) =', 1D24.16, 3X,
     !'  MU(s) =', 1D24.16, 3X, 'MU(e) =', 1D24.16)
  500 FORMAT('#D(u) =', 1D24.16, 3X, '#D(d) =', 1D24.16, 3X,
     !'  #D(s) =', 1D24.16, 3X, '#D(e) =', 1D24.16)
  550 FORMAT('TP(u) =', 1D24.16, 3X, 'TP(d) =', 1D24.16, 3X,
     !'  TP(s) =', 1D24.16, 3X, 'TP(e) =', 1D24.16)
  11  RETURN
      END
*
*=============================================================
C   THIS OUTPUT SUBROUTINE GOES TO DATAFILES 'QS__DAT*.DAT'
*
      SUBROUTINE OUTTWO
      INCLUDE 'QS__LOG.FOR'
      INTEGER I
*
      DO 20 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "R(mu,t), PLOTTING DATA FOR 100% OF
     !INITIAL MASS"
        WRITE(2,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(2,*)
        WRITE(3,*) "R(mu,t), PLOTTING DATA FOR 85.7% OF
     !INITIAL MASS"
        WRITE(3,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(3,*)
        WRITE(4,*) "R(mu,t), PLOTTING DATA FOR 42.2% OF
     !INITIAL MASS"
        WRITE(4,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(4,*)
        WRITE(5,*) "R(mu,t), PLOTTING DATA FOR 25.0% OF
     !INITIAL MASS"
        WRITE(5,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(5,*)
        WRITE(6,*) "R(mu,t), PLOTTING DATA FOR 1.0E-4% OF
     !INITIAL MASS"
        WRITE(6,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(6,*)
       ENDIF
        WRITE(2,100) NUM(I), I, X1(1,I), I, Y1(1,I)
        WRITE(3,100) NUM(I), I, X1(2,I), I, Y1(2,I)
        WRITE(4,100) NUM(I), I, X1(3,I), I, Y1(3,I)
        WRITE(5,100) NUM(I), I, X1(4,I), I, Y1(4,I)
        WRITE(6,100) NUM(I), I, X1(5,I), I, Y1(5,I)
  100 FORMAT(I5, 2X, 'X1(',I3,') =',1D24.16, 2X, 'Y1(',I3,')
     !=',1D24.16)
   20 CONTINUE
      WRITE(2,*)
      WRITE(3,*)
      WRITE(4,*)
      WRITE(5,*)
      WRITE(6,*)
      DO 30 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(2,*)
        WRITE(3,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(3,*)
        WRITE(4,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(4,*)
        WRITE(5,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(5,*)
        WRITE(6,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(6,*)
       ENDIF
        WRITE(2,200) NUM(I), I, X2(1,I), I, Y2(1,I)
        WRITE(3,200) NUM(I), I, X2(2,I), I, Y2(2,I)
        WRITE(4,200) NUM(I), I, X2(3,I), I, Y2(3,I)
        WRITE(5,200) NUM(I), I, X2(4,I), I, Y2(4,I)
        WRITE(6,200) NUM(I), I, X2(5,I), I, Y2(5,I)
  200 FORMAT(I5, 2X, 'X2(',I3,') =',1D24.16, 2X, 'Y2(',I3,')
     !=',1D24.16)
   30 CONTINUE
      WRITE(2,*)
      WRITE(3,*)
      WRITE(4,*)
      WRITE(5,*)
      WRITE(6,*)
      DO 40 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "    pressure and rho data for plotting "
        WRITE(2,*)
        WRITE(3,*) "    pressure and rho data for plotting "
        WRITE(3,*)
        WRITE(4,*) "    pressure and rho data for plotting "
        WRITE(4,*)
        WRITE(5,*) "    pressure and rho data for plotting "
        WRITE(5,*)
        WRITE(6,*) "    pressure and rho data for plotting "
        WRITE(6,*)
       ENDIF
        WRITE(2,300) NUM(I), I, X3(1,I), I, Y3(1,I)
        WRITE(3,300) NUM(I), I, X3(2,I), I, Y3(2,I)
        WRITE(4,300) NUM(I), I, X3(3,I), I, Y3(3,I)
        WRITE(5,300) NUM(I), I, X3(4,I), I, Y3(4,I)
        WRITE(6,300) NUM(I), I, X3(5,I), I, Y3(5,I)
  300 FORMAT(I5, 2X, 'X3(',I3,') =',1D24.16, 2X, 'Y3(',I3,')
     !=',1D24.16)
   40 CONTINUE
      RETURN
      END
*
```

```
*=====================================================================
*
      SUBROUTINE PLOTS(ff)
      DOUBLE PRECISION ff, RR(5), MM(5), Hrho(5), PRES(5),
     !kmtocm, time
      INCLUDE 'QS__LOG.FOR'
      INTEGER I, J
      PARAMETER (kmtocm = 1.D5)
*
C Plotting data for 100%, 85.7%, 42.2%, 25.0%, and 0.0001%
C of the total mass to correlate with that of May and
C White's figure 2.
*
      Time = ETIME + ADJ
C  ADJ is to adjust the time due to integration errors of
C  order delT(t1).
      RR(1)=R(3,400)
      RR(2)=R(3,380)
      RR(3)=R(3,300)
      RR(4)=R(3,252)
      RR(5)=R(3,4)
      MM(1)=M(3,400)
      MM(2)=M(3,380)
      MM(3)=M(3,300)
      MM(4)=M(3,252)
      MM(5)=M(3,4)
      Hrho(1)=rho(3,399)
      Hrho(2)=rho(3,379)
      Hrho(3)=rho(3,299)
      Hrho(4)=rho(3,251)
      Hrho(5)=rho(3,3)
      PRES(1)=P(3,399)
      PRES(2)=P(3,379)
      PRES(3)=P(3,299)
      PRES(4)=P(3,251)
      PRES(5)=P(3,3)
      IF (N .EQ. 0) I = 0
         NUM(I) = N
         DO 10 J = 1, 5, 1
* THIS PLOTS R(mu,t) FOR 100%, 85.7%, 42.2%, 25%, and
* 0.0001% OF MASS
         Y1(J,I)=RR(J)/IRad(J)
         X1(J,I)=PI/2.D0-(TIME*DSQRT(8.D0*PI*G
     !*Irho(J)/3.D0))
* THIS PLOTS R(t) FOR 100%, 85.7%, 42.2%, 25%, and 0.0001%
* OF MASS
         Y2(J,I) = RR(J)
         X2(J,I) = TIME
* THIS PLOTS RHO & P FOR 100%, 85.7%, 42.2%, 25%, and
* 0.0001% OF MASS
         Y3(J,I) = Hrho(J) * Msun / (kmtocm**3) * 1.d3 !g/cc
         X3(J,I) = PRES(J) * Msun / kmtocm * 1.d3    !erg/cc
 10   CONTINUE
      ff=X1(1,I)!TO STOP THE PROGRAM WHEN X TURNS NEGATIVE.
      I=I+1
      RETURN
      END
*===========================================================
*  THIS OUTPUT SUBROUTINE GOES TO THE SCREEN.
*
      SUBROUTINE OUTONE(ff)
      DOUBLE PRECISION ff
      INCLUDE 'QS__LOG.FOR'
*
      IF (N .EQ. 1) rr = R(3,JJ)
*
C  One can adjust the printed iterations by changing DD and
C  N in the following IF-THEN statement.
*
      IF (((DD .EQ. 100) .AND. (N .LE. 2000)) .OR.
     !((DD .EQ. 50) .AND. ((N .GT. 2000) .AND. (N .LE. 3000)
     !.OR. ((DD .EQ. 10) .AND. (N .GT. 3000)))  THEN
*
      IF (ff .GE. -1.D-4) CALL PLOTS(ff)
      CALL OUTPUT
       PRINT*, 'N =',N,' R(3,JJ) =',R(3,JJ)
       IF (R(3,JJ) .LT. rr) THEN
       PRINT*,'THE STAR`S RADIUS HAS COLLAPSED BY A TOTAL OF

     !', rr-R(3,JJ),' km'
       ELSE IF (R(3,JJ) .GT. rr) THEN
     PRINT*,'THE STAR`S RADIUS HAS EXPANDED BY A TOTAL OF ',
     !R(3,JJ)-rr,' km'
       ELSE
       PRINT*,'NO CHANGE '
       ENDIF
      DD = 0      !RESET OUTPUT COUNTER.
      ENDIF
      RETURN
      END
*
*===========================================================
C IF THE READER DESIRES AN UP TO DATE COPY OF THIS PROGRAM,
C FEEL FREE TO CONTACT THE AUTHOR.
```

APPENDIX C

NON-FREE-FALL PROGRAM: FOR MAY AND WHITE'S EQUATIONS OF STATE

```
C     THIS 'LOGIN.FOR' FILE DECLARES THE ARRAYS, VARIABLES,
C     COMMON BLOCKS, AND THE PARAMETERS USED THOUGHOUT THE
C     THE PROGRAM 'MW__NFF.FOR' AND IS CALLED IN EACH
C     SUBROUTINE.
*
      DOUBLE PRECISION R(5,-1:401),delR(5,-1:401),
     !RSQ(5,-1:401),rho(5,-1:401),epsil(5,-1:401),
     !U(5,-1:401),UR(5,-1:401),PQ(5,-1:401),M(5,-1:401),
     !delM(5,-1:401),delMU(-1:401),P(5,-1:401),Q(5,-1:401),
     !W(5,-1:401),gamma(5,-1:401),A(5,-1:401),delT(6)
      DOUBLE PRECISION Irho(5), IRad(5), X1(5,0:200),
     !Y1(5,0:200),X2(5,0:200),Y2(5,0:200),X3(5,0:200),
     !Y3(5,0:200),deltTr(0:200),deltTe(0:200),deltTc(0:200)
      DOUBLE PRECISION ETIME, ADJ, Mtot, radius, dr, SRAD,
     !PI, G, C, CSQ, fourPI, hb, HBC, Msun, toJ
      INTEGER T, t1, t2, t3, t4, t5, t6
      INTEGER N, DD, NUM(0:200), JJ
      COMMON /MASSES/ delM, delMU
      COMMON /TIMES/ delT, ETIME, ADJ
      COMMON /MandR/ M, R
      COMMON /WandA/ W, A
      COMMON /PandQ/ P, Q
      COMMON /URPQ/ UR, PQ
      COMMON /GAMMAU/ gamma, U
      COMMON /RRSQ/ delR, RSQ
      COMMON /THERMO/ rho, epsil
      COMMON /NUMBER/ N, JJ, DD, NUM
      COMMON /OTHERS/ Mtot, radius, dr, Irho, Irad, SRAD
      COMMON /TIMERS/ deltTr, deltTe, deltTc
      COMMON /PLOTDAT/ X1, Y1, X2, Y2, X3, Y3
      PARAMETER (T = 3,t1=T-2,t2=T-1, t3=T, t4=T+1, t5=T+2,
     !t6=T+3)
      PARAMETER (PI = 3.14159265358979323846D0,
     !G = 1.32733D11)
      PARAMETER(fourPI = 4.D0 * PI, C = 2.998D5, CSQ = C**2)
      PARAMETER(Msun = 1.99D30, toJ = 1.60219D-19)
      PARAMETER(hb = 6.582D-22, HBC = (hb * C)**3)
*
C      rhoo IS 1/3 OF THE NUCLEON REST ENERGY IN MeV
C     UNITS ARE; {hb -> h_bar -> MeV * s}
C                {Msun -> Solar Mass -> kg}
C                {toJ -> Mev to Joule -> kg/km^3 * (km/s)^2}
C                {C -> speed of light -> km/s}
*
```

```
*
C         Non-Free-Fall Collapse model for a star of
C                  the equations of state:
C                  P = (gamma - 1) epsil rho
C                      where gamma = 5/3
C                      thus, P = 2/3 epsil rho
C
C Program MW__NFF.FOR VERSION: AA-Working-Unadjusted
C Also uses program 'MW__LOG.FOR'
C============================================================
C Time Array                     ¦  Shell Array
C (1 to 6 centered on t3=3)      ¦ (0 to 401 centered on j3=j)
C                                ¦
C t^n-1   Last current time      ¦j-1  Shell below central shell
C dt^n-.5 Step from t^n to       ¦j-.5 Fictitious zone, j-1 to j
C                         t^n-1
C t^n     Current Time           ¦j    Central Shell
C dt^n+.5 Step from t^n to       ¦j+.5 Fictitious zone, j to j+1
C                         t^n+1
C t^n+1   Next current time      ¦j+1 Shell above central shell
C
C     *** (time,shell)==(t3, j4)=(n, j+.5) and so on. ***
C============================================================
C                         Array Assignment
C         ¦  M. & W.  ¦  Code            ¦ M. & W.   ¦  Code
C_________¦___________¦__________________¦___________¦__________
C      t1 = n - 1  -> t1 = t-2     ¦ j1 = j - 1  -> j1 = j-2
C      t2 = n - .5 -> t2 = t-1     ¦ j2 = j - .5 -> j2 = j-1
C      t3 = n      -> t3 = t       ¦ j3 = j      -> j3 = j
C      t4 = n + .5 -> t4 = t+1     ¦ j4 = j + .5 -> j4 = j+1
C      t5 = n + 1  -> t5 = t+2     ¦ j5 = j + 1  -> j5 = j+2
C                                  ¦
C============================================================
C
C  MAIN UNITS ARE; {Mass in Msun, Radius in km,
C                                        Time in seconds}
C  WE DEFINE {a Joule to be a kg*km^2/s^2}
C            {a Newton to be a kg*km/s^2}
C
C There is no preferred caricature, all variables are
C defined as either as Real or Integer, the defaults aren't
C used in this program. Also there is no difference in using
C upper or lower case caricatures. The only reason why both
C upper and lower case are used is that it make reading the
C code a bit easier.
C
C        PROGRAM'S NOMENCLATURE/ARRAY LIST WITH UNITS:
C
C R(5,-1:401).....R is the radial coordinate for each shell
C                    j3 in units of (km).
C delR(5,-1:401)..del-R is the change in radius from shell
C                    j1 to j3 in units of (km).
C RSQ(5,-1:401)...RSQ is the radial coordinate squared for
C                    each shell j3 in units of (km^2).
```

```
C M(5,-1:401).....M is the total mass inclosed by the Jth
C                    shell in units of (Msun).
C delM(5,-1:401)..delM is the mass enclosed between shells
C                    j1 and j3 or for the j2 zone in(Msun).
C delMU(-1:401)...del-MU is the total rest mass in the j2
C                    zone corresponding to a radius j3 in
C                    units of (Msun).
C rho(5,-1:401)...rho is the proper rest mass density for
C                    each zone at every time step in units
C                    of (Msun/km^3).
C epsil(5,-1:401).epsilon is the total internal energy per
C                    unit mass for each zone at each time in
C                    unit of (km/s)^2.
C P(5,-1:401).....P is the internal pressure of the star
C                    through zone j2 at a time t1, t3, or
C                    t5, in units of (Msun/km^3*(km/s)^2).
C Q(5,-1:401).....Q is the artificial bulk viscosity term
C                    added to P in the same units as P.
C PQ(5,-1:401)....P+Q is pressure plus Artificial Viscosity
C                    for shell j3 at t3 in (Msun/s^2 km).
C W(5,-1:401).....W relativistic enthalpy or heat function,
C                    dimensionless.
C gamma(5,-1:401).gamma is a dimensionless quantity related
C                    to mass and the square root of the 2nd
C                    metric coefficient (b).
C U(5,-1:401).....U is the 1st component of the fluid
C                    4-velocity in units of (km/s).
C UR(5,-1:401)....dU/dR is the rate of change in the 1st
C                    component of the fluid 4-velocity (U)
C                    w.r.t. the radius (R) in (vel/km)=1/s.
C A(5,-1:401).....A is the square root of the first metric
C                    coefficient (a) and is dimensionless
C                    (a=Ac in units of km/s).
C delT(6).........del-T is the time array in units of sec
C deltTe(0:200)...del-T(epsilon), is the time step with a 2%
C                    change in epsilon, in units of s.
C deltTr(0:200)...del-T(rho), is the time step with a 2%
C                    change in rho, in units of s.
C CS..............c(s), is the speed of sound, velocity of
C                    sound in the fluid, in units of km/s.
C deltTc(0:200)...del-T(c) is the time step with a 20%
C                    change in the inverse of CS in units of
C                    seconds
C F...............is the functional part of the recurrence
C                    relations in A and P.
C X#(5,0:200).....these arrays store data to be sent to
C                    output files.
C Y#(5,0:200).....these arrays store data to be sent to
C                    output files.
C IRad(5).........store the initial radii at certain %mass
C                    levels.
C Irho(5).........store the initial densities at certain
C                    %mass levels.
C radius..........the calculated initial radius in km.
```

```
C dr..............the initial sizes of each zone in km.
C rhoavg..........the mean density of all zones in units of
C                    density.
C epsavg..........the mean internal energy per unit mass in
C                    units of (km/s)^2.
C        COUNTERS:
C
C J counts the number of shells and zones by twos,
C                                          J = 0,2,4,...,JJ
C N counts the number of iterations, N = 0,1,2,...,N
C DD is an output counter, DD = 1,2..DD, then starts over.
C
C===========================================================
C        CONSTANTS AND CONVERSIONS:
C
C    C = 2.998E5 km, speed of light, where CSQ = C*C
C   PI = 3.14159265358979323846D0, where fourPI = (4 * pi)
C   The universal gravitational constant,
C                       G = 1.32733D11 km/(Msun*s^2)
C Msun = 1.99D30 kg, the conversion from kg to solar masses
C  toJ = 1.60219D-19, the conversion from MeV to
C                                       kg*(km/s)^2 = Joule
C   hb = 6.582D-22 MeV*s
C  HBC = (hb * C)**3
C
C===========================================================
C         BC's AND IC's
C
C JJ = 400 the total number of shells and zones.
C Mtot -> the initial mass used in Msun.
C  dt = 1.0D-7 s, the initial time step.
C ADJ = ?? * dt s, adjustments to time for errors in the
C                   integrated time.
C
C===========================================================
C          THE BODY OF THE PROGRAM
C
      INCLUDE 'MW__LOG.FOR'
      DOUBLE PRECISION FF
      OPEN (UNIT = 1, FILE= 'MW__NFF1.DAT', STATUS = 'OLD')
      OPEN (UNIT = 2, FILE= 'MW__NFF2.DAT', STATUS = 'OLD')
      OPEN (UNIT = 3, FILE= 'MW__NFF3.DAT', STATUS = 'OLD')
      OPEN (UNIT = 4, FILE= 'MW__NFF4.DAT', STATUS = 'OLD')
      OPEN (UNIT = 5, FILE= 'MW__NFF5.DAT', STATUS = 'OLD')
      OPEN (UNIT = 6, FILE= 'MW__NFF6.DAT', STATUS = 'OLD')
      PRINT*, '*****---> I`M WORKING <---*****'
      N=0                        !SET ITERATION COUNTER.
      DD=1                       !SET OUTPUT COUNTER.
      ff=99.D0                   !SET ff GREATER THEN ZERO,
C                                 IF ff < 0 THEN PROGRAM STOPS.
      CALL INITIAL
      CALL PLOTS(ff)
      N=N+1                       !INCREMENT OUTPUT COUNTER.
  119 CONTINUE
*(119)
      delT(t3) = (delT(t4) + delT(t2))/2.D0
      ETIME = ETIME + delT(t3)
      CALL DYNAMI
      CALL NSTEP
      CALL UPDATE
      CALL OUTONE(ff)
      IF (N .EQ. 3375) THEN
         PRINT*, '**** 1/4 th. DONE ****'
      ELSE IF (N .EQ. 6750) THEN
         PRINT*, '**** 1/2 DONE ****'
      ELSE IF (N .EQ. 10125) THEN
         PRINT*, '**** 3/4 th DONE ****'
      ENDIF
      IF (R(3,JJ) .LE. SRAD) THEN
         PRINT*, 'YES! WE HAVE A BLACK HOLE.'
         GOTO 33
      ENDIF
      IF (N .LE. 13500) THEN
         N = N + 1                !INCREMENT ITERATION COUNTER.
         DD = DD + 1              !INCREMENT OUTPUT COUNTER.
         GOTO 119
      ENDIF
   33 CONTINUE
   22 CALL OUTTWO
      CALL OUTPUT
      PRINT*,' HAAAAAAAA!  N = ',N, 'ITERATIONS I`M DONE.'
      STOP
      END
*
*============================================================
      SUBROUTINE INITIAL
      INCLUDE 'MW__LOG.FOR'
      DOUBLE PRECISION  dt, tau, rhoavg, epsavg, RADTWO
      INTEGER J, j1, j2, j3, j4, j5
      INTEGER K, I
*
*    INITIALIZATION OF ALL ARRAYS.
*
      JJ = 400            !TOTAL NUMBER OF SHELLS AND ZONES
      DO 10 I = 1,5
         DO 20 J = -1,JJ+1,1
            R(I,J) = 0.D0
            delR(I,J) = 0.D0
            rho(I,J) = 0.D0
            U(I,J) = 0.D0
            RSQ(I,J) = 0.D0
            M(I,J) = 0.D0
            delM(I,J) = 0.D0
              IF (I .EQ. 1) delMU(J) = 0.D0
            gamma(I,J) = 1.D0
            UR(I,J) = 0.D0
            PQ(I,J) = 0.D0
            epsil(I,J) = 0.D0
            P(I,J) = 0.D0
            Q(I,J) = 0.D0
            W(I,J) = 1.D0
            A(I,J) = 1.D0
   20    CONTINUE
         delT(I) = 0.D0
         Irho(I) = 0.D0
         IRad(I) = 0.D0
         DO 25 K = 0,200
            X1(I,K) = 0.D0
            Y1(I,K) = 0.D0
            X2(I,K) = 0.D0
            Y2(I,K) = 0.D0
            X3(I,K) = 0.D0
            Y3(I,K) = 0.D0
            IF (I .EQ. 1) THEN
               NUM(K) = 0
               deltTe(K) = 0.D0
               deltTr(K) = 0.D0
               deltTc(K) = 0.D0
               ENDIF
   25    CONTINUE
   10 CONTINUE
          delT(6) = 0.D0
          rhoavg = 0.D0
          epsavg = 0.D0
*
C        ALL ARRAY ELEMENTS THAT ARE NOT RESTATED
C          ARE AT THE ABOVE DEFAULT VALUES.
************************************************************
```

```
C        INITIAL AND BOUNDARY CONDITIONS.
C        The values set here are up to the programmer.
*
      Mtot = 21.0D0                      !INITIAL MASS, in Msun
      SRAD = 2.D0 * Mtot * G / CSQ
      RADIUS = SRAD / 6.2D-3             !in km
      dr = 2.D0 * RADIUS / DREAL(JJ)
      P(3,JJ+1) = 0.D0
*
************************************************************
*
      DO 15 J = 2,JJ,2
         j1 = J - 2
         j2 = J - 1
         j3 = J
         J5 = J + 2
         delR(t3,j2) = dr
*(102)
         CALL ENERGY(epsil(t3,j2))
         CALL DENSITY(rho(t3,j2), Mtot, RADIUS)
*(103)
         R(t3,j3) = R(t3,j1) + delR(t3,j2)
         epsavg = epsavg + epsil(t3,j2)
         rhoavg = rhoavg + rho(t3,j2)
  15  CONTINUE
         rhoavg = rhoavg/(DREAL(JJ)/2.D0)
         epsavg = epsavg/(DREAL(JJ)/2.D0)
        RADTWO = (3.D0 * Mtot / (fourPI * rhoavg * (1.D0 +
     !epsavg / CSQ)))**(1.D0 / 3.D0)
*(102)
      dt = 5.D-5                !SET INITIAL TIME STEP HERE.
      ADJ = 0.D0 * dt
      delT(1) = dt
      delT(2) = dt
      delT(4) = dt
      ETIME = 0.D0
************************************************************
*
C        NEED TO SET ASIDE SOME INITIAL CONDITIONS FOR THE
C                    NON-FREE FALL PLOTS.
C        The shell which contains ___% of total initial mass
*
      IRad(1) = radius        !100%
      IRad(2) = R(3,380)      !85.7%
      IRad(3) = R(3,300)      !42.2%
      IRad(4) = R(3,252)      !25.0%
      IRad(5) = R(3,4)        !1.0E-4%
      Irho(1) = rho(3,JJ-1)   !zone corresponding to 100%
      Irho(2) = rho(3,379)    !zone corresponding to 85.7%
      Irho(3) = rho(3,299)    !zone corresponding to 42.2%
      Irho(4) = rho(3,251)    !zone corresponding to 25.0%
      Irho(5) = rho(3,3)      !zone corresponding to 1.0E-4%
************************************************************
*
```

```
C        HEADER FOR MW__DAT1.DAT
*
     tau = DSQRT(3.D0 * PI / (32.D0 * G * rho(T3, JJ-1)))
*
C    tau -> Is the total time it would take for this star to
C                collapse under the Free-Fall conditions.
*
      WRITE(1,*)
      WRITE(1,*) '          TIME IS ADJUSTED BY  ',ADJ
      WRITE(1,*)
      WRITE(1,*) '    dt = ',dt,'     tau = ',tau
      WRITE(1,*) '   RAD = ',RADIUS,'    SRAD = ',SRAD
      WRITE(1,*) '  Mtot = ',Mtot, '   RADb = ',RADTWO
      WRITE(1,*) 'rhoavg = ',rhoavg, ' epsavg = ',epsavg
      WRITE(1,*)
*************************************************************
*
C        NOW THE SETUP FOR T = 0
*
      DO 60 J = 2, JJ, 2
         j1=J-2
         j2=J-1
         j3=J
         j4=j+1
         j5=j+2
*(104)
         IF (j .EQ. 2) THEN
            RSQ(t3,j2) = R(t3,j3)**2 / 3.D0
         ELSE IF (j .EQ. JJ) THEN
            RSQ(t3,j4) = R(t3,j3)**2 / 3.D0
            RSQ(t3,j2) = (R(t3,j1)**2 + R(t3,j1) * R(t3,j3)
     !+ R(t3,j3)**2) / 3.D0
         ELSE
            RSQ(t3,j2) = (R(t3,j1)**2 + R(t3,j1) * R(t3,j3)
     !+ R(t3,j3)**2) / 3.D0
         ENDIF
*(105 & 106)
         delM(t3,j2) = fourPI * rho(t3,j2) * (1.D0 +
     !epsil(t3,j2) / CSQ) * RSQ(t3,j2) * delR(t3,j2)
         M(t3,j3) = M(t3,j1) + delM(t3,j2)
*(107)
         IF (J .EQ. 2) THEN
          gamma(t3,j2) = DSQRT(1.D0-G/CSQ*M(t3,j3)/R(t3,j3))
         ELSE IF (J .EQ. JJ) THEN
          gamma(t3,j4) = DSQRT(1.D0-G/CSQ*M(t3,j3)/R(t3,j3))
          gamma(t3,j2) = DSQRT(1.D0-G/CSQ*(M(t3,j3)/
     !R(t3,j3) + M(t3,j1) / R(t3,j1)))
         ELSE
          gamma(t3,j2) = DSQRT(1.D0 - G / CSQ * (M(t3,j3) /
     !R(t3,j3) + M(t3,j1) / R(t3,j1)))
         ENDIF
*(108)
         delMU(j2) = fourPI * rho(t3,j2) * RSQ(t3,j2) *
     !delR(t3,j2) / gamma(t3,j2)
```

```
*(110 & 112)
         CALL PRESSURE(P(t3,j2), epsil(t3,j2), rho(t3,j2))
         W(t3,j2) = (epsil(t3,j2) + P(t3,j2) / rho(t3,j2)) /
     !CSQ + 1.D0
   60 CONTINUE
      DO 70 J = 2, JJ, 2
         j1=j-2
         j2=J-1
         j3=J
         j4=J+1
*(109)
         IF (J .EQ. 2) delMU(j1) = delMU(j2) / 2.D0
         delMU(j3) = (delMU(j2) + delMU(j4)) / 2.D0
*(113)
         IF (J .EQ. 2) W(t3,j1) = (W(t3,j2) * delMU(j2)) /
     !(2.D0 * delMU(j1))
         W(t3,j3) = (W(t3,j2) * delMU(j2) + W(t3,j4) *
     !delMU(j4)) / (2.D0 * delMU(j3))
*(114)
         P(t3,j3) = (P(t3,j4) * delMU(j2) + P(t3,j2) *
     !delMU(j4)) / (2.D0 * delMU(j3))
   70 CONTINUE
      A(t3,JJ) = 1.d0/W(t3,JJ-1)
      A(t3,JJ-1) = 1.d0/W(t3,JJ-1)
      DO 80 J = JJ-2, 2, -2
         j2 = J - 1
         j3 = J
         j4 = J + 1
         IF (J .EQ. 0) THEN
            F = (epsil(t3,j4) - epsil(t3,j2) + P(t3,j3) *
     !(1.D0 / rho(t3,j4))) / (W(t3,j3) * CSQ)
         ELSEIF (J .EQ. JJ)  THEN
            F = (epsil(t3,j4) - epsil(t3,j2) + P(t3,j3) *
     !(-1.D0 / rho(t3,j2))) / (W(t3,j3) * CSQ)
         ELSE
            F = (epsil(t3,j4) - epsil(t3,j2) + Q(t3,j3) *
     !(1.D0 / rho(t3,j4) - 1.D0 / rho(t3,j2))) / (W(t3,j3)
     !* CSQ)
         ENDIF
*(142 & 144)
         A(t3,j2) = A(t3,j4) * (W(t3,j4) / W(t3,j2)) *
     !DEXP(-F)
         A(t3,j3) = (A(t3,j2) * W(t3,j2) + A(t3,j4) *
     !W(t3,j4)) / (2.D0 * W(t3,j3))
   80 CONTINUE
      CALL OUTPUT
      RETURN
      END
*
*===========================================================
*
      SUBROUTINE DYNAMI
      INCLUDE ’MW__LOG.FOR’
      DOUBLE PRECISION F
      INTEGER J, j1, j2, j3, j4, j5
      CHARACTER ITERAT*1
      ITERAT = ’Y’
*
  100 DO 10 J = 2, JJ, 2
      j1=J-2
      j2=J-1
      j3=J
      j4=J+1
      j5=J+2
         IF (ITERAT .EQ. ’N’) GOTO 127
*(120)
      PQ(t3,j3) = ((P(t3,j4) + Q(t2,j4)) * delMU(j2) +
     !(P(t3,j2)+ Q(t2,j2)) * delMU(j4)) / (2.D0 * delMU(j3))
*(121)
      IF (J.EQ.2) W(t3,j1) = (W(t3,j2) * delMU(j2)) / (2.D0
     !* delMU(j1))
      W(t3,j3)=(W(t3,j2) * delMU(j2) + W(t3,j4) * delMU(j4))
    !/ (2.D0 * delMU(j3))
*(122)
      IF (J .EQ. 2) THEN
      gamma(t3,j1) = (gamma(t3,j2) * delMU(j2))/(2.D0 *
    !delMU(j1))
      else If (J .EQ. JJ) then
      gamma(t3,j3) = (gamma(t3,j2) * delMU(j2))/(2.D0 *
    !delMU(j3))
      ELSE
      gamma(t3,j3)=(gamma(t3,j2) * delMU(j2) + gamma(t3,j4)
    !* delMU(j4)) / (2.D0 * delMU(j3))
      END IF
*(123)
      U(t4,j3) = U(t2,j3) - delT(t3) * A(t3,j3) * (fourPI *
     !R(t3,j3)**2 * gamma(t3,j3) / W(t3,j3) * (P(t3,j4) +
     !Q(t2,j4) - P(t3,j2) - Q(t2,j2)) / delMU(j3) + G *
     !M(t3,j3) / R(t3,j3)**2 + fourPI * G * PQ(t3,j3) *
     !R(t3,j3) / CSQ)
*(124 & 125)
      IF (J .EQ. 2) THEN
       A(t4,j1)=A(t3,j1) + delT(t4) * (A(t3,j1) - A(t1,j1))
     !/ (2.D0 * delT(t2))
       A(t4,-1)=A(t3,-1) + delT(t4) * (A(t3,-1) - A(t1,-1))
     !/ (2.D0 * delT(t2))
      ELSE IF (J .EQ. JJ) THEN
       A(t4,JJ+1) = A(t3,JJ+1) + delT(t4) * (A(t3,JJ+1) -
     !A(t1,JJ+1)) / (2.D0 * delT(t2))
      END IF
      A(t4,j2)=A(t3,j2) + delT(t4) * (A(t3,j2) - A(t1,j2)) /
     !(2.D0 * delT(t2))
      A(t4,j3)=A(t3,j3) + delT(t4) * (A(t3,j3) - A(t1,j3)) /
     !(2.D0 * delT(t2))
```

```
*(126)
      epsil(t4,j2) = epsil(t3,j2) + delT(t4) * (epsil(t3,j2)
     !- epsil(t1,j2)) / (2.D0 * delT(t2))
  127 CONTINUE
*(127)
      delR(t5,j2) = delR(t3,j2) + delT(t4) * (A(t4,j3) *
     !U(t4,j3) - A(t4,j1) * U(t4,j1))
*(128, 129, & 130)
      R(t5,j3) = R(t5,j1) + delR(t5,j2)
      delR(t4,j2) = (delR(t5,j2) + delR(t3,j2)) / 2.D0
      R(t4,j3) = (R(t5,j3) + R(t3,j3)) / 2.D0
*(131)
      IF (J .EQ. 2) THEN
         RSQ(t5,j2) = R(t5,j3)**2 / 3.D0
      ELSE IF (J .EQ. JJ) THEN
         RSQ(t5,j4) = R(t5,j3)**2 / 3.D0
         RSQ(t5,j2) = (R(t5,j1)**2 + R(t5,j1) * R(t5,j3) +
     !R(t5,j3)**2) / 3.D0
      ELSE
         RSQ(t5,j2) = (R(t5,j1)**2 + R(t5,j1) * R(t5,j3) +
     !R(t5,j3)**2) / 3.D0
      ENDIF
*(133)
      IF ((DABS(gamma(t3,j2)) .LT. 1.D-3) .OR. (delR(t4,j2)
     !.EQ. 0.D0)) THEN
        IF (J .EQ. 2) THEN
       UR(t4,j2) = (fourPI * CSQ * rho(t3,j2) * RSQ(t3,j2) *
     !(gamma(t3,j3)-gamma(t3,j1)) / delMU(j2) + fourPI * G *
     !rho(t3,j2)*(1.D0 + epsil(t3,j2)/CSQ)*DSQRT(RSQ(t3,j2))
     !- G * M(t3,j3) / R(t3,j3)**2 / 2.D0) * 2.D0 / U(t4,j3)
     ELSE
       UR(t4,j2) = (fourPI * CSQ * rho(t3,j2) * RSQ(t3,j2) *
     !(gamma(t3,j3)-gamma(t3,j1)) / delMU(j2) + fourPI * G *
     !rho(t3,j2)*(1.D0 + epsil(t3,j2)/CSQ)*DSQRT(RSQ(t3,j2))
     !- G*(M(t3,j3) / R(t3,j3)**2 + M(t3,j1) / R(t3,j1)**2)
     !/ 2.D0) * 2.D0 / (U(t4,j3) + U(t4,j1))
     END IF
*(132)
      ELSE IF ((DABS(gamma(t3,j2)) .GE. 1.D-3) .AND.
     !(delR(t4,j2) .NE. 0.D0)) THEN
        UR(t4,j2) = (U(t4,j3) - U(t4,j1)) / delR(t4,j2)
      END IF
*(135)
      F = A(t4,j2) * UR(t4,j2) * delT(t4)
*(135 & 136)
      rho(t5,j2) = rho(t3,j2) * (RSQ(t3,j2) / RSQ(t5,j2)) *
     !DEXP(-F)
      rho(t4,j2) = (rho(t5,j2) + rho(t3,j2)) / 2.D0
*(136 & 137)
      IF (rho(t5,j2) .LT. rho(t3,j2)) THEN
         Q(t4,j2) = 0.D0
      ELSE IF (rho(t5,j2) .GT. rho(t3,j2)) THEN
         Q(t4,j2) = 2.D0 * rho(t4,j2) * (R(t4,j3)**2 *
     !U(t4,j3) - R(t4,j1)**2 * U(t4,j1))**2 / (gamma(t3,j2)
     !* RSQ(t3,j2)**2)
      END IF
*(138)
      CALL PRESSURE(P(t4,j2), epsil(t4,j2), rho(t4,j2))
*(139)
      epsil(t5,j2) = epsil(t3,j2) - (P(t4,j2) + Q(t4,j2))
     !* (1.D0 / rho(t5,j2) - 1.D0 / rho(t3,j2))
*(138)
      CALL PRESSURE(P(t5,j2), epsil(t5,j2), rho(t5,j2))
*(140)
      W(t5,j2)=1.D0 + (epsil(t5,j2) + (P(t5,j2) + Q(t4,j2))
     !/ rho(t5,j2)) / CSQ
   10 CONTINUE
      DO 25 J = 2, JJ, 2
      j1=j-2
      j2=J-1
      j3=J
      j4=J+1
*(120)
      PQ(t5,j3) = ((P(t5,j4) + Q(t4,j4)) * delMU(j2) +
     !(P(t5,j2)+Q(t4,j2)) * delMU(j4)) / (2.D0 * delMU(j3))
*(121)
      IF (J .EQ. 2) W(t5,j1) = W(t5,j2) * delMU(j2) /
     !(2.D0 * delMU(j1))
      W(t5,j3)=(W(t5,j2) * delMU(j2) + W(t5,j4) * delMU(j4))
     !/ (2.D0 * delMU(j3))
   25 CONTINUE
      A(t5,jj) = 1.d0/W(t5,jj-1)
      A(t5,jj-1) = 1.d0/W(t5,jj-1)
      DO 30 J = JJ-2, 2, -2
         j2 = J - 1
         j3 = J
         j4 = J + 1
*(143)
         IF (J .EQ. 0) THEN
            F = (epsil(t5,j4) - epsil(t5,j2) + PQ(t5,j3) *
     !(1.D0 / rho(t5,j4))) / (W(t5,j3) * CSQ)
         ELSEIF (J .EQ. JJ)  THEN
            F = (epsil(t5,j4) - epsil(t5,j2) + PQ(t5,j3) *
     !(-1.D0 / rho(t5,j2))) / (W(t5,j3) * CSQ)
         ELSE
            F = (epsil(t5,j4) - epsil(t5,j2) + PQ(t5,j3) *
     !(1.D0/rho(t5,j4) - 1.D0 / rho(t5,j2)))/(W(t5,j3)* CSQ)
         ENDIF
*(142 & 144)
         A(t5,j2) = A(t5,j4) * (W(t5,j4) / W(t5,j2)) *
     !DEXP(-F)
         A(t5,j3) = (A(t5,j2) * W(t5,j2) + A(t5,j4) *
     !W(t5,j4)) / (2.D0 * W(t5,j3))
   30 CONTINUE
      IF (ITERAT .EQ. 'Y') THEN
*(145, 146, & 147)
         DO 50 J = 2, JJ, 2
            j2 = J-1
            j3 = J
            epsil(t4,j2)=(epsil(t5,j2) + epsil(t3,j2))/2.D0
            A(t4,j2) = (A(t5,j2) + A(t3,j2)) / 2.D0
            A(t4,j3) = (A(t5,j3) + A(t3,j3)) / 2.D0
            IF (J .EQ. 2) THEN
               A(t4,j1) = (A(t5,j1) + A(t3,j1)) / 2.D0
               A(t4,-1) = (A(t5,-1) + A(t3,-1)) / 2.D0
            ELSEIF (J .EQ. JJ) THEN
               A(t4,JJ+1) = (A(t5,JJ+1) + A(t3,JJ+1)) / 2.D0
            ENDIF
   50    CONTINUE
         ITERAT = 'N'           !AT THIS POINT WE RETURN TO
         GOTO 100               !  EQ. 127 FOR ONE MORE CYCLE.
      END IF
      DO 60 J = 2, JJ, 2
         j1=J-2
         j2=J-1
         j3=J
*(148)
         gamma(t5,j2) = fourPI * rho(t5,j2) * RSQ(t5,j2) *
     !delR(t5,j2) / delMU(j2)
*(149)
         delM(t5,j2) = gamma(t5,j2) * (1.D0 + epsil(t5,j2)
     !/ CSQ) * delMU(j2)
*(150)
         M(t5,j3) = M(t5,j1) + delM(t5,j2)
   60 CONTINUE
      RETURN
      END
*
*============================================================
*
      SUBROUTINE NSTEP
      DOUBLE PRECISION HOLD1, HOLD2, HOLD3, HOLD4, HOLD5, Cs
      INCLUDE 'MW__LOG.FOR'
      INTEGER J, j2, K
*
      DO 10 J = 2, JJ, 2
         j2 = J-1
         k = DINT(DREAL(J)/2.D0)
         deltTe(K) = 0.D0
         deltTr(K) = 0.D0
         deltTc(K) = 0.D0
*(152)
         deltTe(k) = 2.D-2 * epsil(t5,j2) * delT(t4) /
     !DABS(epsil(t5,j2) - epsil(t3,j2))
*(153)
         deltTr(k) = 2.D-2 * rho(t5,j2) * delT(t4) /
     !DABS(rho(t5,j2) - rho(t3,j2))
*(154)
         Cs = DSQRT(epsil(t5,j2)/(1.D0 + epsil(t5,j2)/CSQ))
         deltTc(k) = 2.D-1 * delMU(j2) / (rho(t5,j2) *
     !RSQ(t5,j2) * A(t5,j2) * Cs)
   10 CONTINUE
```

```
*
C    delT(t6) IS ,INPUT, =  MAX{delT AS SPECIFIED BY INPUT}
C          sorts to find the minimum
*
      HOLD1 = deltTe(1)
      HOLD2 = deltTr(1)
      HOLD3 = deltTc(1)
      DO 20 k = 2, DREAL(JJ)/2.D0
         HOLD1 = DMIN1(HOLD1, deltTe(k))
         HOLD2 = DMIN1(HOLD2, deltTr(k))
         HOLD3 = DMIN1(HOLD3, deltTc(k))
   20 CONTINUE
         HOLD4 = delT(t1)
         HOLD5 = 1.2D0 * delT(t4)
*(151)
      delT(t6) = DMIN1(HOLD1, HOLD2, HOLD3, HOLD4, HOLD5)
      delT(t2) = delT(t4)
      delT(t4) = delT(t6)
      RETURN
      END
*
*============================================================
*
      SUBROUTINE DENSITY(rho, M, R)
       DOUBLE PRECISION rho, PI, M, R
       PARAMETER (PI = 3.14159265358979323846D0)
*
       rho = 3.D0 * M / (4.D0 * PI * R**3)    !in Msum/km^3
*
      RETURN
      END
*
*============================================================
*
      SUBROUTINE ENERGY(epsil)
       DOUBLE PRECISION epsil, C, CSQ
       PARAMETER (PI = 3.14159265358979323846D0,
     !C = 2.998D5, CSQ=C**2)
*
       epsil = (1.92D-5) * CSQ               !in km^2/s^2
!               SPECIFIC INTERNAL ENERGY PER UNIT MASS
      RETURN
      END
*
*============================================================
*
      SUBROUTINE PRESSURE(P, epsil, rho)
      DOUBLE PRECISION rho, epsil, P
*
         P = 2.D0*rho*epsil/3.D0      !in Msun/km/s^2
*                                  or in Msun/km^3*(km/s)^2
      RETURN
      END
*
```

```
*=======================================================
*
      SUBROUTINE UPDATE
      INCLUDE 'MW__LOG.FOR'
      INTEGER J, j2, j3
      DO 10 J = 2, JJ, 2
      j2=J-1
      j3=J
      delR(t1,j2) = delR(t3,j2)
      delR(t3,j2) = delR(t5,j2)
      delR(t5,j2) = 0.D0
      delR(t2,j2) = delR(t4,j2)
      delR(t4,j2) = 0.D0
      R(t1,j3) = R(t3,j3)
      R(t3,j3) = R(t5,j3)
      R(t5,j3) = 0.D0
      R(t2,j3) = R(t4,j3)
      R(t4,j3) = 0.D0
      rho(t1,j2) = rho(t3,j2)
      rho(t3,j2) = rho(t5,j2)
      rho(t5,j2) = 0.D0
      rho(t2,j2) = rho(t4,j2)
      rho(t4,j2) = 0.D0
      RSQ(t1,j2) = RSQ(t3,j2)
      RSQ(t3,j2) = RSQ(t5,j2)
      RSQ(t5,j2) = 0.D0
      epsil(t1,j2) = epsil(t3,j2)
      epsil(t3,j2) = epsil(t5,j2)
      epsil(t5,j2) = 0.D0
      epsil(t2,j2) = epsil(t4,j2)
      epsil(t4,j2) = 0.D0
      gamma(t1,j2) = gamma(t3,j2)
      gamma(t3,j2) = gamma(t5,j2)
      gamma(t5,j2) = 1.D0
      gamma(t1,j3) = gamma(t3,j3)
      gamma(t3,j3) = gamma(t5,j3)
      gamma(t5,j3) = 1.D0
      U(t2,j3) = U(t4,j3)
      U(t4,j3) = 0.D0
      delM(t1,j2) = delM(t3,j2)
      delM(t3,j2) = delM(t5,j2)
      delM(t5,j2) = 0.D0
      M(t1,j3) = M(t3,j3)
      M(t3,j3) = M(t5,j3)
      M(t5,j3) = 0.D0
      P(t1,j2) = P(t3,j2)
      P(t3,j2) = P(t5,j2)
      P(t5,j2) = 0.D0
      P(t1,j3) = P(t3,j3)
      P(t3,j3) = P(t5,j3)
      P(t5,j3) = 0.D0
      Q(t2,j2) = Q(t4,j2)
      Q(t4,j2) = 0.D0
      UR(t2,j2) = 0.D0
      PQ(t3,j3) = 0.D0
      PQ(t5,j3) = 0.D0
      W(t1,j2) = W(t3,j2)
      W(t3,j2) = W(t5,j2)
      W(t5,j2) = 1.D0
      W(t1,j3) = W(t3,j3)
      W(t3,j3) = W(t5,j3)
      W(t5,j3) = 1.D0
      A(t1,j2) = A(t3,j2)
      A(t3,j2) = A(t5,j2)
      A(t5,j2) = 1.D0
      A(t1,j3) = A(t3,j3)
      A(t3,j3) = A(t5,j3)
      A(t5,j3) = 1.D0
   10 CONTINUE
C        RESET THE B.C.'S AND I.C.'S FOR ALL TIMES
      delMU(JJ+1) = 0.D0
      delMU(-1) = 0.D0
      U(t2,0) = 0.D0
      U(t3,0) = 0.D0
      U(t4,0) = 0.D0
      M(t1,0) = 0.D0
      M(t3,0) = 0.D0
      M(t5,0) = 0.D0
      R(t1,0) = 0.D0
      R(t2,0) = 0.D0
      R(t3,0) = 0.D0
      R(t4,0) = 0.D0
      R(t5,0) = 0.D0
      P(t5,JJ+1) = 0.D0
      P(t4,JJ+1) = 0.D0
      RETURN
      END
*
*==============================================================
C     THIS OUTPUT SUBROUTINE GOES TO DATAFILE 'MW__DAT1.DAT'
*
      SUBROUTINE OUTPUT
      INCLUDE 'MW__LOG.FOR'
      INTEGER J, j2, j3, j4, I
*
      IF (N .LE. 2) I = 2
      IF (N .GT. 2) I = 40
      WRITE(1,100)
      WRITE(1,200) N, ETIME + ADJ, delT(t4)
      DO 50 J = 0, JJ, I
        j2=J-1
        j3=J
        j4=J+1
        IF (N .EQ. 0) THEN
        pres = P(t3,j3)
        ELSE
        pres = PQ(t3,j3)
        ENDIF
        dens = rho(t3,j2)    !/ 1.4D-4 dens/nuclear
      WRITE (1, 300) J
      WRITE (1, 600) P(t3,j2), RSQ(t3,j2), delMU(j2),
     !delMU(j3)
      WRITE (1, 601) Pres
      WRITE (1, 700) delM(t3,j2), delR(t3,j2), dens,
     !gamma(t3,j2)
      Write (1, 800) M(t3,j3), R(t3,j3), epsil(t3,j2),
     !U(t2,j3)
      Write (1, 900) A(t3,j2), A(t3,j3), W(t3,j2), W(t3,j3)
      IF (J .EQ. JJ) THEN
        IF (N .EQ. 0) THEN
        pres = P(t3,j3)        !Msun/km^3 (km/s)^2
        ELSE
        pres = PQ(t3,j3)
        ENDIF
        dens = rho(t3,j4)      !/ 1.4D-4     Msun/km^3
        WRITE (1, *)
        WRITE (1, 610) P(t3,j4), RSQ(t3,j4), delMU(j4)
        WRITE (1, 601) Pres
        WRITE (1, 710) delM(t3,j4), delR(t3,j4), dens,
     !gamma(t3,j4)
        Write (1, 810) epsil(t3,j4)
        Write (1, 910) A(t3,j4), W(t3,j4)
      ENDIF
   50 CONTINUE
  100 FORMAT(120('='))
  200 FORMAT('    N =', I10, 7X, 'T33 =', 1D24.16, 6X,
     !'dT4 =', 1D24.16)
  300 FORMAT('    J =', I10)
  601 FORMAT('  P33 =', 1D24.16)
  600 FORMAT('  P32 =', 1D24.16, 3X, 'RSQ32 =', 1D24.16, 3X,
     !'  dtMU2 =', 1D24.16, 3X, 'dtMU3 =', 1D24.16)
  610 FORMAT('  P34 =', 1D24.16, 3X, 'RSQ34 =', 1D24.16, 3X,
     !'  dtMU4 =', 1D24.16, 3X, '       ', 24X)
  700 FORMAT('dtM32 =', 1D24.16, 3X, 'dtR32 =', 1D24.16, 3X,
     !'  rho32 =', 1D24.16, 3X, 'Gam32 =', 1D24.16)
  710 FORMAT('dtM34 =', 1D24.16, 3X, 'dtR34 =', 1D24.16, 3X,
     !'  rho34 =', 1D24.16, 3X, 'Gam34 =', 1D24.16)
  800 FORMAT('  M33 =', 1D24.16, 3X, '  R33 =', 1D24.16, 3X,
     !'epsil32 =', 1D24.16, 3X, '  U23 =', 1D24.16)
  810 FORMAT('       ', 24X, 3X, '       ', 24X, 3X,
     !'epsil34 =', 1D24.16, 3X, '       ', 24X)
  900 FORMAT('  A32 =', 1D24.16, 3X, '  A33 =', 1D24.16, 3X,
     !'    W32 =', 1D24.16, 3X, '  W33 =', 1D24.16)
  910 FORMAT('  A34 =', 1D24.16, 3X, '       ', 24X, 3X,
     !'    W34 =', 1D24.16, 3X, '       ', 24X)
  11  RETURN
      END
*
*============================================================
C    THIS OUTPUT SUBROUTINE GOES TO DATAFILES 'MW__DAT*.DAT'
*
      SUBROUTINE OUTTWO
      INCLUDE 'MW__LOG.FOR'
      INTEGER I
*
      DO 20 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "R(mu,t), PLOTTING DATA FOR 100% OF
     !INITIAL MASS"
        WRITE(2,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(2,*)
        WRITE(3,*) "R(mu,t), PLOTTING DATA FOR 85.7% OF
     !INITIAL MASS"
        WRITE(3,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(3,*)
        WRITE(4,*) "R(mu,t), PLOTTING DATA FOR 42.2% OF
     !INITIAL MASS"
        WRITE(4,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(4,*)
        WRITE(5,*) "R(mu,t), PLOTTING DATA FOR 25.0% OF
     !INITIAL MASS"
        WRITE(5,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(5,*)
        WRITE(6,*) "R(mu,t), PLOTTING DATA FOR 1.0E-4% OF
     !INITIAL MASS"
        WRITE(6,*) "WITH THE TIME ADJUSTED BY ",ADJ," dt"
        WRITE(6,*)
       ENDIF
        WRITE(2,100) NUM(I), I, X1(1,I), I, Y1(1,I)
        WRITE(3,100) NUM(I), I, X1(2,I), I, Y1(2,I)
        WRITE(4,100) NUM(I), I, X1(3,I), I, Y1(3,I)
        WRITE(5,100) NUM(I), I, X1(4,I), I, Y1(4,I)
        WRITE(6,100) NUM(I), I, X1(5,I), I, Y1(5,I)
  100 FORMAT(I5, 2X, 'X1(',I3,') =',1D24.16, 2X, 'Y1(',I3,')
     !=',1D24.16)
   20 CONTINUE
      WRITE(2,*)
      WRITE(3,*)
      WRITE(4,*)
      WRITE(5,*)
      WRITE(6,*)
      DO 30 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(2,*)
        WRITE(3,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(3,*)
        WRITE(4,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(4,*)
        WRITE(5,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(5,*)
        WRITE(6,*) "   DATA FOR R(t) WITH THE TIME ADJUSTED"
        WRITE(6,*)
       ENDIF
        WRITE(2,200) NUM(I), I, X2(1,I), I, Y2(1,I)
        WRITE(3,200) NUM(I), I, X2(2,I), I, Y2(2,I)
        WRITE(4,200) NUM(I), I, X2(3,I), I, Y2(3,I)
        WRITE(5,200) NUM(I), I, X2(4,I), I, Y2(4,I)
        WRITE(6,200) NUM(I), I, X2(5,I), I, Y2(5,I)
  200 FORMAT(I5, 2X, 'X2(',I3,') =',1D24.16, 2X, 'Y2(',I3,')
     !=',1D24.16)
   30 CONTINUE
      WRITE(2,*)
      WRITE(3,*)
      WRITE(4,*)
      WRITE(5,*)
      WRITE(6,*)
      DO 40 I = 0, 200, 1
       IF (I .EQ. 0) THEN
        WRITE(2,*) "   pressure and rho data for plotting "
        WRITE(2,*)
        WRITE(3,*) "   pressure and rho data for plotting "
        WRITE(3,*)
        WRITE(4,*) "   pressure and rho data for plotting "
        WRITE(4,*)
        WRITE(5,*) "   pressure and rho data for plotting "
        WRITE(5,*)
        WRITE(6,*) "   pressure and rho data for plotting "
        WRITE(6,*)
       ENDIF
        WRITE(2,300) NUM(I), I, X3(1,I), I, Y3(1,I)
        WRITE(3,300) NUM(I), I, X3(2,I), I, Y3(2,I)
        WRITE(4,300) NUM(I), I, X3(3,I), I, Y3(3,I)
        WRITE(5,300) NUM(I), I, X3(4,I), I, Y3(4,I)
        WRITE(6,300) NUM(I), I, X3(5,I), I, Y3(5,I)
  300 FORMAT(I5, 2X, 'X3(',I3,') =',1D24.16, 2X, 'Y3(',I3,')
     !=',1D24.16)
   40 CONTINUE
      RETURN
      END
*
*=============================================================
*
      SUBROUTINE PLOTS(ff)
      DOUBLE PRECISION ff, RR(5), MM(5), Hrho(5), PRES(5),
     !kmtocm, time
      INCLUDE 'MW__LOG.FOR'
      INTEGER I, J
      PARAMETER (kmtocm = 1.D5)
*
C Plotting data for 100%, 85.7%, 42.2%, 25.0%, and 0.0001%
C of the total mass to correlate with that of May and
C White's figure 2.
*
      Time = ETIME + ADJ
C ADJ is to adjust the time due to integration errors of
C order delT(t1).
```

```
      RR(1)=R(3,400)
      RR(2)=R(3,380)
      RR(3)=R(3,300)
      RR(4)=R(3,252)
      RR(5)=R(3,4)
      MM(1)=M(3,400)
      MM(2)=M(3,380)
      MM(3)=M(3,300)
      MM(4)=M(3,252)
      MM(5)=M(3,4)
      Hrho(1)=rho(3,399)
      Hrho(2)=rho(3,379)
      Hrho(3)=rho(3,299)
      Hrho(4)=rho(3,251)
      Hrho(5)=rho(3,3)
      PRES(1)=P(3,399)
      PRES(2)=P(3,379)
      PRES(3)=P(3,299)
      PRES(4)=P(3,251)
      PRES(5)=P(3,3)
      IF (N .EQ. 0) I = 0
         NUM(I) = N
         DO 10 J = 1, 5, 1
* THIS PLOTS R(mu,t) FOR 100%, 85.7%, 42.2%, 25%, and
* 0.0001% OF MASS
         Y1(J,I)=RR(J)/IRad(J)

X1(J,I)=PI/2.D0-(TIME*DSQRT(8.D0*PI*G*Irho(J)/3.D0))
* THIS PLOTS R(t) FOR 100%, 85.7%, 42.2%, 25%, and 0.0001%
* OF MASS
         Y2(J,I) = RR(J)
         X2(J,I) = TIME
* THIS PLOTS RHO & P FOR 100%, 85.7%, 42.2%, 25%, and
* 0.0001% OF MASS
         Y3(J,I) = Hrho(J) * Msun / (kmtocm**3) * 1.d3 !g/cc
         X3(J,I) = PRES(J) * Msun / kmtocm * 1.d3    !erg/cc
  10  CONTINUE
         ff = X1(1,I)
*               !TO STOP THE PROGRAM WHEN X TURNS NEGATIVE.
         I=I+1
      RETURN
      END
*===========================================================
*  THIS OUTPUT SUBROUTINE GOES TO THE SCREEN.
*
      SUBROUTINE OUTONE(ff)
      DOUBLE PRECISION ff
      INCLUDE 'MW__LOG.FOR'
*
      IF (N .EQ. 1) rr = R(3,JJ)
*
C  One can adjust the printed iterations by changing DD and
C  N in the following IF-THEN statement.
*
      IF (((DD .EQ. 500) .AND. (N .LE. 4000)) .OR.
     !((DD.EQ.200).AND.((N.GT. 4000) .AND. (N.LE.10000)))
     !.OR.((DD.EQ.100).AND.((N.GT.10000).AND.(N.LE.12900)))
     !.OR.((DD.EQ.20).AND.((N .GT.12900).AND.(N.LE.13400)))
     !.OR. ((DD .EQ. 5) .AND. (N .GT. 13400)))  THEN
*
      IF (ff .GE. -1.D-4) CALL PLOTS(ff)
      CALL OUTPUT
       PRINT*, 'N =',N,' R(3,JJ) =',R(3,JJ)
       IF (R(3,JJ) .LT. rr) THEN
       PRINT*,'THE STAR`S RADIUS HAS COLLAPSED BY A TOTAL OF

     !', rr-R(3,JJ),' km'
       ELSE IF (R(3,JJ) .GT. rr) THEN
     PRINT*,'THE STAR`S RADIUS HAS EXPANDED BY A TOTAL OF ',
     !R(3,JJ)-rr,' km'
       ELSE
       PRINT*,'NO CHANGE '
       ENDIF
      DD = 0     !RESET OUTPUT COUNTER.
      ENDIF
      RETURN
      END
*
*===========================================================
C IF THE READER DESIRES AN UP TO DATE COPY OF THIS PROGRAM,
C FEEL FREE TO CONTACT THE AUTHOR.
```

VITA

STEVEN RICHARD NARF

Candidate for the Degree of

Master of Science

Thesis: NUMERICAL ANALYSIS OF THE STELLAR DYNAMICS OF A HYPOTHETICAL STRANGE QUARK STAR

Major Field: Physics

Biographical:

Personal Data: Born in the city of Madison, Wisconsin, USA, on the 18th of April, 1968; the son of Mr. Richard P. Narf and Mrs. Patrica L. Narf.

Education: I graduated from Stoughtion High School in Stoughtion Wisconsin, in July of 1986. I received a Bachelor of Science Degree in Physics and in Astronomy from Mankato State University in Mankato Minnesota, USA, on June 12, 1992; completed the requirements for the Master of Science Degree at Oklahoma State University in July, 1995.

Professional Experience: Teaching Assistant for the Department of Physics, Oklahoma State University, from 1992 to July, 1995. Photometry data collecting at Andreas Obervatory located at Mankato State University, from September 1991 to June 1992. Observational Assistant at Leo V. Standeford Observatory located at Mankato State University, from 1987 to 1992.