```
# Program for calculating a proton, neutron by shells taking into account quarks

import pandas as pd
import numpy as np
from numpy import *
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.pyplot import *
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.interpolate import PchipInterpolator
from scipy.signal import savgol_filter
from prettytable import PrettyTable

# Error elimination, since it does not affect the values obtained
# The graph is rendered in 3D, the package for this type of charts uses the square root
# The presence of a pair of negative numbers excludes their visualization
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)

comments = [[1, 'Proton and neutron consist of a core \n and two shells around them \n',
             'Robert Hofstadter the Nobel laureate \n'],
            [2, 'The proton consists of two quarks \n "u" and a quark "d" \n',
             'Murray Gell-Mann the Nobel laureate, \n and George Zweig \n'],
            [3, 'The neutron consists of two quarks \n "d" and a quark "u" \n',
             'Murray Gell-Mann the Nobel laureate, \n and George Zweig \n'],
            [4, '"Conditional quark" consists of a core and \n two shells \n', 'The
assumption of the author \n'],
            [5, 'Quark radius \n  "− (0.47 · 10E−16 cm)2 < RE2 < (0.43 · 10E−16 cm)2" \n',
             'https://arxiv.org/pdf/1604.01280.pdf \n'],
            [6, 'Proton, a neutron can be represented \n as the sum of three matrices \n',
             'The mathematical derivation of the author \n'],
            [7, '{x1, x2, x3, 0, 0} + {0, y1, y2, y3, 0} \n + {0, 0, x1, x2, x3} \n',
             'View of three matrices for obtaining \n a proton, neutron \n'],
            [8, 'x1, y1 - quark cores \n', "Usually, quarks proper in today's a view \n"],
            [9, '{x1, x2+y1, x3+y2+x1, y3+x2, x3} \n',
             'A schematic view of the matrix \n for a proton, neutron \n'],
            [10, '{x1, x2+y1, x3+y2+x1} - quark core \n', 'x1, y1 - quark cores \n'],
            [11, '{y3+x2, x3} - quark shells \n', 'x1, y1 - absent \n'],
            [12, 'The proposed approach allows one to obtain many \n different particles,
including pseudo particles',
             'This program does not include their calculation']]

table1 = PrettyTable(['#', 'Description', 'Link to source/ comments'])

for rec in comments:
    table1.add_row(rec)



class Preliminary():
# volume of proton + neutron
# Quark condensate provides about 9 percent of the proton's mass
# Physical Review Letters, 2018, website arXiv.org

# V = 4/3πR**3
    π = 3.14159265358979

    Vy = 4/3 * π * (2.5E-16)**3
    Vs1 = 4/3 * π * (1.4E-15)**3
    Vs2 = 4/3 * π * (2.5E-15)**3
    Vs11 = Vs1 - Vy
    Vs21 = Vs2 -Vs11
```

```
# https://physics.nist.gov/cgi-bin/cuu/Value?mp - 1.67262192369E-27 kg.

    mps2 = 0.09 * 1.67262192369E-27/(Vs11/Vs21 +1)

# https://physics.nist.gov/cgi-bin/cuu/Value?mn - 1.67492749804E-27 kg.

    mns2 = 0.09 * 1.67492749804E-27/(Vs11/Vs21 +1)
    mps1 = 0.09 * 1.67262192369E-27 - mps2
    mns1 = 0.09 * 1.67492749804E-27 - mns2

class Proton():
# The magnitude of the charge of the core, shells in the proton, neutron, respectively
#Robert Hofstadter the Nobel laureate

    SHELLSP1 = 0.35
    SHELLSP2 = 0.5
    SHELLSP3 = 0.15
    SHELLSN1 = 0.35
    SHELLSN2 = -0.5
    SHELLSN3 = 0.15

# The mass of the core, shells in the proton, neutron, respectively

    shellsmp1 = 1.67262192369E-27 * 0.91
    shellsmp2 = Preliminary.mps1
    shellsmp3 = Preliminary.mps2

    shellsmn1 = 1.67492749804E-27 * 0.91
    shellsmn2 = Preliminary.mns1
    shellsmn3 = Preliminary.mns2

    def __init__ (self, array):
        self.array = array

# Array input according to the matrix proposed by the author

a1 = array ([[2.0 , 1.0, 1.0, 1.0, 1.0, 0.0], [0.0, 1.0, 0.0, 0.0, 0.0, 0.0],
            [0.0, 0.0, 1.0, 0.0, 0.0, 0.0], [1.0, 1.0, 0.0, 2.0, 1.0, 1.0],
            [0.0, 0.0, 0.0, 0.0, 1.0, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, 1.0]])


unit = Proton(a1)
unit.array

b1 = array ([Proton.SHELLSP1, Proton.SHELLSP2, Proton.SHELLSP3, Proton.SHELLSN1,
Proton.SHELLSN2, Proton.SHELLSN3])

# The calculation of electric charges of quark "u" and "d" for each shells in electron
charges

x1 = linalg.solve (unit.array, b1)

qvark = list(x1)

data1 = {'index': ['uq1', 'uq2', 'uq3', 'dq1', 'dq2', 'dq3'],
        'Qe': [ qvark[0], qvark[1],  qvark[2],  qvark[3],  qvark[4],  qvark[5]]}

uq1 = qvark[0]
uq2 = qvark[1]
uq3 = qvark[2]
```

```
dq1 = qvark[3]
dq2 = qvark[4]
dq3 = qvark[5]

shell = [[1, 'uq1', uq1], [2, 'uq2', uq2], [3, 'uq3', uq3], [4, 'dq1', dq1],
     [5, 'dq2', dq2], [6, 'dq3', dq3]]

table = PrettyTable(['#', 'Index', 'Charge in the Qe'])

for rec in shell:
    table.add_row(rec)

# Calculation of the amount of charge on the shells, charge of an electron is taken modulo

Qe = 1.602176620898e-19
uq11 = Qe * qvark[0]
uq21 = Qe * qvark[1]
uq31 = Qe * qvark[2]
dq11 = Qe * qvark[3]
dq21 = Qe * qvark[4]
dq31 = Qe * qvark[5]

# The calculation of mass of quark "u" and "d" for each shells

b2 = array ([Proton.shellsmp1, Proton.shellsmp2, Proton.shellsmp3, Proton.shellsmn1,
Proton.shellsmn2, Proton.shellsmn3])

x2 = linalg . solve (unit.array, b2)

qvarkm = list(x2)

data2 = {'index': ['um1', 'um2', 'um3', 'dm1', 'dm2', 'dm3'],
        'mass': [ qvarkm[0], qvarkm[1], qvarkm[2], qvarkm[3], qvarkm[4], qvarkm[5]]}

um1 = qvarkm[0]
um2 = qvarkm[1]
um3 = qvarkm[2]
dm1 = qvarkm[3]
dm2 = qvarkm[4]
dm3 = qvarkm[5]

# The calculation of volume of quark shells "u" and "d"

b3 = ([Preliminary.Vy, Preliminary.Vs11, Preliminary.Vs21, Preliminary.Vy,
Preliminary.Vs11, Preliminary.Vs21])

x3 = linalg . solve (unit.array, b3)

qvarkv = list(x3)

data3 = {'index': ['uv1', 'uv2', 'uv3', 'dv1', 'dv2', 'dv3'],
        'mass': [ qvarkv[0], qvarkv[1], qvarkv[2], qvarkv[3], qvarkv[4], qvarkv[5]]}

uv1 = qvarkv[0]
uv2 = qvarkv[1]
uv3 = qvarkv[2]
dv1 = qvarkv[3]
dv2 = qvarkv[4]
dv3 = qvarkv[5]

# Data entry for quarks "u" and "d"
```

```
data = {'Index "u"': ['uq11', 'um1', 'uv1', 'uq21', 'um2', 'uv2', 'uq31', 'um3', 'uv3'],
        'Value "u"': [ uq11,   um1,   uv1,   uq21,   um2,   uv2,   uq31,   um3,   uv3],
       'Index "d"': ['dq11', 'dm1', 'dv1', 'dq21', 'dm2', 'dv2','dq31', 'dm3', 'dv3'],
       'Value "d"': [ dq11,   dm1,   dv1,   dq21,   dm2,   dv2,  dq31,   dm3,   dv3]}


quarku = [[1, 'uq11', uq11, 'um1', um1, 'uv1', uv1],
         [2, 'uq21', uq21, 'um2', um2, 'uv2', uv2],
         [3, 'uq31', uq31, 'um3', um3, 'uv3', uv3]]

table2 = PrettyTable(['#', 'Charge symbol', 'Charge in Cl', 'Mass symbol',
                      'Mass in kg.', 'Volume symbol', 'Volume in cubic meters'])

for rec in quarku:
    table2.add_row(rec)

quarkd = [[1, 'dq11', dq11, 'dm1', dm1, 'dv1', dv1],
         [2, 'dq21', dq21, 'dm2', dm2, 'dv2', dv2],
         [3, 'dq31', dq31, 'dm3', dm3, 'dv3', dv3]]

table3 = PrettyTable(['#', 'Charge symbol', 'Charge in Cl', 'Mass symbol',
                      'Mass in kg.', 'Volume symbol', 'Volume in cubic meters'])

for rec in quarkd:
    table3.add_row(rec)


# Description for proton, neutron, by shells
# The top line - the center, the bottom line - the upper shell
# The presented interactions in date4 are the author's approach


proton = [[1, 'pq1', uq11, 'pm1', um1, 'pv1', uv1],
         [2, 'pq2', uq21, 'pm2', um2, 'pv2', uv2],
         [3, 'pq3', dq11, 'pm3', dm1, 'pv3', dv1],
         [4, 'pq4', uq31, 'pm4', um3, 'pv4', uv3],
         [5, 'pq5', uq11, 'pm5', um1, 'pv5', uv1],
         [6, 'pq6', dq21, 'pm6', dm2, 'pv6', dv2],
         [7, 'pq7', dq31, 'pm7', dm3, 'pv7', dv3],
         [8, 'pq8', uq21, 'pm8', um2, 'pv8', uv2],
         [9, 'pq9', uq31, 'pm9', um3, 'pv9', uv3]]

table4 = PrettyTable(['#', 'Charge symbol', 'Charge in Cl', 'Mass symbol',
                      'Mass in kg.', 'Volume symbol', 'Volume in cubic meters'])

for rec in proton:
    table4.add_row(rec)


neutron = [[1, 'dq1', dq11, 'dm1', dm1, 'dv1', dv1],
          [2, 'dq2', dq21, 'dm2', dm2, 'dv2', dv2],
          [3, 'dq3', uq11, 'dm3', um1, 'dv3', uv1],
          [4, 'dq4', dq31, 'dm4', dm3, 'dv4', dv3],
          [5, 'dq5', dq11, 'dm5', dm1, 'dv5', dv1],
          [6, 'dq6', uq21, 'dm6', um2, 'dv6', uv2],
          [7, 'dq7', uq31, 'dm7', um3, 'dv7', uv3],
          [8, 'dq8', dq21, 'dm8', dm2, 'dv8', dv2],
          [9, 'dq9', dq31, 'dm9', dm3, 'dv9', dv3]]

table5 = PrettyTable(['#', 'Charge symbol', 'Charge in Cl', 'Mass symbol',
                      'Mass in kg.', 'Volume symbol', 'Volume in cubic meters'])

for rec in neutron:
    table5.add_row(rec)


# Visualization

# Distribution by shells in proton and neutron, 3D graph

n = 9

Xp = np.array([uq11, uq21, dq11, uq31, uq11, dq21, dq31, uq21, uq31])
Yp = np.array([um1,  um2,  dm1,  um3,  um1,  um2,  dm2,   dm3, um3])
Zp = np.array([uv1,  uv2,  dv1,  uv3,  uv1,  uv2,  dv2,   dv3, uv3])

Xn = np.array([dq11, dq21, uq11, dq31, dq11, uq21, uq31, dq21, dq31])
Yn = np.array([dm1,  dm2,  um1,  dm3,  dm1,  um2,   um3,  dm2,  dm3])
Zn = np.array([dv1,  dv2,  uv1,  dv3,  dv1,  uv2,   uv3,  dv2,  dv3])

fig = plt.figure(figsize=(14.,14.))
ax = fig.add_subplot(111, projection='3d')

plt.scatter(Xp,Yp,Zp)
plt.scatter(Xn,Yn,Zn)

xs = (n, -10e-19, 10e-19)
ys = (n, -10e-28,10e-28)
zs = (n, -10e-44,10e-44)

scat = ax.scatter(Xp, Yp, Zp, c=Zp.flatten(), alpha=1, s = 1600, marker='^')
fig.colorbar(scat, shrink=0.5, aspect=5)

scat = ax.scatter(Xn, Yn, Zn, c=Zn.flatten(), alpha=0.7, s = 1000, marker='D')

plt.tick_params(axis='both', which='major', labelsize=16)

ax.set_xlabel('\n \n The quantity charge shell \n in Cl x E-20', fontsize = 15)
ax.set_ylabel('\n \n Mass in \n kg. x E-28', fontsize = 15)
ax.set_zlabel('\n Shell volume in\n cbm*E-44', fontsize = 15)

ax.text2D(0.2, 0.95,
          "DISTRIBUTION BY SHELLS IN PROTON AND NEUTRON \n proton - triangle  neutron -
rhombus",
          transform=ax.transAxes, fontsize = 20)

# The cycle of charge distribution over shells in a free neutron and proton, 2D graph

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8])

# neutron free state
y22 = np.array([dq11, dq21, uq11, dq31, dq11, uq21, uq31, dq21, dq31])

# proton free state
z22 = np.array([uq11, uq21, dq11, uq31, uq11, dq21, dq31, uq21, uq31])

xx = np.linspace(x.min(),x.max(), 1000)
fig, axs = plt.subplots(1, 1, figsize=(14, 11))

itp1 = PchipInterpolator(x,y22)
itp2 = PchipInterpolator(x,z22)
```

```
window_size, poly_order = 57, 2

y22y22_sg = savgol_filter(itp1(xx), window_size, poly_order)
z22z22_sg = savgol_filter(itp2(xx), window_size, poly_order)

axs.plot(x, y22, 'gs', label= 'The charge distribution in a free neutron over shells')

axs.plot(xx, y22y22_sg, 'green', label= "Smoothed curve")

axs.plot(x, z22, 'bs', label= 'The charge distribution in a free proton over shells')
axs.plot(xx, z22z22_sg, 'b', label= "Smoothed curve")

# or fit to a global function
def func(x, A, B, x0, sigma):
    return abs(A)+B*np.tanh((x-x0)/sigma)

fit, _ = curve_fit(func, x, y22)
y22y22_fit = func(xx, *fit)

fit, _ = curve_fit(func, x, z22)
z22z22_fit = func(xx, *fit)

axs.plot(xx, y22y22_fit, 'g--', label=r"$f(xn) = |A| + B \tanh\left(\frac{x-x_0}
{\sigma}\right)$")

axs.plot(xx, z22z22_fit, 'b--', label=r"$f(xp) = |A| + B \tanh\left(\frac{x-x_0}
{\sigma}\right)$")

plt.ylabel('The amount of charge \n \n in Cl x E-20', fontsize=15)

plt.xlabel('Shell number', fontsize=15)

yticks(fontsize=12)

plt.title('THE CHARGE DISTRIBUTION OVER SHELLS IN A FREE NEUTRON AND PROTON \n',
fontsize=17)

print('\n Significant comments')
print(table1, "\n")

print("\n Values of quarks 'u' and 'd' by \n"
      "shells in Qe (electron charges) \n")
print(table)

print("   ", '\n Values of quarks "u" by shells \n')
print(table2)

print("   ", '\n Values of quarks "d" by shells \n')
print(table3)

print('\n Detailed description for proton by shells \n')
print(table4)

print('\n Detailed description for neutron, by shells \n')
print(table5)

plt.legend(loc='upper left', fontsize=12)

plt.show()

savefig('The charge distribution over shells in a free neutron and proton.pdf')
```