TOVST_solver_momin

July 16, 2022

[ ]:
```
# source: https://www.codesansar.com/numerical-methods/
#         runge-kutta-fourth-order-rk4-python-program.htm

import os
import cmath #To help us out with the complex square root
import numpy as np #For the arrays
import matplotlib.pyplot as plt #Visualization

# deleting profile and radmass files
# os.remove("radmass.txt")
# os.remove("profile.txt")

# the TOV GR equation

def b1(r,P,f,b,params):
    PCC=params[0]
    Q=params[1]
    eta=params[2]
    Y=params[3]
    return (1-f)*b/(r*f)

def P1(r,P,f,b,params):
    PCC=params[0]
    Q=params[1]
    eta=params[2]
    Y=params[3]
    return -(eden(P,params)+P) * b1(r,P,f,b,params)/(2*b) -␣
 ↪2*sig(P,params,b,f,r)/r

def f1(r,P,f,b,params):
    PCC=params[0]
    Q=params[1]
    eta=params[2]
    Y=params[3]
    A = r*b/f*(P*r**2+4*kappa)-3*eta*Q**2*r
    B = 3*(1-f)*eta*Q**2
```

```
    B = B + b/f*(6*r**2*f*P + (1+f)*r**2*eden(P,params) -4*kappa*(1-f)
 ↪-4*r**2*f*sig(P,params,b,f,r))
    return -B/A

def om1(r,P,f,b,om,ka,params):
    PCC=params[0]
    Q=params[1]
    eta=params[2]
    Y=params[3]
    return 6*ka/r**4

def ka1(r,P,f,b,om,ka,params):
    PCC=params[0]
    Q=params[1]
    eta=params[2]
    Y=params[3]
    # om2 = K_1 om1 + K_2 om
    K_2 = 4*b*(eden(P,params)+P-sig(P,params,b,f,r))
    K_2 = K_2/(f*(-Q**2*eta*f+b*(4*kappa+r**2*P)))
    K_1 = 4*Q**2*eta*f**2
    K_1 = K_1 + b*( -4*f*(4*kappa+r**2*P) + r**2*(eden(P,params)+P) )
    K_1 = K_1/(r*f*(-Q**2*eta*f+b*(4*kappa+r**2*P)))
    return (4/r + K_1)*ka + (r**4/6)*K_2*om
```

[ ]:
```
# this is only for a single PCC
def single_PCC(PCC,Qinf,eta,Y,bCC,radmassdata,profile):
    # define initial parameters
    rCC = .000000001 # radius near center in m--the starting point
    rmax = 100000. # radius at far distances in m
    # PCC = PCC_Input # pressure at the center in MeV / fm^3
    fCC = 1. # metric function f(r) at the center
    # bCC = bCC_Input # metric function b(r) at the center

    h = 1. # h-step

    UBQ = np.sqrt(4*kappa/3)
    Qinf = Qinf*UBQ #.29*UBQ   # not more than .29*UBQ is save up to PCC=1200
    # eta = eta_Input
    Q = Qinf
    params = np.array([PCC,Q,eta,Y])
    # print(params[0]/UBQ,eta)

    # calculate the surface values
    output=RungeKutta(rCC,bCC,PCC,fCC,params,h,0)
    # print(output)

    # at the surface, b = 1-2Gm/r, which is different to the result
```

```
    rSurface=output[0]
    bSurface=output[1]
    mSurface=output[3]
    # print(PCC, rSurface, mSurface,
        #   GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ)
    bSurfaceTarget = 1-2*GS*MSS*mSurface/rSurface

    # NOTICE: cannot use it since R and M chaages
    # instead we follow the paper by Cisterna PRD92,044050(2015)
    # i.e. bCC will not be modified, but Q is modified instead

    # Or rather inputting Qinf fixed instead and calculate Q,
    # which is done by only modifying bCC,
    # then Q=Q/np.sqrt(bCorrection)

    # So, we redefine bCC as follows
    bCorrection=bSurfaceTarget/bSurface  # bCorrection=1/binf
    # print(abs(bCorrection-1))
    # bCC and Q will be modified into bCC*bCorrection
    # and Q*np.sqrt(bCorrection)

    # if abs(bCorrection) not near 1, then recalculate
    # bCC=bCC*bCorrection
    # Q=Q/np.sqrt(bCorrection)
    while (abs(bCorrection-1)>10**(-3)):
        bCC=bCC*bCorrection
        Q=Q/np.sqrt(bCorrection)
        params = np.array([PCC,Q,eta,Y])
        # print(params[0]/UBQ,eta)
        output=RungeKutta(rCC,bCC,PCC,fCC,params,h,0)
        # print(output)
        rSurface=output[0]
        bSurface=output[1]
        mSurface=output[3]
        # print(PCC, rSurface, mSurface,
        #         GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ)
        bSurfaceTarget = 1-2*GS*MSS*mSurface/rSurface
        bCorrection=bSurfaceTarget/bSurface
        # print(abs(bCorrection-1))
        # the result R and M does change
        # so this loop is essential
    # We NEED to redefine both b and Q
    # so that R and M don'o't change
    # --they change if we didn't
    # redefine Q

    rSurface=output[0]
```

```
    bSurface=output[1]
    mSurface=output[3]
    print(PCC, rSurface, mSurface,
                GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ)

    if profile==1:
        output=RungeKutta(rCC,bCC,PCC,fCC,params,h,1)
    if radmassdata == 1:
        print(PCC, (eden(PCC,params)/1000), (rSurface/1000), mSurface,
            GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ,
            file=open('radmassST.dat', 'a'))

    output = np.array([PCC, (eden(PCC,params)/1000), (rSurface/1000), mSurface,␣
↪GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ])
    return output

# define the Runge-Kutta 4th order for the problem
# if we want to print the profile, set profile=1
# if we not, set profile=0
def RungeKutta(rCC,bCC,PCC,fCC,params,h,profile):
    # input initial values
    r0 = rCC
    b0 = bCC
    P0 = PCC
    f0 = fCC
    m0 = (1-f0)*r0/(2*GS*MSS)
    while (P0 > 0.):
        # print(r0, b0, P0, m0)
        if profile == 1:
            print(r0, b0, P0, m0, file=open('profileST.dat', 'a'))
        # calculate k1
        r01 = r0
        b01 = b0
        P01 = P0
        f01 = f0
        k1_b = h * b1(r01,P01,f01,b01,params)
        k1_P = h * P1(r01,P01,f01,b01,params)
        k1_f = h * f1(r01,P01,f01,b01,params)
        # calculate k2
        r01 = r0 + h/2
        b01 = b0 + k1_b/2
        P01 = P0 + k1_P/2
        f01 = f0 + k1_f/2
        k2_b = h * b1(r01,P01,f01,b01,params)
        k2_P = h * P1(r01,P01,f01,b01,params)
        k2_f = h * f1(r01,P01,f01,b01,params)
        # calculate k3
```

```
        r01 = r0 + h/2
        b01 = b0 + k2_b/2
        P01 = P0 + k2_P/2
        f01 = f0 + k2_f/2
        k3_b = h * b1(r01,P01,f01,b01,params)
        k3_P = h * P1(r01,P01,f01,b01,params)
        k3_f = h * f1(r01,P01,f01,b01,params)
        # calculate k4
        r01 = r0 + h
        b01 = b0 + k3_b
        P01 = P0 + k3_P
        f01 = f0 + k3_f
        k4_b = h * b1(r01,P01,f01,b01,params)
        k4_P = h * P1(r01,P01,f01,b01,params)
        k4_f = h * f1(r01,P01,f01,b01,params)
        # calculate the next r0, P0, m0, and b0
        r0 = r0 + h
        b0 = b0 + (k1_b+2*k2_b+2*k3_b+k4_b)/6
        P0 = P0 + (k1_P+2*k2_P+2*k3_P+k4_P)/6
        f0 = f0 + (k1_f+2*k2_f+2*k3_f+k4_f)/6
        m0 = (1-f0)*r0/(2*GS*MSS)
        if P0 > 2*PCC:
            print(PCC,"dP/dr>0")
            break
    # the results at the surface
    if profile == 1:
        print(r0, b0, P0, m0, file=open('profileST.dat', 'a'))
    return np.array([r0,b0,P0,m0,f0])
```

[ ]:
```
def single_momin(PCC,Qinf,eta,Y,bCC,radmassdata,profile):
    # define initial parameters
    rCC = .000000001 # radius near center in m--the starting point
    rmax = 100000. # radius at far distances in m
    fCC = 1. # metric function f(r) at the center

    h = 1. # h-step

    UBQ = np.sqrt(4*kappa/3)
    Qinf = Qinf*UBQ
    Q = Qinf
    params = np.array([PCC,Q,eta,Y])
    output=RungeKutta(rCC,bCC,PCC,fCC,params,h,0)

    rSurface=output[0]
    bSurface=output[1]
    mSurface=output[3]
    bSurfaceTarget = 1-2*GS*MSS*mSurface/rSurface
```

```
    bCorrection=bSurfaceTarget/bSurface

    Pcont=PCC
    while (abs(bCorrection-1)>10**(-3)):
        # print("tes1",abs(bCorrection-1))
        bCorr_old=bCorrection
        bCC=bCC*bCorrection
        Q=Q/np.sqrt(bCorrection)
        params = np.array([PCC,Q,eta,Y])
        output=RungeKutta(rCC,bCC,PCC,fCC,params,h,0)
        PSurface=output[2]
        if PSurface>PCC:
            # print("tes1: dP/dr>0")
            Pcont=PSurface
            break
        rSurface=output[0]
        bSurface=output[1]
        mSurface=output[3]
        bSurfaceTarget = 1-2*GS*MSS*mSurface/rSurface
        bCorrection=bSurfaceTarget/bSurface
        bCorr_new=bCorrection
        if abs(bCorr_new-1)>abs(bCorr_old-1):
            print(abs(bCorr_new-1),abs(bCorr_old-1))
            break


    if Pcont>PCC:
        print(PCC, PSurface)
        output = np.array([PCC, PSurface])
    else:
        rSurface=output[0]
        bSurface=output[1]
        mSurface=output[3]
        # print(PCC, rSurface, mSurface,
        #             GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ)

        # kelar hitung TOV, lanjut ke momen inersia
        omCC = 1.e-8
        omCC = omCC + (8./5)*GS*PI*rCC**2*(PCC+eden(PCC,params))*omCC
        kaCC = (8./15)*GS*PI*rCC**2*(PCC+eden(PCC,params))*omCC
        output = RungeKutta1(rCC,bCC,PCC,fCC,omCC,kaCC,params,h,0)
        # at the surface, moment of inertia is momin
        rSurface=output[0]
        bSurface=output[1]
        mSurface=output[3]
        # bSurfaceTarget = 1-2*GS*MSS*mSurface/rSurface
        # bCorrection=bSurfaceTarget/bSurface
```

```
        omSurface=output[5]
        kaSurface=output[6]
        zeta=1/(omSurface+2*kaSurface/(rSurface**3))
        momin=kaSurface*zeta/GS
        momin1=momin/(MSS*mSurface*rSurface**2)  # I/MR^2 tanpa satuan, diplot␣
↪thd compactness
        momin2=momin1*1.98892e33*1.e10*mSurface*(rSurface*1.e-3)**2/1.e45 # I␣
↪satuan 10^45 g cm^2 diplot thd massa
        print(PCC, rSurface, mSurface, GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ,␣
↪Q/UBQ, momin1, momin2, zeta)
        if Q/UBQ >= 1: print("Q>UBQ")

        if profile==1:
            output=RungeKutta1(rCC,bCC,PCC,fCC,omCC*zeta,kaCC*zeta,params,h,1)
        if radmassdata == 1 and Q/UBQ < 1:
            print(PCC, (eden(PCC,params)/1000), (rSurface/1000), mSurface,
                GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ,
                momin1, momin2, omCC*zeta, kaCC*zeta,
                file=open('radmassSTmomin.dat', 'a'))

        output = np.array([PCC, (eden(PCC,params)/1000), (rSurface/1000),␣
↪mSurface, GS*MSS*mSurface/rSurface, bCC, Qinf/UBQ, Q/UBQ,  momin1, momin2,␣
↪omCC*zeta, kaCC*zeta])
    return output

def RungeKutta1(rCC,bCC,PCC,fCC,omCC,kaCC,params,h,profile):
    # input initial values
    r0 = rCC
    b0 = bCC
    P0 = PCC
    f0 = fCC
    m0 = (1-f0)*r0/(2*GS*MSS)
    om0 = omCC
    ka0 = kaCC
    while (P0 > 0.):
        if profile == 1:
            print(r0, b0, P0, m0, om0, ka0, file=open('profileSTmomin.dat',␣
↪'a'))
        # calculate k1
        r01 = r0
        b01 = b0
        P01 = P0
        f01 = f0
        om01 = om0
        ka01 = ka0
        k1_b = h * b1(r01,P01,f01,b01,params)
        k1_P = h * P1(r01,P01,f01,b01,params)
```

```
        k1_f = h * f1(r01,P01,f01,b01,params)
        k1_om = h * om1(r01,P01,f01,b01,om01,ka01,params)
        k1_ka = h * ka1(r01,P01,f01,b01,om01,ka01,params)
        # calculate k2
        r01 = r0 + h/2
        b01 = b0 + k1_b/2
        P01 = P0 + k1_P/2
        f01 = f0 + k1_f/2
        om01 = om0 + k1_om/2
        ka01 = ka0 + k1_ka/2
        k2_b = h * b1(r01,P01,f01,b01,params)
        k2_P = h * P1(r01,P01,f01,b01,params)
        k2_f = h * f1(r01,P01,f01,b01,params)
        k2_om = h * om1(r01,P01,f01,b01,om01,ka01,params)
        k2_ka = h * ka1(r01,P01,f01,b01,om01,ka01,params)
        # calculate k3
        r01 = r0 + h/2
        b01 = b0 + k2_b/2
        P01 = P0 + k2_P/2
        f01 = f0 + k2_f/2
        om01 = om0 + k2_om/2
        ka01 = ka0 + k2_ka/2
        k3_b = h * b1(r01,P01,f01,b01,params)
        k3_P = h * P1(r01,P01,f01,b01,params)
        k3_f = h * f1(r01,P01,f01,b01,params)
        k3_om = h * om1(r01,P01,f01,b01,om01,ka01,params)
        k3_ka = h * ka1(r01,P01,f01,b01,om01,ka01,params)
        # calculate k4
        r01 = r0 + h
        b01 = b0 + k3_b
        P01 = P0 + k3_P
        f01 = f0 + k3_f
        om01 = om0 + k3_om
        ka01 = ka0 + k3_ka
        k4_b = h * b1(r01,P01,f01,b01,params)
        k4_P = h * P1(r01,P01,f01,b01,params)
        k4_f = h * f1(r01,P01,f01,b01,params)
        k4_om = h * om1(r01,P01,f01,b01,om01,ka01,params)
        k4_ka = h * ka1(r01,P01,f01,b01,om01,ka01,params)
        # calculate the next r0, P0, m0, and b0
        r0 = r0 + h
        b0 = b0 + (k1_b+2*k2_b+2*k3_b+k4_b)/6
        P0 = P0 + (k1_P+2*k2_P+2*k3_P+k4_P)/6
        f0 = f0 + (k1_f+2*k2_f+2*k3_f+k4_f)/6
        om0 = om0 + (k1_om+2*k2_om+2*k3_om+k4_om)/6
        ka0 = ka0 + (k1_ka+2*k2_ka+2*k3_ka+k4_ka)/6
        m0 = (1-f0)*r0/(2*GS*MSS)
```

```
    # the results at the surface
    if profile == 1:
        print(r0, b0, P0, m0, om0, ka0, file=open('profileSTmomin.dat', 'a'))
    return np.array([r0,b0,P0,m0,f0,om0,ka0])
```

[ ]:
```
# some constants
GS = 1.325 * 10**(-12) # Newton constant in m^4 / MeV fm^3
MSS = 1.1155 * 10**(15) # Sun's mass in MeV m^3 / fm^3
PI = np.pi
HC = 197.327 # hc=1=197.327 MeV fm
kappa = 1/(16*PI*GS)

UBQ = np.sqrt(4*kappa/3)

# PCC = 300. # pressure at the center in MeV / fm^3
# bCC = 1. # metric function b(r) at the center
# LBQ2 = 12*PCC*kappa*bCC/(abs(eta)*(3*PCC-eden(PCC)))
# UBQ2 = 4*kappa*bCC/(3*abs(eta))
# print(Qinf**2/LBQ2,Qinf**2/UBQ2)


# define energy density as function of pressure

def dedP(P,params):
    dP = 0.00001
    x1 = P-2*dP
    x2 = P-dP
    x3 = P+dP
    x4 = P+2*dP
    dedP = eden(x1,params) - 8*eden(x2,params) + 8*eden(x3,params) -␣
 ↪eden(x4,params)
    dedP = dedP/(12*dP)
    return dedP

# choose eos
def eden(P0,params):
    lmdbar = 10. # dimensionless; (0,0.1,5,10,20,50,100)
    Bag = 145.**4/(HC**3) # satuan MeV/fm^3
    lambda2 = 4*Bag*lmdbar
    return 3*P0 + 4*Bag + (4*lambda2/(np.pi)**2)*(1-np.sqrt(1+(np.pi**2/
 ↪lambda2)*(P0 + Bag)))
def sig(P0,params,b,f,r):
    Y=params[3]
    m=(1-f)*r/(2*GS*MSS)
    return Y*GS*MSS*m/r
```

[ ]:
```
# single_momin(PCC,Qinf,eta,Y,b0,radmass,profil)
# Q and Qinf in the code is rescaled as Q -> Q*UBQ   # not more than .29 is␣
 ↪save up to PCC=1200

output=single_momin(625,.04,-1,0.,1.,0,0)
output=single_momin(625,.04,-1,-1.,1.,0,0)
```

```
625 16655.000000001 4.1992129330229675 0.37265651068703454 0.0302609886558849
0.04 0.22994206908919937 0.6274565947284242 14.536443831098936
0.0017045839227741472
625 16697.000000001 4.221537210370674 0.37369529284140085 0.029901015134583108
0.04 0.23132204601365897 0.6301097240991574 14.749626155074528
0.0016742603331368533
```

[ ]:
```
# # if we want to print the profile, run this
# if os.path.exists('profileST.dat'):
#     os.remove('profileST.dat')

# if os.path.exists('profileSTmomin.dat'):
#     os.remove('profileSTmomin.dat')

# if os.path.exists('radmassST.dat'):
#     os.remove('radmassST.dat')

# if os.path.exists('radmassSTmomin.dat'):
#     os.remove('radmassSTmomin.dat')


# Qinf = .0 # in the code Q=Q*UBQ   # not more than .29 is save up to PCC=1200
# eta = -1
# Y = 0.

# output=single_PCC(55,Qinf,eta,Y,1.,1,1)
# output=single_momin(55,Qinf,eta,Y,1.,1,1)

# Qinf = None
# eta = None
# Y = None
```

[ ]:
```
if os.path.exists('radmassSTmomin.dat'):
    os.remove('radmassSTmomin.dat')


Qinf = .0
eta = -1
Y = 0.
for x in range(1, 5, 1):
```

```
    single_momin(x,Qinf,eta,Y,1.,1,0)
for x in range(5, 801, 5):
    single_momin(x,Qinf,eta,Y,1.,1,0)
Qinf = None
eta = None
Y = None
```

```
1 4837.000000001 0.0517991359367 0.015828212814146037 0.9526602074223 0.0 0.0
0.4050773414131416 0.009764049129857271 0.3528206653043836
2 6696.000000001 0.13798221482455417 0.030457420526242007 0.9091724747219446 0.0
0.0 0.40993173292594837 0.050440903272420436 0.3360332592575212
3 8032.000000001 0.23914672989781607 0.044007449550709565 0.8691316929533837 0.0
0.0 0.4145711226473245 0.1272120375701418 0.32052231298487677
4 9089.000000001 0.3480038810775584 0.05659179077765596 0.8321596579527455 0.0
0.0 0.4190104346417099 0.23958435090139998 0.30615456319759093
5 9964.000000001 0.46047035181378226 0.06830514327768916 0.7979406366468728 0.0
0.0 0.4232601706660856 0.38485207102625363 0.2928169254361523
10 12871.000000001 1.014362217835327 0.11648398698964069 0.6593310406128691 0.0
0.0 0.4420804708932215 1.4775299410222633 0.23842366623190492
15 14549.000000001 1.4973497661298736 0.15211623513341332 0.5592933565133432 0.0
0.0 0.45757170183189305 2.8844699558825124 0.19881776138549964
20 15633.000000001 1.897772284556115 0.17942676409098884 0.48424137619150265 0.0
0.0 0.4704960074884071 4.340122753127254 0.16894595200531765
25 16373.000000001 2.226030949405743 0.20095017525084524 0.4261838670799553 0.0
0.0 0.48139735621373625 5.713585591174702 0.14577747481196343
30 16895.000000001 2.495357572646877 0.21830317065882443 0.38012326966792026 0.0
0.0 0.4906865732776657 6.951374210565687 0.1273887110807706
35 17270.000000001 2.7172006931336052 0.2325491904734932 0.34282788581489676 0.0
0.0 0.49865919492489846 8.037621265654058 0.11251613181823464
40 17542.000000001 2.900950249525443 0.24442556461251375 0.312097505307006 0.0
0.0 0.5055488254476512 8.975918793981418 0.10029161162787731
45 17740.000000001 3.0541380880303057 0.25446057634085895 0.2863838040456681 0.0
0.0 0.5115456901231972 9.779072029490585 0.0901002937921698
50 17882.000000001 3.1821228485829627 0.2630185046310355 0.2646281811420407 0.0
0.0 0.5167647024179012 10.4582562671867 0.08151154365257178
55 17984.000000001 3.290243031599939 0.27041273269673727 0.2459491516724536 0.0
0.0 0.5213720813620448 11.03483013916279 0.07418002900602276
60 18053.000000001 3.3812095749187656 0.2768268180960902 0.2298217067783299 0.0
0.0 0.5254031220658077 11.515447468816234 0.06788095612912022
65 18098.000000001 3.458476881205823 0.28244880778566517 0.21573189289842226 0.0
0.0 0.5289730577447394 11.917822480770111 0.06241295162232771
70 18124.000000001 3.5242226105939576 0.28740527349401224 0.20333319140723222
0.0 0.0 0.5321374525934197 12.252157794067722 0.05763289974455043
75 18135.000000001 3.580276009834747 0.29179940462563175 0.19234726366711716 0.0
0.0 0.5349474053515262 12.52795049654112 0.053426544488397586
80 18134.000000001 3.628101535904104 0.29571358353775035 0.1825543778001732 0.0
0.0 0.5374442787411938 12.753148433709713 0.04970324577499116
```

85 18123.000000001 3.6687961100652497 0.29921195334824624 0.17378480679093822
0.0 0.0 0.5396563605505789 12.93356865811561 0.04639162174954838
90 18104.000000001 3.7034071533391137 0.30235167092372717 0.16588992245149448
0.0 0.0 0.5416173624494346 13.07556415331012 0.0434310384729842
95 18080.000000001 3.7332424571058773 0.3051920546567656 0.15871881043348848 0.0
0.0 0.543384701570447 13.188875560815513 0.04076522620824906
100 18050.000000001 3.758318209126794 0.3077526454305781 0.15221928099222293 0.0
0.0 0.5449366945301094 13.271234667902373 0.03836422876437383
105 18017.000000001 3.7798544496243562 0.3100830671635871 0.14626754423605043
0.0 0.0 0.5463344324826275 13.332632900219588 0.03618492107781955
110 17980.000000001 3.797801715587545 0.3121965157509684 0.14082966871362443 0.0
0.0 0.5475598059273663 13.370896803917308 0.034206714078548535
115 17941.000000001 3.813000603798953 0.3141272994781327 0.13582091230847315 0.0
0.0 0.5486550967534153 13.392970101415058 0.03239990800080164
120 17901.000000001 3.8259528395708924 0.31589865203714573 0.1311839100985052
0.0 0.0 0.5496425556493332 13.402686608926864 0.030742654225916724
125 17858.000000001 3.8362393026373156 0.31751067019103385 0.1269191159513603
0.0 0.0 0.5504874454228966 13.394794691776871 0.02922655899372166
130 17815.000000001 3.8449183118076173 0.31899710633107103 0.122943572080905 0.0
0.0 0.551250714214277 13.379049198895652 0.02782706755241251
135 17771.000000001 3.8518338346432874 0.3203620984396633 0.1192480981301308 0.0
0.0 0.5519170493565804 13.35310925793539 0.026535953607199247
140 17726.000000001 3.857116037692369 0.3216158267832795 0.11580870683442678 0.0
0.0 0.5524904629248382 13.317610069618006 0.02534270685419598
145 17681.000000001 3.8611724733453676 0.32273346923657485 0.11258852817686328
0.0 0.0 0.5529930018751745 13.276078141984707 0.02423505785644809
150 17635.000000001 3.8638169615975206 0.3238370491849657 0.10958440419436288
0.0 0.0 0.5534103803819383 13.226108727218126 0.023207955566118356
155 17590.000000001 3.865718480532377 0.3248252915673421 0.10674965873049543 0.0
0.0 0.5537816401291084 13.174003380910372 0.02224856093017034
160 17544.000000001 3.8663871848359954 0.32573331330977573 0.10409923292219189
0.0 0.0 0.5540749573547711 13.114399720722446 0.021356209893853347
165 17499.000000001 3.8664684548576704 0.326577825524108 0.10159094245607753 0.0
0.0 0.5543282972907664 13.0534496238694 0.020519932447235552
170 17454.000000001 3.8657457531784334 0.3273586105570723 0.09922739358722563
0.0 0.0 0.5545273678273401 12.988635535948864 0.01973745269877394
175 17410.000000001 3.864560634155886 0.3280853267263556 0.09698435091495888 0.0
0.0 0.5546917763660828 12.923100153033158 0.019001912268214658
180 17365.000000001 3.8624119635963106 0.32875264743124977 0.09487888746286438
0.0 0.0 0.5547911587489623 12.851535354251677 0.01831406102381592
185 17322.000000001 3.8601829368188265 0.3293785439024377 0.09286281005290624
0.0 0.0 0.5548773063906252 12.782571579315574 0.01766341625731369
190 17278.000000001 3.857091394514545 0.3299528719770495 0.09096704921510998 0.0
0.0 0.5549039657801865 12.708140642543441 0.01705349696085905
195 17235.000000001 3.853730389830406 0.33048784630453304 0.08915877637961046
0.0 0.0 0.5549057058665343 12.633986505894203 0.016476990518108265
200 17193.000000001 3.8501359272011273 0.3309861734717736 0.08743184989468215
0.0 0.0 0.5548843513663299 12.560275944612535 0.015931391369833195

205 17151.000000001 3.8460719694067103 0.33144648116620823 0.08579244843779148
0.0 0.0 0.5548263714571284 12.484487269171662 0.01541634396089474
210 17110.000000001 3.841840514630081 0.33187518115968584 0.08422318441137076
0.0 0.0 0.5547490488306313 12.409469969604924 0.014927737832039708
215 17069.000000001 3.837202993999885 0.3322707786187692 0.08273072319929434 0.0
0.0 0.5546390503752106 12.332714846589006 0.014465497134262012
220 17029.000000001 3.832453458044616 0.33263902331283607 0.08129897623358504
0.0 0.0 0.554513045693008 12.257002178544083 0.014025997998896965
225 16990.000000001 3.8276149326946154 0.3329816601578742 0.07992399531347877
0.0 0.0 0.5543723643025085 12.182428896294244 0.013607682736551812
230 16951.000000001 3.8224465927412212 0.33329711555769087 0.07861291485990955
0.0 0.0 0.5542039788453401 12.106511760871468 0.01321076847102203
235 16912.000000001 3.816971765907835 0.3335872402112504 0.07736183196324281 0.0
0.0 0.5540096135353177 12.029387177297169 0.012833804340142868
240 16875.000000001 3.811730845629525 0.33385962250351486 0.07614631067495567
0.0 0.0 0.5538187141231614 11.95624295453564 0.012472360664465729
245 16837.000000001 3.805961899908093 0.3341066942825368 0.07499476371253813 0.0
0.0 0.5535905612021494 11.879546226993115 0.012129994108215508
250 16801.000000001 3.800458118127271 0.33433840936677583 0.07387332930215805
0.0 0.0 0.5533678188312875 11.806941964522949 0.011801052119881989
255 16765.000000001 3.794718133778224 0.3345502954759258 0.07279991613134602 0.0
0.0 0.5531240728315086 11.733471240006189 0.011487581557223594
260 16729.000000001 3.7887581447046177 0.33474365570587095 0.07177181234927668
0.0 0.0 0.5528606474639864 11.65922915697051 0.011188612423070081
265 16694.000000001 3.7828466812576766 0.3349220828830153 0.07077687541971082
0.0 0.0 0.5525918448279046 11.58674220090611 0.010901901009067265
270 16660.000000001 3.776994426710176 0.3350863985575213 0.06981318774571325 0.0
0.0 0.5523183613471436 11.516039192964053 0.010626725557869846
275 16626.000000001 3.77095995008644 0.3352351868895435 0.06888830484288344 0.0
0.0 0.5520281137330633 11.444741380979188 0.01036369636627123
280 16593.000000001 3.7650057599209528 0.33537152418964794 0.06799095571091689
0.0 0.0 0.5517347496092079 11.37530693835692 0.010110849446758554
285 16560.000000001 3.75889100215763 0.33549407364741674 0.06712867789060531 0.0
0.0 0.551426398000108 11.305382620441835 0.009868827900115304
290 16528.000000001 3.7528746018508374 0.3356056022708598 0.06629070787258344
0.0 0.0 0.5511162987220352 11.237384254273664 0.009635825264959872
295 16496.000000001 3.7467164255373593 0.3357048605001114 0.06548454304664568
0.0 0.0 0.5507927923863415 11.168984379760397 0.009412506312555753
300 16464.000000001 3.7404259004676605 0.3357926231087301 0.0647085847591077 0.0
0.0 0.5504567459509636 11.100237826643104 0.0091983301865689
305 16434.000000001 3.734502396141997 0.33587286024932395 0.06394414377370786
0.0 0.0 0.5501326762044352 11.035806171744866 0.008990591022009903
310 16403.000000001 3.7282155411529647 0.33594113137270853 0.06321593443749997
0.0 0.0 0.5497853987627764 10.968774263700587 0.008792199976356637
315 16374.000000001 3.7223062055850282 0.3360026968570321 0.06249716796940131
0.0 0.0 0.5494507961359186 10.906057682818043 0.008599487807071352
320 16344.000000001 3.716048686063351 0.33605355542260296 0.06181203611309067
0.0 0.0 0.5490944388197114 10.840828217656783 0.008415265524857373

325 16315.000000001 3.709935871009042 0.33609710940522153 0.061142706226546516
0.0 0.0 0.5487400227742704 10.777660789070286 0.008237067646418128
330 16287.000000001 3.7039716395009425 0.3361336637882076 0.060488363892435476
0.0 0.0 0.5483877997419526 10.716548768021037 0.008064598892172252
335 16259.000000001 3.6979195910114315 0.3361623610000136 0.059856214129836986
0.0 0.0 0.54802693461805 10.655267186759854 0.007898541435667974
340 16231.000000001 3.6917854946716284 0.33618368572979973 0.05924525350318576
0.0 0.0 0.5476580051504569 10.593848787145598 0.007738572893220624
345 16204.000000001 3.6858131237002456 0.3361990875599089 0.0586467513874043 0.0
0.0 0.5472923854135575 10.53451393022915 0.007583472993434771
350 16177.000000001 3.6797676338797416 0.3362078601817513 0.058067777752685415
0.0 0.0 0.5469195317512595 10.47507435461626 0.007433914886764409
355 16151.000000001 3.6738906507294526 0.3362112657220727 0.05749987411408029
0.0 0.0 0.546550537565144 10.417720485827246 0.007288750504147804
360 16125.000000001 3.6679484040574026 0.33620870010924997 0.05695001408064296
0.0 0.0 0.5461750507322362 10.360288211922851 0.007148643515533529
365 16100.000000001 3.6621805026490195 0.3362012493592399 0.05640999113928284
0.0 0.0 0.54580389859367 10.30493950392889 0.007012511093746427
370 16075.000000001 3.6563542598711782 0.3361884111585698 0.05588668038057245
0.0 0.0 0.5454269160861717 10.249533788714734 0.006881005868001038
375 16050.000000001 3.6504735448348815 0.3361705166369812 0.0553793880765899 0.0
0.0 0.5450445128828441 10.194092398251634 0.006753914059162782
380 16026.000000001 3.6447751427446424 0.3361484050944775 0.05488025484841148
0.0 0.0 0.5446671394943644 10.140736703900476 0.006630240302094638
385 16002.000000001 3.6390279692638363 0.3361217224172266 0.054396009821766046
0.0 0.0 0.5442848992232424 10.087360130578285 0.006510619450727 3275
390 15978.000000001 3.633235300371419 0.3360907510497173 0.05392605630943412 0.0
0.0 0.5438981456711481 10.0339804807911 0.006394870629416996
395 15955.000000001 3.627631339720483 0.33605610506310096 0.05346284577616873
0.0 0.0 0.5435169811152257 9.98268102683617 0.006282074402878306
400 15932.000000001 3.6219865871632764 0.33601757471277954 0.0530129620391029
0.0 0.0 0.5431317663381675 9.931388071264594 0.00617284624471846
405 15909.000000001 3.6163038293612018 0.33597540204847065 0.05257588885863047
0.0 0.0 0.5427428073692172 9.88011652383868 0.00606703217250857
410 15887.000000001 3.6108148959534225 0.3359299944468697 0.05214435811939127
0.0 0.0 0.5423598840782917 9.830913857690161 0.005963779189284218
415 15865.000000001 3.6052918427764182 0.3358812821977506 0.051724810318569944
0.0 0.0 0.5419736036685713 9.781738059400299 0.005863682537990838
420 15843.000000001 3.5997370509343023 0.3358294736741761 0.051316796330190734
0.0 0.0 0.5415842317197553 9.732601661033467 0.005766610367566491
425 15822.000000001 3.5943801551521317 0.33577478565101326 0.05091330277293493
0.0 0.0 0.5412012491153358 9.685518464223524 0.005671768237695683
430 15801.000000001 3.5889947325137235 0.3357172838400997 0.05052062966052657
0.0 0.0 0.5408154983823243 9.6384769005722 0.005579730838 8167025
435 15780.000000001 3.5835828265059755 0.33565714841137445 0.05013838419168038
0.0 0.0 0.5404272106285584 9.591487542315976 0.005490384650232308
440 15760.000000001 3.5783720323951127 0.3355944195958664 0.04975978589282812
0.0 0.0 0.5400455890845615 9.546532498465224 0.005402987188720561

445 15740.000000001 3.573137409830932 0.33552929379813523 0.04939099763776245
0.0 0.0 0.5396617005683862 9.501629382017141 0.0053180926375154865
450 15720.000000001 3.5678807200690272 0.33546192746747383 0.04903167416313942
0.0 0.0 0.5392757467411828 9.456787145401613 0.005235602750766366
455 15700.000000001 3.562603628403957 0.3353924688163552 0.04868148597181842 0.0
0.0 0.5388879188162385 9.412014184129303 0.005155424117483322
460 15681.000000001 3.5575312174592466 0.3353207414594149 0.04833393460548436
0.0 0.0 0.5385070783999877 9.369252872075128 0.00507687521835979
465 15662.000000001 3.552440444855121 0.335247107266774 0.04799500770320379 0.0
0.0 0.5381245729020305 9.326557760052644 0.0050004837100170296
470 15643.000000001 3.547332753366186 0.33517169561165616 0.04766441381936966
0.0 0.0 0.5377405697256227 9.283935987871834 0.004926168223373634
475 15625.000000001 3.542431488556546 0.3350941812010918 0.0473358462809554 0.0
0.0 0.5373636960502177 9.243302135279444 0.0048532873419113234
480 15606.000000001 3.5372934987768243 0.3350155350312709 0.04702114335579905
0.0 0.0 0.5369770979054522 9.20083790009709 0.004782904006984416
485 15588.000000001 3.532363494877812 0.3349349312971599 0.04670804148923716 0.0
0.0 0.5365977899210251 9.16035652005332 0.004713828578359802
490 15571.000000001 3.5276412159741284 0.3348523539756615 0.04639643925400795
0.0 0.0 0.5362257124947105 9.121838258862265 0.004646019705360068
495 15553.000000001 3.5226862649520982 0.3347690092158298 0.046097908631084664
0.0 0.0 0.5358443955915131 9.081515275108863 0.004580495570482752
500 15536.000000001 3.5179403162531564 0.3346838124474568 0.04580050436353426
0.0 0.0 0.5354704423666716 9.043149492650928 0.004516127645463984
505 15519.000000001 3.5131838790756507 0.3345974300965875 0.04550988649380641
0.0 0.0 0.5350957063134247 9.004863388049598 0.004453393460385643
510 15502.000000001 3.5084179386659895 0.3345099522010513 0.045225842647105476
0.0 0.0 0.5347203039855458 8.966661561950108 0.0043922358828356175
515 15486.000000001 3.5038616934522 0.3344207011323467 0.04494251424933109 0.0
0.0 0.5343523100357838 8.930390938979775 0.0043321030904717745
520 15469.000000001 3.499079074399881 0.3343312487832425 0.0446710575733672 0.0
0.0 0.5339758438464886 8.892362563134851 0.004273944656478222
525 15453.000000001 3.4945070325792016 0.33424011118652985 0.04440001648445185
0.0 0.0 0.5336068780962135 8.856258039972888 0.004216723288256474
530 15437.000000001 3.489928124597002 0.33414812725650744 0.04413486266243736
0.0 0.0 0.5332375464816835 8.820238475478773 0.004160887799155935
535 15421.000000001 3.4853431293003023 0.33405536900803195 0.04387542013974276
0.0 0.0 0.5328679417653444 8.78430735367383 0.004106391855146754
540 15406.000000001 3.480968963645618 0.3339609674544999 0.04361607121433834 0.0
0.0 0.5325058410352321 8.75027352779477 0.004052729381522865
545 15391.000000001 3.4765893561506607 0.3338658593003182 0.0433621799020855 0.0
0.0 0.5321435354119685 8.716320242867736 0.004000332763092594
550 15376.000000001 3.47220499522752 0.33377010865200707 0.043113587159942485
0.0 0.0 0.5317811071576735 8.682450519580248 0.003949161607421576315
555 15361.000000001 3.467816534107507 0.33367377648138724 0.04287014001432958
0.0 0.0 0.5314186343251771 8.648667186697683 0.0038991737577878695
560 15346.000000001 3.4634245925891443 0.3335769207786161 0.04263169128104062
0.0 0.0 0.5310561909598057 8.614972891092338 0.003850333862355575

565 15331.000000001 3.459029758693284 0.3334795966971685 0.042398099300174415
0.0 0.0 0.5306938472910964 8.58137010723992 0.0038026046428148485
570 15317.000000001 3.4548464666631173 0.33338072954692516 0.04216400451906365
0.0 0.0 0.5303390651431522 8.549625869251539 0.003755527019954465
575 15303.000000001 3.4506606483276747 0.3332814374960649 0.04193456878136478
0.0 0.0 0.5299844213584184 8.517964514286827 0.0037095030841633172
580 15289.000000001 3.4464728269669105 0.33318176996453464 0.04170966316525776
0.0 0.0 0.5296299784656442 8.486388202734247 0.003664500185700886
585 15275.000000001 3.4422834992638975 0.33308177397989713 0.04148916343403142
0.0 0.0 0.5292757957581531 8.454898952952163 0.0036204869535493315
590 15261.000000001 3.4380931365882184 0.3329814942906542 0.04127294982976476
0.0 0.0 0.5289219294462065 8.4234986485589 0.0035774332354214754
595 15248.000000001 3.434114296664821 0.3328797028958812 0.04105585534509998 0.0
0.0 0.5285756102976407 8.39392234290982 0.003534913636940103
600 15235.000000001 3.430134622066386 0.33277765680749016 0.040842885028454505
0.0 0.0 0.5282296295179025 8.364426228266941 0.003493307045554688
605 15221.000000001 3.425943098836556 0.332676721171165 0.04063891544301143 0.0
0.0 0.5278769404266099 8.33329052989306 0.0034529723731943534
610 15209.000000001 3.422174435530845 0.3325729599089741 0.04042888243146418 0.0
0.0 0.5275388813210632 8.305681086036849 0.0034127251866261053
615 15196.000000001 3.4181947045810537 0.33247038402684165 0.040227642327540836
0.0 0.0 0.5271942065800145 8.27643504171386 0.003373698583826696
620 15183.000000001 3.414215702890889 0.33236770348160827 0.04003011042773271
0.0 0.0 0.5268500557980224 8.247275165925025 0.003335482227529464
625 15171.000000001 3.4104479261539082 0.3322635242668495 0.03983132916969461
0.0 0.0 0.5265134149205137 8.219901157830357 0.0032976859952400486
630 15158.000000001 3.406471089536847 0.33216070807500886 0.039640960341862784
0.0 0.0 0.526170395020781 8.190911559389484 0.0032610254765467373
635 15146.000000001 3.402705506388255 0.3320564069654032 0.039449219324861826
0.0 0.0 0.525834889069734 8.163698966459755 0.0032247498587254543
640 15134.000000001 3.39894133257866 0.3319520780924341 0.039260880767490965 0.0
0.0 0.5254999877870841 8.136566025876665 0.003189204714937441
645 15122.000000001 3.3951788614795033 0.33184775006438816 0.039075859864178865
0.0 0.0 0.5251657254657256 8.10951376144517 0.0031543696061809524
650 15110.000000001 3.391418371451779 0.33174345011213274 0.03889407459955975
0.0 0.0 0.5248321345276754 8.082543120901292 0.0031202248185176634
655 15099.000000001 3.387868488526689 0.331637636340846 0.03871075550417657 0.0
0.0 0.52450598760539 8.057321237402643 0.003086410770312694
660 15087.000000001 3.3841124596383105 0.33153344730975853 0.038535232783202936
0.0 0.0 0.5241737980277645 8.030511170200208 0.003053594277311445
665 15076.000000001 3.380566967690847 0.3314277493703919 0.038358078634579525
0.0 0.0 0.5238490472261255 8.005441244489388 0.0030210804309489334
670 15065.000000001 3.3770238908390926 0.33132213402295074 0.03818390968933295
0.0 0.0 0.5235250207882496 7.980445989945215 0.002989192435966465
675 15054.000000001 3.3734834569376924 0.3312166238197962 0.03801265477899077
0.0 0.0 0.5232017447529179 7.955526132990359 0.002957913529550154
680 15043.000000001 3.3699458820939863 0.33111124022503213 0.037844244977963425
0.0 0.0 0.5228792436833415 7.930682342071844 0.0029272275187301054

685 15032.000000001 3.3664113711911003 0.33100600366195676 0.037678613517406284
0.0 0.0 0.5225575407335238 7.905915230503298 0.002897118757086668
690 15021.000000001 3.362880118386635 0.33090093355832195 0.03751569570295512
0.0 0.0 0.5222366577115979 7.881225359170755 0.0028675721225627743
695 15010.000000001 3.3593523075884937 0.33079604838954 0.037355428836127504 0.0
0.0 0.5219166151403155 7.856613239110075 0.0028385729963224617
700 15000.000000001 3.3560340560820054 0.330689612411065 0.03719329387928458 0.0
0.0 0.5216039159783804 7.833701799932539 0.0028098007034961744
705 14990.000000001 3.3527190860705565 0.3305833579838344 0.03703373795515916
0.0 0.0 0.5212920434414757 7.810859819698227 0.002781554979030909
710 14979.000000001 3.3492021274256816 0.3304790933583419 0.03688111243118643
0.0 0.0 0.5209745825308315 7.786474222695201 0.0027541220822675967
715 14969.000000001 3.345894464116066 0.3303732706931404 0.03672651733929599 0.0
0.0 0.5206644420513769 7.763776941053133 0.0027268863167208895
720 14959.000000001 3.3425905589431397 0.33026767787041844 0.03657433146356427
0.0 0.0 0.5203551800285323 7.741150371694655 0.002700138184433436
725 14949.000000001 3.3392905555314063 0.33016232955187097 0.03642450138862796
0.0 0.0 0.5200468118058266 7.718594863275286 0.0026738655097160286
730 14940.000000001 3.3361990727724775 0.33005537731075096 0.03627265702963948
0.0 0.0 0.5197456911109541 7.697706793578597 0.002647769730902529
735 14930.000000001 3.3329070442494557 0.32995054222468345 0.03612740641514159
0.0 0.0 0.5194391302872101 7.675290011484692 0.0026224160514418423
740 14921.000000001 3.329823327735685 0.3298440953533813 0.03598008631308754 0.0
0.0 0.5191397998118294 7.654532872679499 0.0025972228724396845
745 14911.000000001 3.3265397752796293 0.3297398251696422 0.035839218078728495
0.0 0.0 0.5188350983579066 7.63225590746548 0.0025727437059170025
750 14902.000000001 3.323464314356787 0.32963393413842357 0.035696229270656 0.0
0.0 0.5185376084072849 7.611630772973819 0.0025484097984011067
755 14893.000000001 3.320393037192252 0.3295283303369847 0.03555534784962016 0.0
0.0 0.5182410455472414 7.591069983149344 0.0025244915558081476
760 14883.000000001 3.317122916394427 0.3294249857246521 0.035420717898328694
0.0 0.0 0.5179392057515189 7.5690021666603515 0.0025012480487692844
765 14874.000000001 3.31406053547639 0.329320004619057 0.03528389808843842 0.0
0.0 0.5176445470868488 7.548574572144126 0.0024781289037681996
770 14866.000000001 3.311205346020497 0.3292133507075501 0.03514490817352319 0.0
0.0 0.5173570214701133 7.529775606325548 0.002455133166459655
775 14857.000000001 3.3081518127413228 0.3291090014756898 0.03501202266342675
0.0 0.0 0.5170642653005988 7.509473989220755 0.002432781393963698
780 14848.000000001 3.305102939574197 0.3290049896316384 0.03488103367478518 0.0
0.0 0.5167724818656643 7.48923748759787 0.00241079861615635
785 14839.000000001 3.302058805765121 0.3289013236825753 0.03475190246594138 0.0
0.0 0.5164816782894732 7.469066184121222 0.002389176318022862
790 14831.000000001 3.2992213774640895 0.3287959622879947 0.034620522272859805
0.0 0.0 0.5161979679760772 7.4505088165086075 0.002367653290705973
795 14822.000000001 3.2961867473477193 0.3286929982176916 0.034495013199563454
0.0 0.0 0.5159091263491474 7.430464305364196 0.002346729901369722
800 14814.000000001 3.293358568582993 0.32858832626648143 0.03436722037184989
0.0 0.0 0.5156273572854129 7.412026520230031 0.0023258949264712207

[ ]: