Allgemeine Relativitätstheorie mit dem Computer

PC-POOL RAUM 01.120
JOHANN WOLFGANG GOETHE UNIVERSITÄT
08. JUNI, 2018

MATTHIAS HANAUSKE

FRANKFURT INSTITUTE FOR ADVANCED STUDIES
JOHANN WOLFGANG GOETHE UNIVERSITÄT
INSTITUT FÜR THEORETISCHE PHYSIK
ARBEITSGRUPPE RELATIVISTISCHE ASTROPHYSIK
D-60438 FRANKFURT AM MAIN
GERMANY

9. Vorlesung

Plan für die heutige Vorlesung

- Wiederholung: Parallele Programmierung mit OpenMP und MPI, das parallele C++ Programm zum Berechnen der Tolman-Oppenheimer-Volkoff (TOV) Gleichungen einer Sequenz von Neutronen/Quark Sternen.
- Die OpenMP- und MPI- C++ Versionen mit geordneter Ausgabe in eine Datei, variabler Zustandsgleichung und Terminalausgabe der benötigten Zeit.

Einführung in die Parallele Programmierung

fias.uni-frankfurt.de/~hanauske/VARTC/T2/intro/Hanauske_ParallelizationTut.odp
fias.uni-frankfurt.de/~hanauske/VARTC/T2/intro/Hanauske_ParallelizationTut.pdf

Introduction

1. Parallelization on shared memory systems using OpenMP
2. Parallelization on distributed memory systems using MPI
3. Further resources

Die TOV Gleichungen

Allgemeine Relativiätstheorie mit dem Computer: Teil II

Grundlagen zur numerischen Lösung der Tolman-Oppenheimer-Volkoff Gleichung (einfaches Euler Verfahren)

Das Differentialgleichungssystem der Tolman-Oppenheimer-Volkoff (TOV) Gleichung besitzt das folgende Aussehen

$$\frac{dp}{dr} = -\frac{(p+e)\,(m+4\pi r^3 p)}{r\,(r-2m)} \tag{1}$$

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

$$\frac{d\phi}{dr} = \frac{m+4\pi r^3 p}{r\,(r-2m)} \quad , \tag{3}$$

wobei $p = p(r)$ und $e = e(r)$ der Druck und die Energiedichte der Materie darstellen, $m = m(r)$ die radiusabhängige gravitative Masse ist und die Funktion $\phi = \phi(r)$ die 00- bzw. tt-Komponente der Metrik bestimmt ($g_{00} = e^{2\phi}$; hier bezeichnet e die Eulersche Zahl!).

TOV-Gleichungen: Numerisches Vorgehen

Eine numerische Lösung der Sterneigenschaften benötigt lediglich Gleichung (1) und (2) und geht im einfachsten Fall (einfaches Euler Verfahren) nach folgendem Schema vor:

- Man definiert die Zustandsgleichung (EOS) der Sternmaterie als eine Funktion e(p).

- Man startet im Sternzentrum $r = r_0$ und legt den Wert des zentralen Druckes $p = p_0 := p(r_0)$, der zentralen Energiedichte $e = e_0 := e(r_0)$ und der Masse $m = m_0 := m(r_0) = 0$ fest. Da die TOV Gleichung (1) bei $r_0 = 0$ singulär wird, wählt man hier einen sehr, sehr kleinen Wert für r_0 (z.B. $r_0 = 10^{-14}$).

$$r = 10^{-14}\,, \quad p = p_0\,, \quad e = e_0\,, \quad m = 0 \tag{4}$$

- Die TOV Gleichungen werden als Differenzengleichungen umgeschrieben und eine kleine Schrittweite $dr = \Delta r << 1$ wird festgelegt. In einer Schleife wird dann in jedem Radiusschritt die Druck- und Massenänderung berechnet und die jeweiligen Größen beim nächsten Schritt um diesen Faktor erhöht bzw. verringert:

$$dp \;=\; -\frac{(p+e)\,(m+4\pi r^3 p)}{r\,(r-2m)}\,dr$$

TOV-Gleichungen: Numerisches Vorgehen

- Die TOV Gleichungen werden als Differenzengleichungen umgeschrieben und eine kleine Schrittweite $dr = \Delta r << 1$ wird festgelegt. In einer Schleife wird dann in jedem Radiusschritt die Druck- und Massenänderung berechnet und die jeweiligen Größen beim nächsten Schritt um diesen Faktor erhöht bzw. verringert:

$$\begin{aligned} dp &= -\frac{(p+e)\,(m+4\pi r^3 p)}{r\,(r-2m)}\,dr \\ dm &= 4\pi r^2 e\,dr \\ p &= p + dp \\ m &= m + dm \\ r &= r + dr \end{aligned}$$

- Im Laufe der iterativen Lösung verringert sich der Druck ständig. Die Schleife wird solange ausgeführt bis der Wert des Druckes gleich Null bzw. negativ wird (Abruchbedingung: $p \leq 0$), da an der Sternoberfläche der Druck verschwindet.

C++ Lösen der TOV-Gleichung

```cpp
#include <iostream>                    //Ein-/Ausgabe (Include-Dateien)
#include <math.h>                      //Mathematisches
using namespace std;                   //Fuer cout

//Definition der Zustandsgleichung
double eos(double p)
  {
    double e;
    e=pow(p/10,3.0/5);
    return e;
  }

main(void)  //Hauptprogramm
  {
    //Variablendeklarationen
    double M,p,e,r,dM,dp,de,dr;
    double eos(double);

    //Variableninitialisierung
    M=0;
    r=pow(10,-14);
    p=10*pow(0.0005,5.0/3);
    dr=0.000001;

    //do-while Schleife (Numerische Lösung der TOV-Gleichung)
    do
      {
        e=eos(p);                                  //Wert der Energiedichte bei momentanen Druck
        dM=4*M_PI*e*r*r*dr;                        //Massenzunahme bei momentanem r und Schrittweite dr
        dp=-(p+e)*(M+4*M_PI*r*r*r*p)/(r*(r-2*M))*dr;  //Druckzunahme bei momentanem r und Schrittweite dr (TOV-Gleichung)
        r=r+dr;                                    //momentaner Radius des Neutronensterns
        M=M+dM;                                    //momentane Masse des Neutronensterns innerhalb des Radius r
        p=p+dp;                                    //momentaner Druck des Neutronensterns innerhalb des Radius r
      }
      while(p>0);

    //Ausgabe der Masse und des Radius auf dem Bildschirm
    cout<<"Neutronensternradius [km]           = "<<r<<"\n";
    cout<<"Neutronensternmasse [Sonnenmassen] = "<<M/1.4766<<"\n";

    return 0;                                      //main beenden (Programmende)
  }
```

Die polytrope **Zustandsgleichung** ist als eine Funktion außerhalb des Hauptprogramms definiert

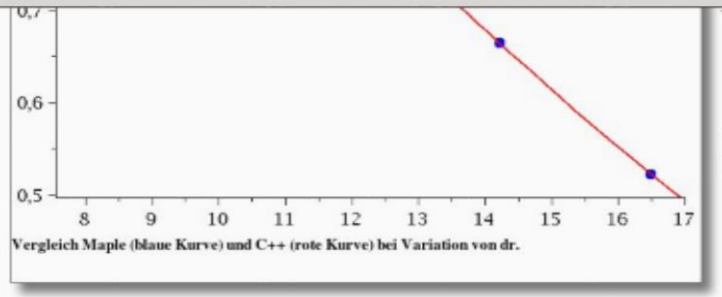
Deklaration der nötigen **Variablen** und der Zustandsgleichungsfunktion

Festlegung der **Anfangswerte** im Sternzentrum (M,r,p) und der Radiusschrittweite dr

TOV-Gleichungen

Ausgabe auf dem Bildschirm

Parallele Programme siehe Teil 2 der Internetseite der Vorlesung



Vergleich Maple (blaue Kurve) und C++ (rote Kurve) bei Variation von dr.

2.3) Parallele OpenMP-Version 1 von 2.2)

Das sequentielle Programm 2.2) wurde nun mittels OpenMP (siehe TOV OpenMP Version) parallelisiert. Hiebei wurde einfach das OpenMP-Pragma *'#pragma omp parallel for private(i,M,p,e,r,nu,dM,dp,de,dnu)'* vor die for-Schleife der unabhängigen Berechnung der einzelnen Neutronensterne geschrieben. Wichtig ist nun, dass man das Programm mit dem folgenden Befehl compiliert: 'c++ -fopenmp TOV_parallel_omp.cpp'. Führt man das Programm mit './a.out' aus, so erkennt man als erstes, dass es (in Abhängigkeit wieviele CPU-Kerne man in seinem Computer hat), viel schneller läuft. Die im Terminal ausgegebenen Werte sind jedoch nicht mehr geordnet, sondern die Berechnung der 40 Neutronensterne erfolgt parallel und ungeordnet. Die nebenstehende Abbildung zeigt die Terminalausgabe des parallelen Programms und die Auslastung der 8 CPU-Kerne meines Laptops, wobei zuerst das parallele Programm und dannach das sequentielle ausgeführt wurde.

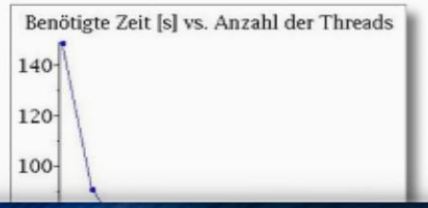
```
Datei  Bearbeiten  Ansicht  Lesezeichen  Einstellungen  Hilfe
hanauske@ITPRelAstro-Aspire-VN7-591G:~$ c++ -fopenmp TOV_parallel_omp.cpp
hanauske@ITPRelAstro-Aspire-VN7-591G:~$ ./a.out
35
Neutronensternradius [km]           = 9.13464
Neutronensternmasse [Sonnenmassen]  = 1.20785
00-Metrikkomponente im Sternzentrum = 0.198145
30
Neutronensternradius [km]           = 9.50537
Neutronensternmasse [Sonnenmassen]  = 1.21506
00-Metrikkomponente im Sternzentrum = 0.220643
25
Neutronensternradius [km]           = 9.94998
Neutronensternmasse [Sonnenmassen]  = 1.21927
00-Metrikkomponente im Sternzentrum = 0.248056
20
Neutronensternradius [km]           = 10.4976
Neutronensternmasse [Sonnenmassen]  = 1.21836
00-Metrikkomponente im Sternzentrum = 0.28223
15
Neutronensternradius [km]           = 11.1977
Neutronensternmasse [Sonnenmassen]  = 1.20841
00-Metrikkomponente im Sternzentrum = 0.326145
10
Neutronensternradius [km]           = 12.1444
```

2.4) Parallele OpenMP-Version 2 (2.3) mit geordneter Terminal-Ausgabe)

Diese Version entspricht Version 2.3) mit einer geordneten Ausgabe. Die geordnete Ausgabe wird hierbei realisiert, indem für die drei ausgegebenen, numerischen Werte (Radius, Masse, zentraler g_{00}-Wert), drei Datenfelder (Arrays) der Länge 40 eingerichtet werden. Nachdem ein OpenMP-Thread mit seiner Berechnung fertig ist, speichert er sein individuelles Ergebnis in die spezifische Position innerhalb des Arrays und berechnet den nächsten Stern. Die geordnete Ausgabe aller Werte erfolgt dann sequentiell, außerhalb der parallelisierten Schleife.

2.5) Parallele OpenMP-Version 3 (2.4) mit Ausgabe in eine Datei)

Diese Version entspricht Version 2.4) wobei die geordnete Ausgabe nun nicht mehr in dem Terminal geschieht, sondern die berechneten Werte werden in eine externe Datei (tov.txt) ausgegeben - die Ausgabedatei erfolgt im Unterordner 'output', welcher vor dem Ausführen des Programms angelegt werden muss. Der Vorteil hierbei ist, dass nachdem das Programm ausgefürt wurde, die Ergebnisse einfacher verarbeitet und dargestellt werden können. So kann man z.B. mittels Gnuplot sich das Radius-Masse Diagramm darstellen. Noch einfacher, kann man sich die einzelnen Gnuplot-Befehle zum Darstellen diverser Diagramme in ein ausführbares Shell-Script schreiben, das dann automatisch die jeweiligen Plots erzeugt (siehe Gnuplot Shell-Script).



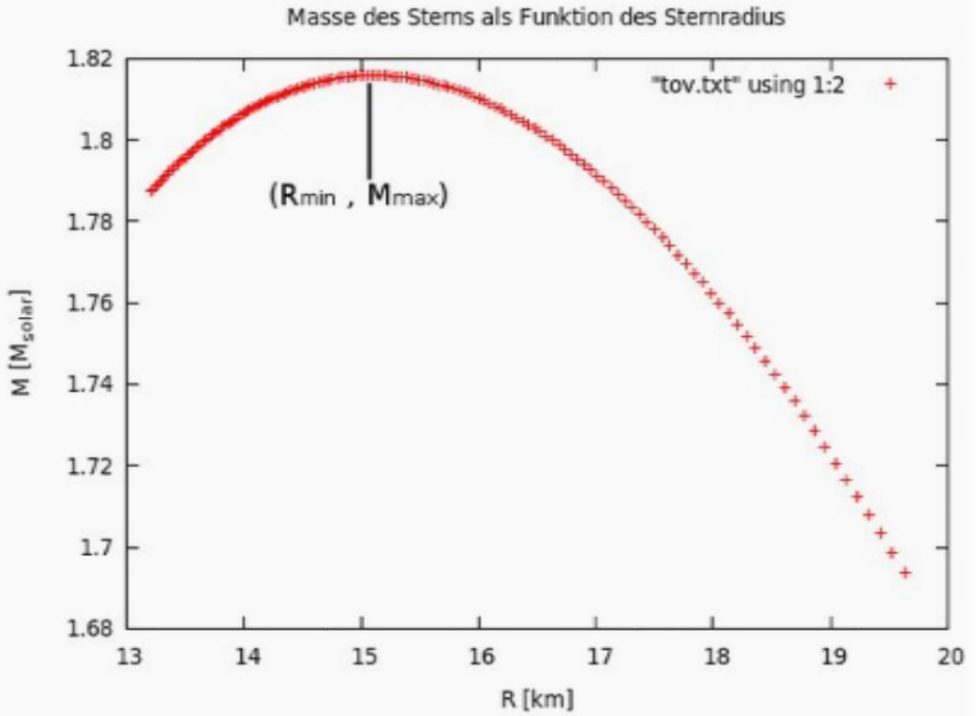
2.6) Parallele OpenMP-Version mit geordneter Ausgabe in eine Datei und variabler Zustandsgleichung

Diese Version entspricht Version 2.5), wobei die als Funktion definierte Zustandsgleichung variabler gestaltet wurde (EOS: ($e(P, K, \gamma) = (P/K)^{1/\gamma}$) für Neutronensterne und Weiße Zwerge bzw. (e(P,Bag)=3 p + 4 Bag) für Quarksterne im MIT-Bag Model).

Struktur und Performance des parallelen OpenMP - C++ Programms

Aufgaben zum 2.Teil der Vorlesung siehe E-Learning „Lon Capa“

Berechnen Sie unter Verwendung des C++ Programms aus Teil II der Vorlesung die maximale Masse M_{max} in $[\mathrm{M}_\odot]$ und den zugehörigen minimalen Radius R_{min} eines Neutronensterns in [km]. Verwenden Sie eine polytrope Zustandsgleichung der Form $p = K * e^{\gamma}$, wobei $\gamma = 5/3$ und $K = 20.25 \left[\mathrm{km}^{4/3}\right]$ ist.

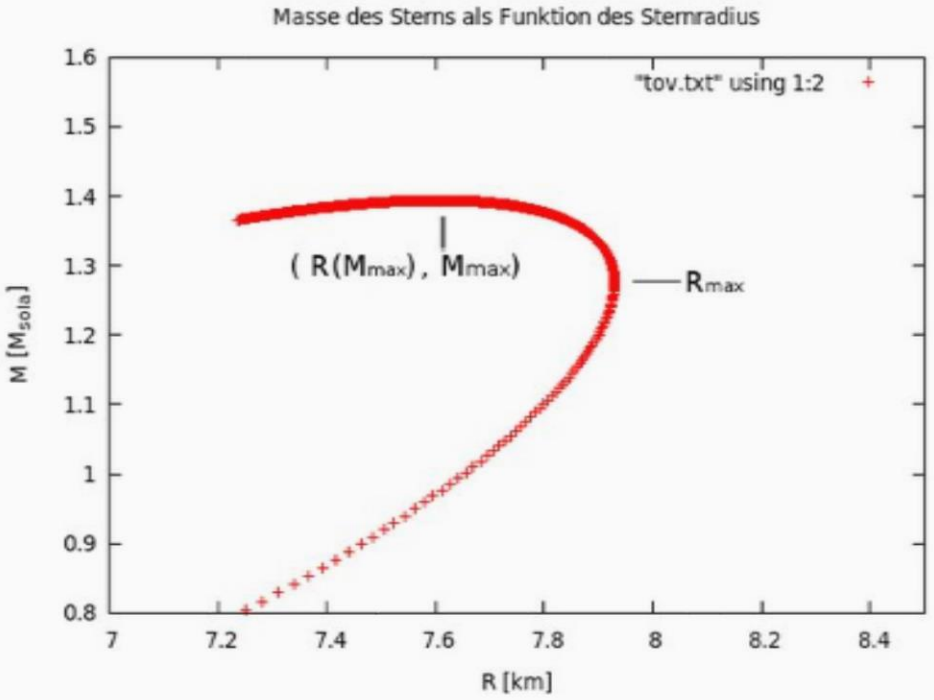


M_{max} = _____ , R_{min} = _____

Antwort einreichen Versuche 0/20

Aufgaben zum 2.Teil der Vorlesung siehe E-Learning „Lon Capa“

Berechnen Sie unter Verwendung des C++ Programms aus Teil II der Vorlesung die maximale Masse M_{max} in $[\mathrm{M}_\odot]$ und den zugehörigen Radius $R(M_{max})$ eines Quarkstern Modells in [km]. Verwenden Sie die lineare Zustandsgleichung des MIT-Bag Modells $p = \frac{1}{3}\left(e - 4 * B\right)$;, wobei der Parameter B die für das Confinement nötige Bag Konstante ist; verwenden Sie B=0.000152869496944 (entspricht ungefähr $B^{1/4} =$ 194 [MeV]). Geben Sie desweiteren auch dem maximalen Radius R_{max} des Quarksternmodells an.

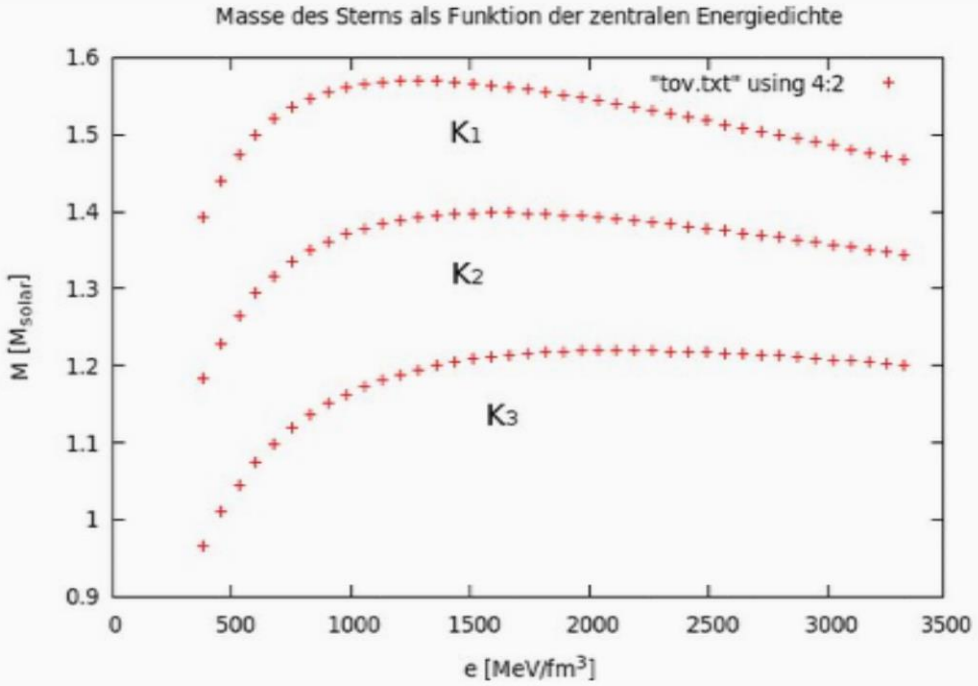


M_{max} = , $R(M_{max})$ = , R_{max} =

Antwort einreichen Versuche 0/20

Aufgaben zum 2.Teil der Vorlesung siehe E-Learning „Lon Capa“

Die maximale Masse M_{max} eines Neutronensterns sei gegeben, und die zugrunde liegende Zustandsgleichung der Neutronensternmaterie sei durch folgenden polytropen Ansatz $p = K * e^{\gamma}$ bestimmt, wobei $\gamma = 5/3$ und $K =$ eine noch zu bestimmende unbekannte Konstante ist. Bei Variation von K ändert sich das gesamte Masse-Radius, bzw. Masse-zentrale Energiedichte Profil einer Sequenz von Sternen und der Wert der maximale Masse M_{max} verschiebt sich (siehe nebenstehende Abbildung). Berechnen Sie unter Verwendung des C++ Programms aus Teil II der Vorlesung den Wert der Konstanten K in $\left[\mathrm{km}^{4/3}\right]$ und geben Sie den zugehörigen Radius des maximalen Massen Sterns ($R_{M_{max}}$) an. Der Wert der maximalen Masse beträgt $M_{max} = 1.735147\ [\mathrm{M}_{\odot}]$.



$K =$, $R_{M_{max}} =$

Antwort einreichen Versuche 0/20

08. Juni 2018, 21.00 Uhr: Night of Science 2018, H5, Campus Riedberg in Frankfurt am Main

M.Hanauske: Tanz der Neutronensterne