Numerical Relativity - PHY 6938

Solutions to HW 9

1. Recall $dE = TdS - pdV + \mu dN$ and $E = TS - pV + \mu N$
a)
$d\rho = dE/V - E/V^2 dV = (dE - \rho dV)/V$
$dS = Nds + sdN$
$dN = Vdn + ndV$
Thus $d\rho = [T(Nds+sdN)-pdV+\mu(Vdn+ndV)-\rho dV]/V = [T(Nds+s(Vdn+ndV))-pdV+\mu(Vdn+ndV)-\rho dV]/V = T(nds+sdn)+TsndV/V-pdV/V+\mu dn+\mu ndV/V-\rho dV/V = Tnds+(Ts+\mu)dn+(Tsn-p+\mu n-\rho)dV/V$. Now note that the last bracket is zero because

$$TS - pV + \mu N = E,$$

so $d\rho = Tnds + (Ts + \mu)dn = Tnds + (TS + \mu N)dn/N = Tnds + (E + pV)dn(V/N)/V$, which gives

$$d\rho = Tnds + \frac{\rho + p}{n} dn.$$

b) $\rho_0 = m_0 n$. Thus

$$d\rho = T(\rho_0/m_0)ds + \frac{\rho + p}{\rho_0} d\rho_0.$$

2. We define ϵ by $\rho = (1 + \epsilon)\rho_0$. Thus $h = (\rho + p)/\rho_0$, which gives $dh = (d\rho + dp)/\rho_0 - (\rho + p)d\rho_0/\rho_0^2 = (T/m_0)ds + \frac{\rho+p}{\rho_0^2} d\rho_0 + dp/\rho_0 - (\rho + p)d\rho_0/\rho_0^2$, so

$$dh = (T/m_0)ds + dp/\rho_0$$

a) From $p = \kappa\rho_0^{1+1/n}$ we get $dp = (1 + 1/n)\kappa\rho_0^{1/n} d\rho_0$. For $T = 0$ we then have

$$dh = (1 + 1/n)\kappa\rho_0^{1/n-1} d\rho_0$$

which yields

$$h = 1 + (n + 1)\kappa\rho_0^{1/n}.$$

So:

$$\rho_0 = \left(\frac{h-1}{(n+1)\kappa}\right)^n, \quad p = \rho_0 \frac{h-1}{n+1}, \quad \rho = h\rho_0 - p = \rho_0 + np, \quad \epsilon = np/\rho_0$$

b) Thus

$$\rho_0 = \left(\frac{p}{\kappa}\right)^{n/(n+1)}, \quad \rho = \rho_0 + np, \quad \epsilon = np/\rho_0, \quad h = 1 + (n+1)\kappa \left(\frac{p}{\kappa}\right)^{1/(n+1)}$$

3. TOV star
a) From 2. b)

$$\rho_0 = \left(\frac{p}{\kappa}\right)^{n/(n+1)}, \quad \rho = np + \rho_0$$

b) $m(r) \approx \frac{4\pi}{3} r^3 \rho$ gives:

$$\frac{dm}{dr} = 4\pi r^2 \rho \tag{1}$$

$$\frac{dp}{dr} = -(\rho + p)((4\pi/3)\rho + 4\pi p) r/(1 - (8\pi/3)\rho r^2) \tag{2}$$

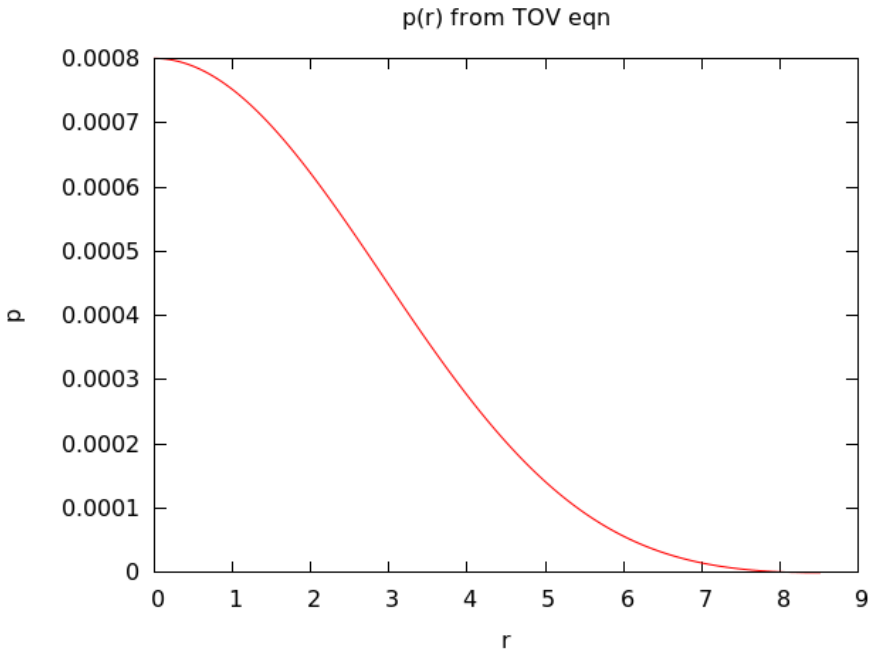
$$\frac{d\phi}{dr} = ((4\pi/3)\rho + 4\pi p) r/(1 - (8\pi/3)\rho r^2). \tag{3}$$

c) see next pages

```
# import some packages
from __future__ import print_function
import numpy as np

# some constants
PI = np.pi
A = 4*PI;
Ao3 = A/3.0;

########################
# define some functions:
########################

# rhoE from P
# rho0 = pow(P/kappa, n/(n+1)
# rhoE = n*P + k*rho0,  k=1 for a single polytrope
def rhoE_OF_P(P, n, kappa):
    k = 1.0
    rho0 = np.power(P/kappa, n/(n+1))
    rhoE = n*P + k*rho0
    return rhoE

# RHS of $\partial_r u$
def set_rhs(u, rhs, r, n, kappa):
    m = u[0]
    P = u[1]
    Phi = u[2]
    rhoE = rhoE_OF_P(P, n, kappa)

    # TOV ODEs
    dm_dr = A*rhoE*r*r;
    if r > 1e-4:
        dP_dr   =-(rhoE+P)*(m+A*r*r*r*P)/(r*(r-2.0*m));
        dPhi_dr = (m+A*P*r*r*r)/(r*(r-2.0*m));
    else: #  m ~ Ao3 * rhoE * r*r*r
        dP_dr   =-(rhoE+P)*(Ao3*rhoE+A*P)*r / (1.0-2.0*Ao3*rhoE*r*r);
        dPhi_dr = (Ao3*rhoE + A*P)*r / (1.0-2.0*Ao3*rhoE*r*r);

    rhs[0] = dm_dr
    rhs[1] = dP_dr
    rhs[2] = dPhi_dr
    return rhs

# u(r+dr) = u(r) + rhs * dr
def calc_unew(u, rhs, dr):
    return u + rhs * dr

# print colums with data at r
def pr_data(r, u):
    print(r, end="  ")
    for i in range(0, len(u)):
        print(u[i], end="  ")
    print()


###############
# main program:
###############

# step size
dr = 0.0002

# make evo vec. u and its rhs
```

```
N = 3
u = np.empty([N])
rhs = np.empty([N])

# EoS and central pressure
n = 1.0
kappa = 123.6489
Pc = 8.2e-4

# initial vals
r = 0.0
m = 0.0
P = Pc
Phi = 1.0

# convert to u vec.
u[0] = m
u[1] = P
u[2] = Phi

print("# r, m, P, Phi")
# make steps until P<0
while P >= 0.0:
    # print colums with data
    pr_data(r, u)
    # make step, using simple Euler method
    r = r + dr
    set_rhs(u, rhs, r, n, kappa)
    u = calc_unew(u, rhs, dr)
    P = u[1]

# we still need to add a const to Phi in u[2] to make metric continuous
```

d) For $p_c = 8 \times 10^{-4}$ we find



e) We find the maximum mass near $p_c = 0.00082 \Rightarrow m = 1.8206, \quad r_* = 8.487$. See program below:

```
# import some packages
from __future__ import print_function
import numpy as np

# some constants
PI = np.pi
A = 4*PI;
Ao3 = A/3.0;

########################
# define some functions:
########################

# rhoE from P
# rho0 = pow(P/kappa, n/(n+1)
# rhoE = n*P + k*rho0,  k=1 for a single polytrope
def rhoE_OF_P(P, n, kappa):
    k = 1.0
    rho0 = np.power(P/kappa, n/(n+1))
    rhoE = n*P + k*rho0
    return rhoE

# RHS of $\partial_r u$
def set_rhs(u, rhs, r, n, kappa):
    m = u[0]
    P = u[1]
    Phi = u[2]
    rhoE = rhoE_OF_P(P, n, kappa)

    # TOV ODEs
    dm_dr = A*rhoE*r*r;
    if r > 1e-4:
        dP_dr   =-(rhoE+P)*(m+A*r*r*r*P)/(r*(r-2.0*m));
        dPhi_dr = (m+A*P*r*r*r)/(r*(r-2.0*m));
    else: #  m ~ Ao3 * rhoE * r*r*r
        dP_dr   =-(rhoE+P)*(Ao3*rhoE+A*P)*r / (1.0-2.0*Ao3*rhoE*r*r);
        dPhi_dr = (Ao3*rhoE + A*P)*r / (1.0-2.0*Ao3*rhoE*r*r);

    rhs[0] = dm_dr
    rhs[1] = dP_dr
    rhs[2] = dPhi_dr
    return rhs

# u(r+dr) = u(r) + rhs * dr
def calc_unew(u, rhs, dr):
    return u + rhs * dr

# print colums with data at r
def pr_data(r, u):
    print(r, end="  ")
    for i in range(0, len(u)):
        print(u[i], end="  ")
    print()


###############
# main program:
###############

# step size
dr = 0.0002

# make evo vec. u and its rhs
```

```
N = 3
u = np.empty([N])
rhs = np.empty([N])

# EoS and central pressure
n = 1.0
kappa = 123.6489

print("# Pc, ms, rs")
Pc = 0.0004
while Pc <= 0.0018:
    # initial vals
    r = 0.0
    m = 0.0
    P = Pc
    Phi = 1.0

    # convert to u vec.
    u[0] = m
    u[1] = P
    u[2] = Phi

    # make steps until P<0
    while P >= 0.0:
        # save data
        rs = r
        ms = u[0]
        # make step, using simple Euler method
        r = r + dr
        set_rhs(u, rhs, r, n, kappa)
        u = calc_unew(u, rhs, dr)
        P = u[1]

        # we still need to add a const to Phi in u[2] to make metric continuous

    # print Pc, ms, rs
    print(Pc, ms, rs)
    Pc += 0.00002
```