Indice

1 Introduzione

Studiamo la stabilità delle stelle di neutroni in regime relativistico considerando 3 possibili equazioni di stato per la materia. Una volta risolte le equazioni è possibile ottenere l'espressione del potenziale gravitazione della stella e calcolare l'effetto sulla radiazione emessa dalla stella.

Viene calcolata la radianza per ogni stella a 3 distanze diverse e la potenza totale di emissione in funzione della distanza dalla stella. Viene quindi calcolata la temperatura apparente delle 3 stelle più massive in funzione di quella effettiva e poi viene studiata la temperatura apparente in funzione della pressione centrale della stella.

2 Stabilità

Le equazioni che descrivono la stabilità di una stella in funzione della massa (m) e della pressione (P) sono quelle di Tolman-Oppenheimer-Volkoff

$$\begin{cases} \dfrac{\mathrm{d}P(r)}{\mathrm{d}r} = -G\dfrac{m(r)\epsilon(r)}{r^2c^2}\left(1+\dfrac{P(r)}{\epsilon(r)}\right)\left(+\dfrac{4\pi r^3 P(r)}{m(r)c^2}\right)\left(1-\dfrac{2Gm(r)}{rc^2}\right)^{-1} \\ \dfrac{\mathrm{d}m(r)}{\mathrm{d}r} = 4\pi r^2\dfrac{\epsilon(r)}{c^2} \\ \dfrac{\mathrm{d}\Phi(r)}{\mathrm{d}r} = -\dfrac{1}{P(r)+\epsilon(r)}\dfrac{\mathrm{d}P(r)}{\mathrm{d}r} \end{cases} \tag{1}$$

Dove la terza equazione è l'equazione disaccoppiata e descrive il potenziale gravitazionale della stella. Usiamo 3 diverse densità di energia per la materia della stella (eq. 2b viene presa con due coppie di valori diversi di Γ e K):

$$\epsilon_1(n) = a\left(\frac{n}{n_0}\right)^{\alpha} + b\left(\frac{n}{n_0}\right)^{\beta} \tag{2a}$$

$$\epsilon_{2/3}(n) = \mu c^2 n + Kc^2 n^{\Gamma} \tag{2b}$$

$$\text{con} \quad a = 13.4\text{MeV fm}^{-3}\ , \quad \alpha = 0.514, \quad b = 5.62\text{MeV fm}^{-3}\ , \quad \beta = 2.436\ , \quad n_0 = 0.16\text{fm}^{-3} \tag{3}$$

dove n è la densità numerica, μ la massa di una singola particella e quindi $\rho = \mu n$ è la densità di massa.

Visto che la densità di energia è in funzione di ρ e le incognite del sistema 1 sono P e m possiamo scrivere la densità di energia in funzione di P e m partendo dalla relazione termodinamica

$$P = -\frac{\mathrm{d}E}{\mathrm{d}V} \Rightarrow \begin{cases} P = (\alpha-1)a\left(\dfrac{n}{n_0}\right)^{\alpha} + (\beta-1)b\left(\dfrac{n}{n_0}\right)^{\beta} & \text{per } \epsilon_1 \quad (4a) \\ n = \left(\dfrac{P}{K(\Gamma-1)c^2}\right)^{1/\Gamma} & \text{per } \epsilon_{1/2} \quad (4b) \end{cases}$$

Nel primo caso (eq. 4a) non è stato possibile invertire l'equazione per trovare n in funzione di P e m quindi utilizzeremo un metodo numerico per trovare n di volta in volta.

Facciamo le seguenti sostituzioni per rendere le variabili adimensionali e con valori più vicini a 0.

$$m = M_0 \hat{m}, \quad r = R_0 \hat{r}, \quad P = P_0 \hat{P}, \quad \rho = \rho_0 \hat{\rho}, \quad K = \hat{K} \frac{\mu^\Gamma}{\rho_0^{\Gamma - 1}},$$

$$\begin{cases} \dfrac{\mathrm{d}\hat{P}}{\mathrm{d}\hat{r}} = -\dfrac{(\hat{P} + \hat{\epsilon})(\hat{m} + \hat{r}^3 \hat{P})}{\hat{r}^2 - 2\hat{m}\hat{r}} & \text{(5a)} \\ \dfrac{\mathrm{d}\hat{m}}{\mathrm{d}\hat{r}} = \hat{r}^2 \hat{\epsilon} & \text{(5b)} \\ \dfrac{\mathrm{d}\Phi}{\mathrm{d}\hat{r}} = -\dfrac{1}{\hat{P} + \hat{\epsilon}} \dfrac{\mathrm{d}\hat{P}}{\mathrm{d}\hat{r}} & \text{(5c)} \end{cases}$$

Otteniamo il sistema 5 dove, grazie alle equazioni in 4, $\hat{m}$, $\hat{P}$ e $\hat{\epsilon}$ sono funzioni di $\hat{r}$. Come valori delle costanti sono stati usati

$$M_0 = 12.655756 M_\odot \quad R_0 = 20.06145 \mathrm{km} \quad \epsilon = P_0 = \rho_0 c^2 = 150.174 \frac{\mathrm{MeV}}{c^2 \mathrm{fm}} \tag{6}$$

Per il potenziale gravitazionale Φ si può inoltre trovare una soluzione analitica all'esterno della stella che possiamo mettere in forma adimensionale (eq. 7), dove $\hat{M}$ e $\hat{R}$ sono rispettivamente la massa totale e il raggio della stella in forma adimensionale.

$$\Phi_{\mathrm{ext}}(r) = \frac{1}{2} \log\left(1 - \frac{2GM}{rc^2}\right) \implies \Phi_{\mathrm{ext}}(\hat{r}) = \frac{1}{2} \log\left(1 - \frac{2\hat{M}}{\hat{r}}\right) \qquad \hat{r} \geq \hat{R} \tag{7}$$

3 Curva massa raggio

Cominciamo con il risolvere le prime due equazioni 5a e 5b del sistema adimensionale con il metodo `RK4` (appendice A). Per il caso con equzione di stato più complessa (eq. 2a) viene risolta numericamente l'equazione 4a per trovare ρ da P (Appendice B).

Segliamo massa iniziale 0 e pressioni iniziali differenti in modo da trovare soluzioni con R compreso tra i 3 e i 60 km. Il grafico massa raggio trovato viene riportato in figura 1.

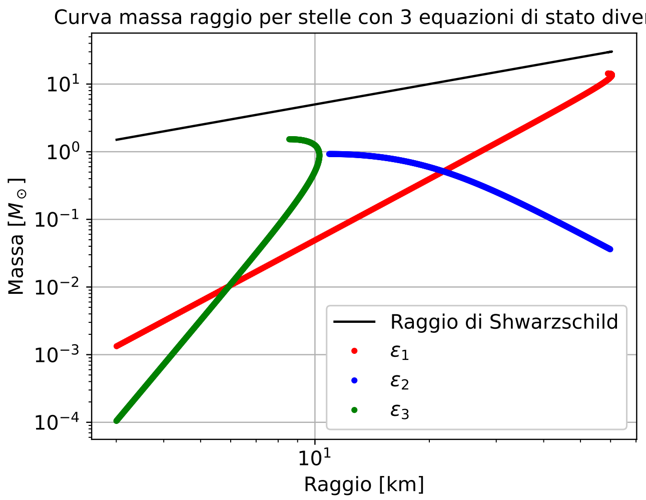

Figura 1: Caption

Durante l'esecuzione del programma abbiamo inoltre smesso di incrementare la pressione centrale iniziale quando la condizione di stabilità $\frac{\mathrm{d}M}{\mathrm{d}r} > 0$ veniva meno.

Notiamo subito che con l'equazione di stato 2a il modello prevede stelle con massa e raggio maggiore rispetto al limite previsto dagli altri modelli.

A RK4

Codice con cui è stato implementato il metodo `RK4`. `fun_P()` e `fun_m()` sono le funzioni presenti a destra dell'uguale nella prima e nella seconda riga del sistema 5.

```
void rungeKutta4(double h, double r, double *P, double *m, int tipo_politropica){

    double k1, k2, k3, k4, l1, l2, l3, l4;

    k1 = h * fun_m(r, *P, tipo_politropica);
    l1 = h * fun_P(r, *P, *m, tipo_politropica);

    k2 = h * fun_m(r + h / 2, *P + l1 / 2, tipo_politropica);
    l2 = h * fun_P(r + h / 2, *P + l1 / 2, *m + k1 / 2, tipo_politropica);

    k3 = h * fun_m(r + h / 2, *P + l2 / 2, tipo_politropica);
    l3 = h * fun_P(r + h / 2, *P + l2 / 2, *m + k2 / 2, tipo_politropica);

    k4 = h * fun_m(r + h, *P + l3, tipo_politropica);
    l4 = h * fun_P(r + h, *P + l3, *m + k3, tipo_politropica);

    *m += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    *P += (l1 + 2 * l2 + 2 * l3 + l4) / 6;
}
```

B Risoluzione numerica di $\hat{P}(\rho)$

Quando si calcola il valore della derivata di P o m ad un dato r serve anche il valore dell'energia interna, infatti

```
// fun_m = r^3 E
double fun_m(double r, double P, int tipo_politropica){
    return r * r * fun_E(P, tipo_politropica);
}


// fun_P = - (P + E)(m + r^3 P)/(r^2 - 2mr)
double fun_P(double r, double P, double m, int tipo_politropica){
    if (m == 0)
        return 0;
    return (P + fun_E(P, tipo_politropica)) * (m + pow(r, 3) * P) / ((2 * m - r) * r);
}
```

Dove la funzione `fun_E()` è definita come

```
double fun_E(double P, int tipo_politropica){
    // Politropica quasi realistica (a*rho^alpha + b*rho^beta)
    if (tipo_politropica == 0){
        double rho = findRho(P);
        return A * pow(rho, ALPHA) + B * pow(rho, BETA);
    }

    double lambda, K;

    // Politropiche semplici
    if (tipo_politropica == 1){
        lambda = 5. / 3.;
        K = 0.05;
    } else if (tipo_politropica == 2){
        lambda = 2.54;
        K = 0.01;
    } else {
        printf("Tipo politropica non riconosciuto\n");
```

```
19            return 0;
20        }
21
22        double a1 = P / (lambda - 1);
23        return a1 + pow(a1 / K, 1. / lambda);
24    }
```

Per le politropiche semplici (eq. 2b) abbiamo potuto trovare un'espressione analitica per $\rho(P)$ (eq. 4b). L'energia può quindi essere calcolata in modo diretto come viene fatto nelle righe 22-23 del codice sopra riportato.

Per la politropica 2a, la relazione tra ρ e P che si trova è data in 4a, che riportiamo

$$P = (\alpha - 1)a\left(\frac{n}{n_0}\right)^{\alpha} + (\beta - 1)b\left(\frac{n}{n_0}\right)^{\beta} \ . \tag{8}$$

Data una certa pressione P bisopgna quindi risolvere numericamente l'equazione per trovare il valore n che la soddisfa. Per fare ciò utiliziamo la funzione `findRho()` così definita

```
double findRho(double P){

    // Cominciamo prima con il metodo di Newton-Raphson
    if (P > 0.01){

        double rho = pow(P / (BETA1 * B), 1 / BETA);     // buona approx. iniziale
        while (fabs(P - P_of_rho(rho)) > 1e-6){
            rho -= (P_of_rho(rho) - P) / DP_of_rho(rho);
        }
        return rho;
    }

    // Per P piccoli meglio usare bisezione
    double rho_sx = 0.7, rho_dx = 1.;
    double rho = (rho_sx + rho_dx) / 2;
    while (fabs(P - P_of_rho(rho)) > 1e-6){
        if (P_of_rho(rho) > P)
            rho_dx = rho;
        else
            rho_sx = rho;
        rho = (rho_sx + rho_dx) / 2;
    }
    return rho;
}
```

dove `P_odf_rho()` e `DP_of_rho()` sono rispettivamente la funzione 8 e la sua derivata. In questo modo possiamo sempre trasformare $\epsilon(\rho)$ in $\epsilon(P)$.