Κβαντική Πληροφορία Και Επεξεργασία

ΥΦΥ209

ΣΤΕΦΑΝΙΑ ΑΡΧΟΝΤΗ

ΑΕΜ: 4396

Κώδικας: Ορισμός Συναρτήσεων

```
def ode_1(t, y):
    eq = [y[1], -(2/t)*y[1]-y[0]**(1/(i-1))]
    return eq


def hak(theta, n_val, t, k_val):
    return 1/k_val * pow(theta, n_val) * np.sin(np.multiply(k_val, t)) * t


def sak(f, k_val2):
    return f * np.log(f) * np.multiply(k_val2, k_val2)


def mass(theta_1, n_val, t):
    return pow(theta_1, n_val) * pow(t, 2)
```

$$\frac{1}{\xi^2}\frac{d}{d\xi}\xi^2\frac{d\theta}{d\xi}+\theta^{1/(\gamma-1)}=0.$$

$$h(\kappa)=\left(\frac{4\pi\rho_0 a^3}{\kappa}\int_0^{\xi_R}\theta^{1/(\gamma-1)}(\xi)\sin(\kappa\xi)\xi d\xi\right)^2$$

$$S=-4\pi a^{-3}\int_{\kappa_{\min}}^{\infty}\tilde{f}(\kappa)\log(\tilde{f}(\kappa))\kappa^2 d\kappa$$

$$\mathrm{M}=4\pi\rho_0\alpha^3\int_0^{\xi_R}\theta^{1/(\gamma-1)}(\xi)\xi^2 d\xi$$

Αρχικοποίηση πινάκων

```
# Initialization of arrays and constants
gamma_1: np.ndarray = np.array([1.2, 1.4, 1.7])
y0: np.ndarray = np.array([1, 0])
xi: np.ndarray = np.array([pow(10, -8), 20])
k_min: np.ndarray = np.zeros(3)
value_0: np.ndarray = np.zeros(3)
gamma: np.ndarray = np.linspace(1.2, 1.9, 200)
k_min_1: np.ndarray = np.zeros(len(gamma))
k_min_2: np.ndarray = np.zeros(len(gamma))
k_min_3: np.ndarray = np.zeros(len(gamma))
M: np.ndarray = np.zeros(len(gamma))
S: np.ndarray = np.zeros(len(gamma))
S1: np.ndarray = np.zeros(len(gamma))
S2: np.ndarray = np.zeros(len(gamma))
S3: np.ndarray = np.zeros(len(gamma))
```

```python
for index, i in enumerate(tqdm(gamma_1)):
    n = 1 / (i - 1)

    # Solving Lane-Emden
    sol = solve_ivp(ode_1, xi, y0, method='RK45', atol=1e-25, rtol=1e-25, events=stop)
    k_min[index] = (np.pi / sol.t[-1])
    k: np.ndarray = np.linspace(0, 20, len(sol.t))
    value_1: list = []
#   plt.plot(sol.t, sol.y[0], label="g="+str(i))

    # Calculation of minimum h(ak)
    h1 = hak(sol.y[0], n, sol.t, k_min[index])
    I1 = simpson(h1, sol.t, dx=0.0001)
    value_0[index] = I1**2

    for j in k:
        # Calculating h(ak) for every k and saving result into value_1 array
        h = hak(sol.y[0], n, sol.t, j)
        I0 = (simpson(h, sol.t, dx=0.0001))**2
        value_1.append(I0)
    plt.figure(1)
    plt.plot(k / np.sqrt(i/(i-1)), value_1 / value_0[index], label="γ = " + str(i))
```

Για το Figure (1):

1) Επίλυση Lane-Emden με solve_ivp

2) Υπολογισμός h(ak) min

3) Υπολογισμός h(ak) για κάθε τιμή του k

4) Υπολογισμός f (k)

$$\tilde{f}(\kappa) = \frac{h(\kappa)}{h(\kappa_{\min})}$$

Κώδικας για το Figure(2)

1) Επίλυση Lane-Emden

2) Εύρεση h(ak) min

3) Υπολογισμός h(ak) και f(k)

4) Υπολογισμός της μάζας

5) Υπολογισμός της εντροπίας

```
for index_1, i in enumerate(tqdm(gamma)):
    n_1 = 1 / (i - 1)
    # Solving Lane-Emden
    sol_1 = solve_ivp(ode_1, xi, y0, method='RK45', atol=1e-25, rtol=1e-25, events=stop)
    k_min_1[index_1] = (np.pi / sol_1.t[-1])
```

```
h2 = hak(sol_1.y[0], n_1, sol_1.t, k_min_1[index_1])
I2 = (simpson(h2, sol_1.t, dx=0.0001))**2
```

```
 for index_2, val_1 in enumerate(k2):
     h5 = hak(sol_1.y[0], n_1, sol_1.t, val_1)
     I5 = (simpson(h5, sol_1.t, dx=0.0001)) ** 2
     value_2[index_2] = I5
 f_1 = value_2 / I2
```

```
# Calculation of mass
mass_1 = mass(sol_1.y[0], n_1, sol_1.t)
M[index_1] = (4*np.pi*((i/(i-1))**(3/2))*simpson(mass_1, sol_1.t, dx=0.0001))/200

# Calculation of Configurational entropy
s_1 = sak(f_1, k2)
S[index_1] = -4 * np.pi * (i/(i - 1)) ** (-3 / 2) * simpson(s_1, k2, dx=0.0001)
```

Κώδικας για Figure(3)

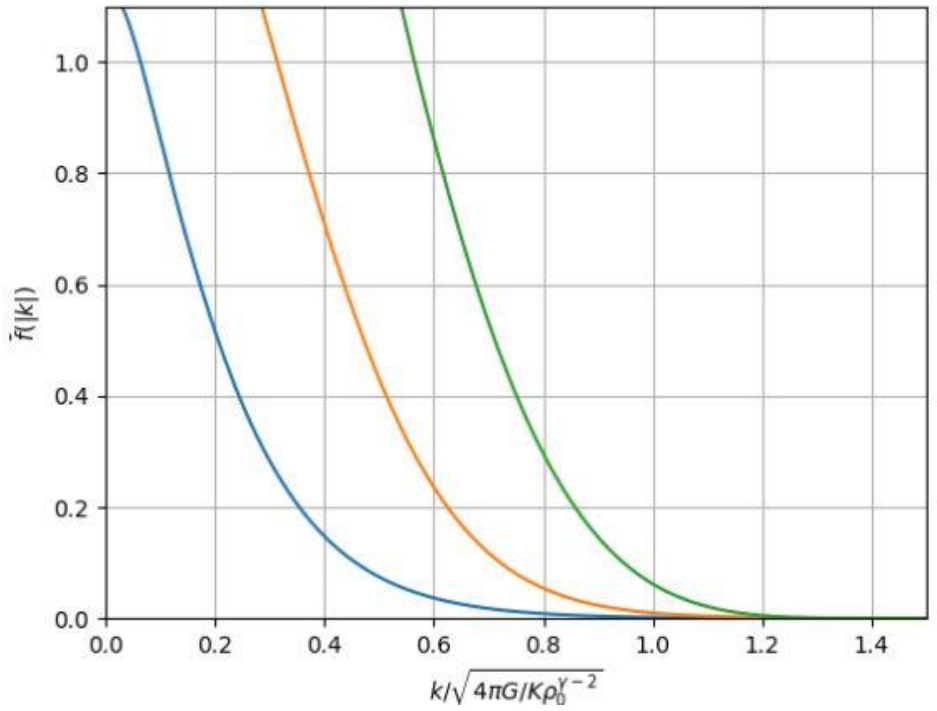
```
k_min_2[index_1] = (np.pi / (0.95*sol_1.t[-1]))
k_min_3[index_1] = (np.pi / (1.05*sol_1.t[-1]))
```

```
h3 = hak(sol_1.y[0], n_1, sol_1.t, k_min_2[index_1])
I3 = (simpson(h3, sol_1.t, dx=0.0001)) ** 2

h4 = hak(sol_1.y[0], n_1, sol_1.t, k_min_3[index_1])
I4 = (simpson(h4, sol_1.t, dx=0.0001)) ** 2
```

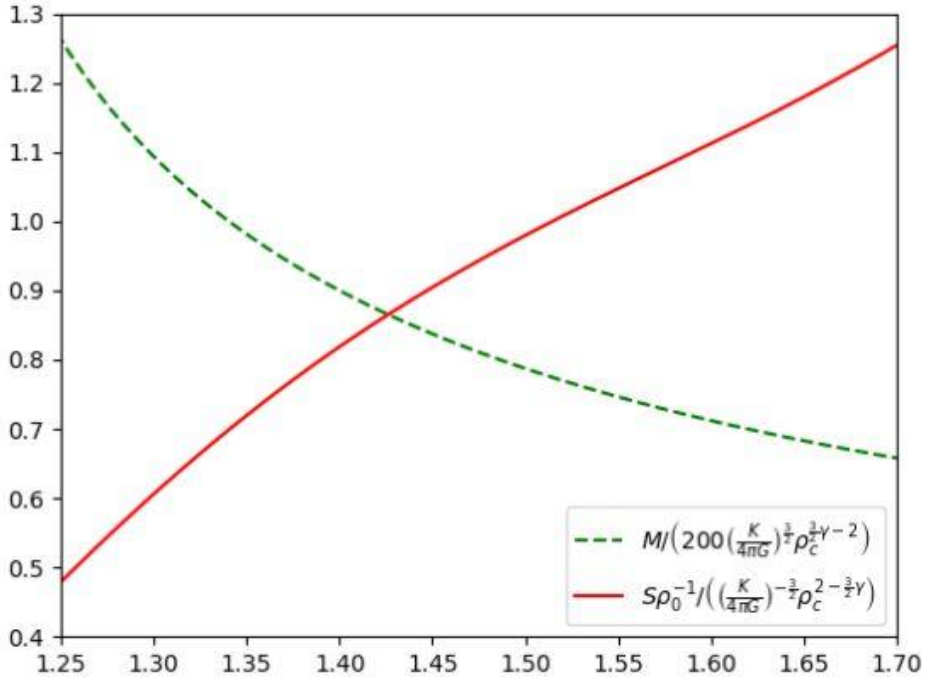
Κώδικας για Figure(3)

```
for index_3, val_2 in enumerate(k3):
    h6 = hak(sol_1.y[0], n_1, sol_1.t, val_2)
    I6 = (simpson(h6, sol_1.t, dx=0.0001)) ** 2
    value_3[index_3] = I6
f_2 = value_3/I3
s_2 = sak(f_2, k3)
S2[index_1] = -4 * np.pi * simpson(s_2, k3, dx=0.0001)

for index_4, val_3 in enumerate(k4):
    h7 = hak(sol_1.y[0], n_1, sol_1.t, val_3)
    I7 = (simpson(h7, sol_1.t, dx=0.0001)) ** 2
    value_4[index_4] = I7
f_3 = value_4 / I4
s_3 = sak(f_3, k4)
S3[index_1] = -4 * np.pi * simpson(s_3, k4, dx=0.0001)
```

Αποτελέσματα



Αποτελέσματα



Αποτελέσματα