

Indice

1	Introduzione	2
2	Stabilità	2
3	Curva massa raggio	3
4	Potenziale gravitazionale	4
5	Radianza	5
6	Potenza emessa	5
6.1	Convergenza dell'integrale	7
6.2	Potenza totale in funzione della distanza	8
7	Temperatura Percepita	9
8	Temperatura efficace in funzione della pressione centrale	10
A	RK4	11
B	Risoluzione numerica di $\hat{P}(\rho)$	11
C	Risoluzione dell'integrale del potenziale gravitazionale	12
D	Integrali con trapezi e simpson	13
E	Calcolo Temperatura efficace	16

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 gravitazionale 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

$$\begin{cases} \frac{dP(r)}{dr} = -G \frac{m(r)\epsilon(r)}{r^2 c^2} \left(1 + \frac{P(r)}{\epsilon(r)}\right) \left(1 + \frac{4\pi r^3 P(r)}{m(r)c^2}\right) \left(1 - \frac{2Gm(r)}{rc^2}\right)^{-1} \end{cases} \quad (1a)$$

$$\begin{cases} \frac{dm(r)}{dr} = 4\pi r^2 \frac{\epsilon(r)}{c^2} \end{cases} \quad (1b)$$

$$\begin{cases} \frac{d\Phi(r)}{dr} = -\frac{1}{P(r) + \epsilon(r)} \frac{dP(r)}{dr} \end{cases} \quad (1c)$$

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):

$$\epsilon_1(n) = \mu c^2 n + a n_0 \left(\frac{n}{n_0}\right)^{\alpha+1} + b n_0 \left(\frac{n}{n_0}\right)^{\beta+1} \quad (2a)$$

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

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

dove n è la densità numerica, μ la massa del neutrone 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

$$P = -\frac{dE}{dV} \Rightarrow \begin{cases} P = \alpha a n_0 \left(\frac{n}{n_0}\right)^{\alpha+1} + \beta b n_0 \left(\frac{n}{n_0}\right)^{\beta+1} & \text{per } \epsilon_1 \end{cases} \quad (4a)$$

$$\begin{cases} n = \left(\frac{P}{K(\Gamma-1)c^2}\right)^{1/\Gamma} & \text{per } \epsilon_{2/3} \end{cases} \quad (4b)$$

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.

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

$$\begin{cases} \frac{d\hat{P}}{d\hat{r}} = -\frac{(\hat{P} + \hat{\epsilon})(\hat{m} + \hat{r}^3 \hat{P})}{\hat{r}^2 - 2\hat{m}\hat{r}} \end{cases} \quad (5a)$$

$$\begin{cases} \frac{d\hat{m}}{d\hat{r}} = \hat{r}^2 \hat{\epsilon} \end{cases} \quad (5b)$$

$$\begin{cases} \frac{d\Phi}{d\hat{r}} = -\frac{1}{\hat{P} + \hat{\epsilon}} \frac{d\hat{P}}{d\hat{r}} \end{cases} \quad (5c)$$

Otteniamo il sistema 5, dove le densità di energia sono

$$\hat{\epsilon}_1(\hat{n}) = \hat{n} + \frac{an_0}{P_0} \hat{n}^{\alpha+1} + \frac{bn_0}{P_0} \hat{n}^{\beta+1} \quad \text{dove vale} \quad \hat{P} = \frac{\alpha an_0}{P_0} \hat{n}^{\alpha+1} + \frac{\beta bn_0}{P_0} \hat{n}^{\beta+1} \quad (6a)$$

$$\hat{\epsilon}_2(\hat{P}) = \left(\frac{\hat{P}}{\hat{K}(\Gamma-1)} \right)^{1/\Gamma} + \frac{\hat{P}}{\Gamma-1} \quad (6b)$$

Nel caso di ϵ_1 non è possibile invertire la relazione $P(n)$ e quindi si utilizzerà un metodo numerico. Per le politropiche ϵ_2 e ϵ_3 si utilizzano i seguenti valori

	\hat{K}	Γ
ϵ_2	5/3	0.05
ϵ_3	2.54	0.01

Infine, i valori delle costanti utilizzate per definire le variabili adimensionali sono i seguenti

$$M_0 = 12.655756 M_\odot \quad R_0 = 20.06145 \text{ km} \quad \epsilon = P_0 = \rho_0 c^2 = n_0 \mu c^2 = 150.174 \frac{\text{MeV}}{c^2 \text{ fm}^3} \quad (7)$$

Per il potenziale gravitazionale Φ si può inoltre trovare una soluzione analitica all'esterno della stella che possiamo mettere in forma adimensionale (eq. 8), dove \hat{M} e \hat{R} sono rispettivamente la massa totale e il raggio della stella in forma adimensionale.

$$\Phi_{\text{ext}}(r) = \frac{1}{2} \log \left(1 - \frac{2GM}{rc^2} \right) \implies \Phi_{\text{ext}}(\hat{r}) = \frac{1}{2} \log \left(1 - \frac{2\hat{M}}{\hat{r}} \right) \quad \hat{r} \geq \hat{R} \quad (8)$$

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 equazione di stato più complessa (eq. 6a) viene risolta numericamente l'equazione $\hat{P}(\hat{n})$ a fianco per trovare ρ da un P fissato (Appendice B).

Scegliamo massa iniziale 0 e pressioni iniziali differenti in modo da trovare soluzioni con R compreso tra i 3 e i 40 km. Il grafico massa raggio trovato viene riportato in figura 1.

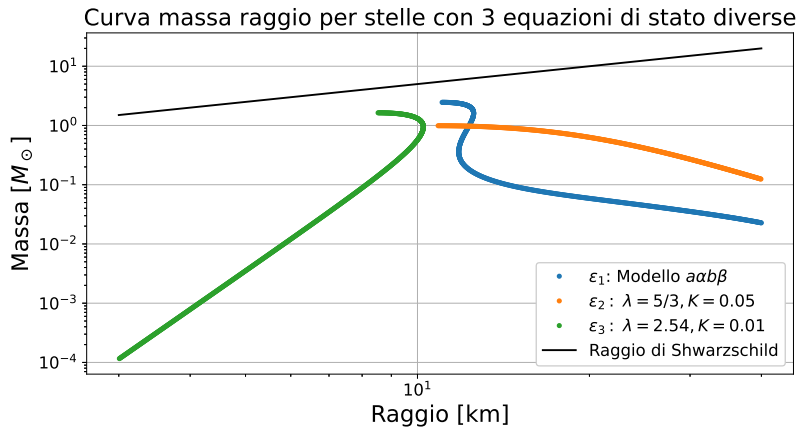


Figura 1: Curva massa raggio per stelle di equazioni di stato $\epsilon_{1/2/3}$. La prima equazione di stato, quella più realistica, prevede stelle di neutroni più massive.

Durante l'esecuzione del programma abbiamo inoltre smesso di incrementare la pressione centrale iniziale quando la condizione di stabilità $\frac{dM}{dr} > 0$ 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.

I valori della stella più massiva per ognuna delle 3 equazioni di stato vengono riportati nella tabella 1.

	$P_0 [\frac{\text{MeV}}{c^2 \text{fm}^3}]$	$R [\text{km}]$	$M [M_\odot]$
ϵ_1	885.2114	11.04289	2.456841
ϵ_2	217.2936	10.86752	0.990100
ϵ_3	948.5211	8.531525	1.635845

Tabella 1: Valori della pressione iniziale P_0 , massa totale M e raggio R della stella più massiva per ogni equazione di stato utilizzata.

Ci riferiremo alla stella più massiva ottenuta con la politropica 1 (politropica realistica) come *stella 1* che ha mass M_1 raggio R_1 , alla stella con la politropica 2 *stella 2* e così via.

4 Potenziale gravitazionale

Come mostrato nel sistema 1, l'equazione che determina il potenziale gravitazionale è disaccoppiata dalle altre due e, per $r \geq R$, può anche essere risolta analiticamente. Utilizzando le equazioni 5c e 8 possiamo trovare l'espressione generale per il potenziale all'interno della stella riportata in eq. 9.

$$\Phi_{\text{int}}(\hat{r}) = \Phi_{\text{ext}}(\hat{R}) - \int_{\hat{r}}^{\hat{R}} \frac{d\Phi}{dx} dx = \frac{1}{2} \log \left(1 - \frac{2\hat{M}}{\hat{R}} \right) + \int_{\hat{r}}^{\hat{R}} \frac{1}{\hat{P}(x) + \hat{\epsilon}(x)} \frac{d\hat{P}(x)}{dx} dx \quad (9)$$

dove siamo stati attenti a rendere $\Phi(r)$ continuo per ogni $r \geq 0$ usando il valore di Φ_{ext} in \hat{R} . Infine sostituendo alla derivata della pressione l'espressione in 5a otteniamo

$$\Phi_{\text{int}}(\hat{r}) = \frac{1}{2} \log \left(1 - \frac{2\hat{M}}{\hat{R}} \right) + \int_{\hat{r}}^{\hat{R}} \frac{\hat{m}(x) + x^3 \hat{P}(x)}{2\hat{m}(x)x - x^2} dx \quad (10)$$

Dati i valori di $\hat{P}(\hat{r})$ e $\hat{m}(\hat{r})$ che si ottengono risolvendo le equazioni di stabilità possiamo quindi risolvere l'integrale con il metodo dei trapezi per ottenere il valore di Φ a ogni r (Appendice C). Il grafico del potenziale gravitazionale ottenuto è mostrato in figura 2.

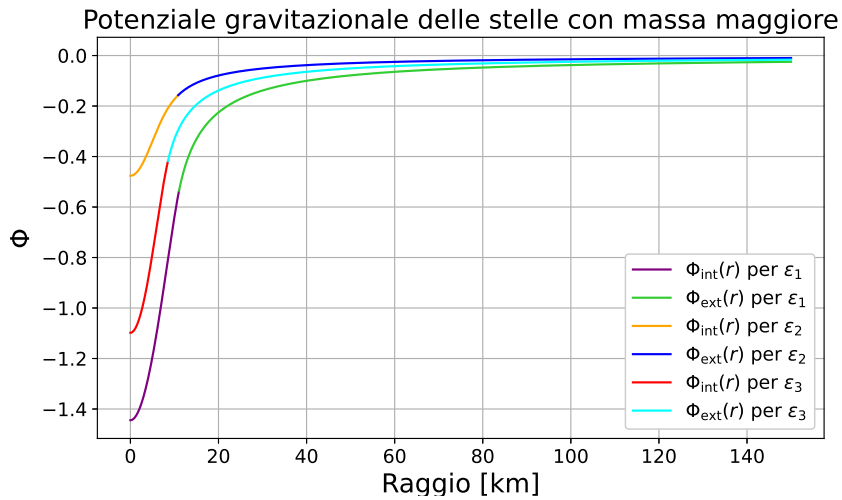


Figura 2: Grafico del potenziale gravitazionale all'interno e all'esterno della stella. Per $r < R$ è stato ottenuto integrando con il metodo dei trapezi l'eq. 10, per $r \geq R$ è stata plottata l'eq. 8

Vediamo dalla figura 2 che in tutti e 3 i casi $\Phi(r)$ è continuo, grazie alla condizione imposta.

Il potenziale Φ (che fa parte del termine $e^{\Phi(r)}c^2dt^2$ della metrica ds^2 , che descrive la geometria dello spazio vicino alla stella) assume un andamento familiare: raggiunge valori più bassi per le stelle più massive al diminuire di r e tende a 0 per r grandi.

5 Radianza

In metrica di Schwarzschild un fotone emesso a distanza r con frequenza ν_{em} viene ricevuto da un osservatore a distanza r' con una frequenza ν_{ric} data da

$$\frac{\nu_{\text{ric}}}{\nu_{\text{em}}} = e^{\Phi(r) - \Phi(r')}. \quad (11)$$

Dove $\Phi(r)$ è proprio il potenziale gravitazionale mostrato in figura 2.

La radianza di una stella si può esprimere con l'equazione di Plank per il corpo nero

$$B(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{h\nu/(k_B T)} - 1} \quad (12)$$

dove T è la temperatura della stella e h la costante di Plank. Per fare i conti considerando solo il caso con $K_B T = 1\text{MeV}$ e mettiamo l'equazione 12 in funzione di variabili adimensionali. Otteniamo

$$\hat{B}(\hat{\nu}) = \frac{\hat{\nu}^3}{e^{\hat{\nu}} - 1} \quad (13)$$

dove abbiamo definito

$$\nu_0 = \frac{\nu}{\hat{\nu}} = \frac{1\text{MeV}}{h} \simeq 2.417989 \times 10^{20}\text{Hz} \quad (14)$$

$$B_0 = \frac{B}{\hat{B}} = \frac{2h\nu_0^3}{c^2} = \frac{2\text{MeV}}{(hc)^2} \simeq 1.301059 \times 10^{-6} \frac{\text{MeV}}{\text{fm}^2} \quad (15)$$

Infine, se consideriamo l'effetto doppler dovuto al potenziale gravitazionale descritto in eq. 11 e utilizziamo la formula analitica per il potenziale gravitazionale all'esterno della stella (eq. 8), otteniamo

$$\hat{\nu}_{\text{em}} = \hat{\nu}_{\text{ric}} \left(\frac{1 - \frac{2\hat{M}}{r}}{1 - \frac{2\hat{M}}{R}} \right)^{1/2} \implies \hat{B}(\hat{\nu}_{\text{ric}}, \hat{r}) = \frac{\hat{\nu}_{\text{ric}}^3 \left(1 - \frac{2\hat{M}}{\hat{r}}\right)^{3/2} \left(1 - \frac{2\hat{M}}{R}\right)^{-3/2}}{\exp\left(\hat{\nu}_{\text{ric}} \left(1 - \frac{2\hat{M}}{\hat{r}}\right)^{1/2} \left(1 - \frac{2\hat{M}}{R}\right)^{-1/2}\right) - 1} \quad (16)$$

Nel figure 3, 4 e 5 viene studiato lo spettro della radiazione emessa per le 3 stelle massive di cui abbiamo studiato il potenziale in 2 a distanze diverse dalla stella. In blu è plottata l'eq. 12, ovvero la radianza senza correzioni relativistiche.

La curva in tutti e 3 i casi subisce uno spostamento verso le frequenze più basse, *redshift* per l'appunto, e l'effetto è tanto maggiore quanto più la stella è massiva e l'osservatore distante. Già a una distanza di 8 volte il raggio della stella la curva della radianza è molto simile a quella all'infinito, in cui il termine $\Phi(r')$ in eq. 11 è trascurabile.

6 Potenza emessa

La potenza totale emessa dalla stella per un elemento della sua superficie, si può ottenere integrando la radianza $B(\nu, T)$ sulle frequenze e sull'angolo solido, con l'accortezza di moltiplicare per un fattore $\cos(\theta)$ (Legge di Lambert) per proiettare la radiazione lungo la normale alla superficie sottesa da $d\Omega$.

$$\mathcal{P} = \int_0^\infty d\nu \int B(\nu, T) \cos(\theta) d\Omega = \int_0^{Ak_B T} B(\nu, T) d\nu \int_0^{2\pi} d\phi \int_0^1 d(\cos(\theta)) \cos(\theta) \quad (17)$$

$$\simeq \pi \int_{\nu_{\text{ric}}=0}^{\nu_{\text{ric}}=Ak_B T} B(\nu_{\text{em}}, T) d\nu_{\text{ric}} \quad (18)$$

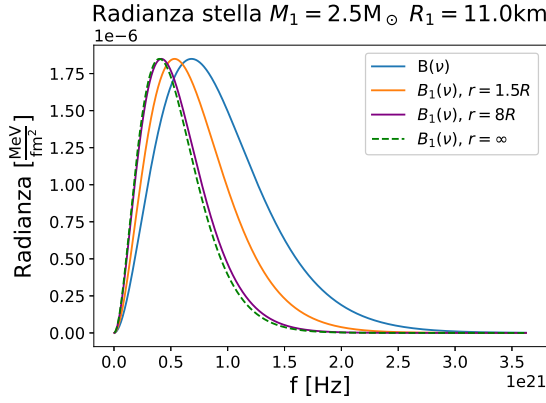


Figura 3: Radianza di una stella con $M = 2.5M_{\odot}$ e $R = 11.0\text{km}$, percepita a 16.5km, 88km e a distanza infinita. Essendo la stella con massa maggiore è anche quella in cui la curva è più spostata verso sinistra.

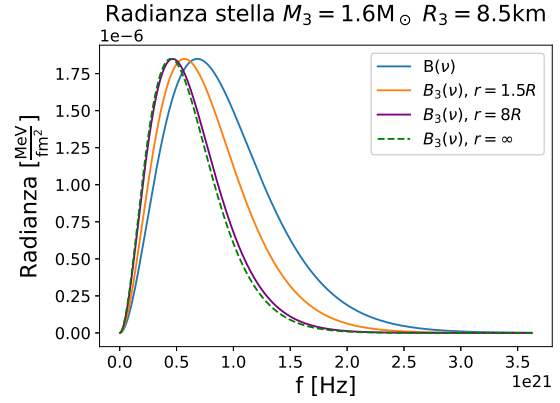


Figura 4: Radianza da una stella con $M = 1.6M_{\odot}$ e $R = 8.5\text{km}$, percepita a 12.8km, 68km e a distanza infinita. In blu la radianza senza correzioni relativistiche.

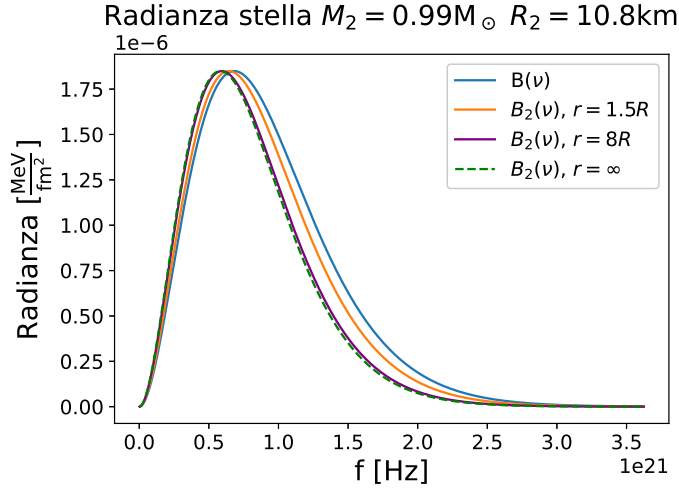


Figura 5: Radianza da una stella con $M = 0.99M_{\odot}$ e $R = 10.8\text{km}$, percepita a 16.2, 86.4 km e a distanza infinita. In blu la radianza senza correzioni relativistiche.

Nell'ultimo passaggio (eq. 18) è esplicitato che l'integrazione viene fatta su ν_{ric} , ovvero le frequenze misurate a distanza r , abbiamo messo un upperbound finito all'integrale e, per semplificare i calcoli per il prossimo punto, abbiamo lasciato la dipendenza da ν_{em} in B .

Passiamo quindi alle unità adimensionali, riportate in eq. 22, e facciamo un cambio di variabile all'interno dell'integrale utilizzando la formula del redshift $\nu_{\text{ric}} = \nu_{\text{em}} e^{\Phi(R) - \Phi(r)}$:

$$\hat{\mathcal{P}}(\hat{r}) = \int_{\hat{\nu}_{\text{ric}}=0}^{\hat{\nu}_{\text{ric}}=\frac{AT_0}{\nu_0}} \hat{B}(\hat{\nu}_{\text{em}}, \hat{T}) d\hat{\nu}_{\text{ric}} = e^{\Phi(\hat{R}) - \Phi(\hat{r})} \int_{\hat{\nu}_{\text{em}}=0}^{\hat{\nu}_{\text{em}}=\frac{AT_0\hat{T}}{\nu_0} e^{\Phi(\hat{R}) - \Phi(\hat{r})}} \frac{\hat{\nu}_{\text{em}}^3}{\exp(\hat{\nu}_{\text{em}}/\hat{T}) - 1} d\hat{\nu}_{\text{em}} \quad (19)$$

$$= \left(1 - \frac{2\hat{M}}{\hat{R}}\right)^{1/2} \left(1 - \frac{2\hat{M}}{\hat{r}}\right)^{-1/2} \int_0^{\frac{AT_0\hat{T}}{\nu_0} e^{\Phi(\hat{R}) - \Phi(\hat{r})}} \frac{\nu^3}{e^{\nu/\hat{T}} - 1} d\nu \quad (20)$$

$$= \left(1 - \frac{2\hat{M}}{\hat{R}}\right)^{1/2} \left(1 - \frac{2\hat{M}}{\hat{r}}\right)^{-1/2} \int_0^{\frac{AT_0\hat{T}}{\nu_0}} \frac{\nu^3}{e^{\nu/\hat{T}} - 1} d\nu. \quad (21)$$

Dove sono state usate le definizioni di $B(\hat{\nu}, \hat{T})$ (eq. 13) e quella di $\Phi(r)$ (eq. 8) e nell'ultimo

passaggio in 21 si è utilizzato il fatto che nell'integrale $A \rightarrow \infty$ e che $e^{\Phi(R)-\Phi(r)} < 1$, ignorare il termine migliora quindi la nostra approssimazione.

$$\mathcal{P}_0 = \frac{\mathcal{P}}{\hat{\mathcal{P}}} = \pi B_0 \nu_0 \simeq 9.883\,290 \times 10^{14} \frac{\text{MeV}}{\text{s fm}^2}, \quad T_0 = T/\hat{T} = 1 \text{ MeV}. \quad (22)$$

6.1 Convergenza dell'integrale

Dal momento che numericamente non è possibile integrare sulle frequenze fino a $\nu = +\infty$, nei passaggi in eq. 18 e poi in eq. 21 abbiamo utilizzato un parametro A che deve essere sufficientemente grande da poter approssimare in modo ragionevole l'integrale, per i valori di \hat{T} utili. Riportiamo in eq. 23 l'integrale da calcolare.

$$\mathcal{I}(\hat{T}) = \int_0^{\frac{AT_0\hat{T}}{\nu_0}} \frac{\nu^3}{e^{\nu/\hat{T}} - 1} d\nu = \int_0^{\frac{AT_0}{\nu_0}} \frac{\nu^3}{e^{\nu/\hat{T}} - 1} d\nu. \quad (23)$$

A priori gli estremi di integrazione dipendono da \hat{T} , dal momento che stiamo studiando la radiazione emessa da stelle di neutroni siamo interessati a studiare un range di temperature più piccolo. In particolare non di troppo superiori a $k_B T = 1 \text{ MeV}$, ovvero $\hat{T} = 1$ che decidiamo essere il nostro upperbound per la temperatura. Per temperature inferiori la funzione va a 0 più velocemente e quindi l'approssimazione di A come finito è ancora migliore.

Facendo riferimento alla figura 4 dove in blu è plottata la funzione che dobbiamo integrare per $\hat{T} = 1$, possiamo vedere che per $\nu > 3.5 \times 10^{21} \text{ Hz}$ la radianza è quasi nulla. In quell'occasione era stata calcolata $\hat{B}(\hat{\nu})$ con $\hat{\nu}$ tra 0 e 15. Come stima iniziale possiamo quindi integrare fino a $\hat{\nu} = 20$ che corrisponde a

$$\frac{AT_0}{\nu_0} = 20 \quad \Longleftrightarrow \quad A = 20 \frac{\nu_0}{T_0} \simeq 4.835\,978 \times 10^{21} \text{ MeV}^{-1} \text{ s}^{-1} \quad (24)$$

Possiamo ottenere un valore migliore valutando la convergenza dell'integrale.

Per prima cosa verifichiamo per quale N (numero di step nell'integrazione) il valore di \mathcal{I} converge (il codice si trova in Appendice D). In questo caso la situazione peggiore è quella per temperature molto piccole, scegliamo come lowerbound $\hat{T} = 0.01$, poiché la curva è molto piccola e stretta. Otteniamo il grafico in figura 6. I valori sono degli scarti rispetto a $\mathcal{I}_{N=1e8}$ (l'integrale calcolato con $N = 1 \times 10^8$) e sono normalizzati rispetto allo stesso. Al contrario di quello che succede di solito, il metodo Simpson (che ha un costo computazionale maggiore), non presenta vantaggi rispetto a quello dei trapezi. Dal codice (Appendice D) che genera i dati mostrati in figura 6 otteniamo un errore relativo minore di 1×10^{-7} per $N_{\text{trap}} = 19060$ e $N_{\text{simp}} = 25368$.

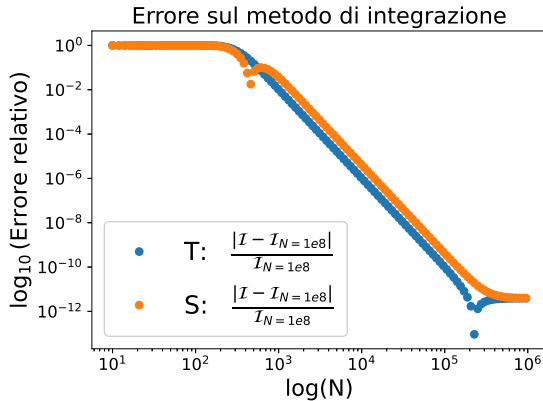


Figura 6: Errore relativo sul calcolo di $\mathcal{I}(1)$ per diversi N (numero di step nell'integrazione con i trapezi, T, e con simpson, S).

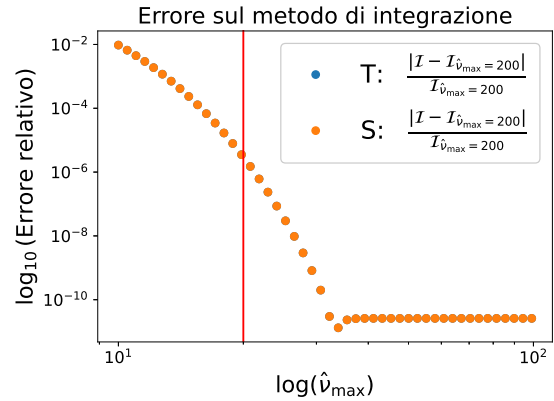


Figura 7: Errore relativo sul calcolo di $\mathcal{I}(1)$ per diversi valori di $\hat{\nu}_{\text{max}}$. Poco dopo la stima iniziale di $\hat{\nu}_{\text{max}} = 20$ (linea verticale rossa) l'errore si stabilizza. L'errore non diminuisce come prima perché c'è una sorta di offset dato dall'aver fissato il rapporto $N / \text{nu_max}$.

Fissati N_{trap} e N_{simp} per un intervallo $\Delta\hat{\nu} = 20$ proponiamo quindi in figura 7 lo stesso tipo di grafico degli scarti fatto però in funzione della scelta di ν_{max} , dove abbiamo tenuto costante il valore $N / \text{nu_max}$ appena ottenuto. Questa volta si studia il caso in cui $r = R_1$, ovvero quello in cui la curva è più spostata verso destra e serve un upper bound maggiore.

Dallo script per generare i dati mostrati in figura 7 otteniamo un errore di 1×10^{-7} per

	N	$\hat{\nu}_{\text{max}}$
Trapezi	22936	24.0661923
Simpson	30516	24.0661923

Di conseguenza possiamo valutare

$$A = 24.0661923 \nu_0 \text{ MeV}^{-1} \simeq 5.819179 \times 10^{21} \text{ MeV}^{-1} \text{ s}^{-1} \quad (25)$$

Con questi parametri otteniamo i valori \mathcal{I}_T , per i trapezi, e \mathcal{I}_S , per Simpson:

$$\mathcal{I}_T(1) = 6.4939388 \quad \mathcal{I}_S(1) = 6.4939388. \quad (26)$$

Gli integrali hanno lo stesso valore poiché ci siamo assicurati di avere una precisione di 1×10^{-7} per il caso peggiore a $\hat{T} = 0.01$, che servirà nella sezione 7.

6.2 Potenza totale in funzione della distanza

Studiato l'integrale che serve per il calcolo della potenza possiamo riscrivere $\hat{\mathcal{P}}$ così

$$\hat{\mathcal{P}}(\hat{r}, \hat{T}) = \left(1 - \frac{2\hat{M}}{\hat{R}}\right)^{1/2} \left(1 - \frac{2\hat{M}}{\hat{r}}\right)^{-1/2} \mathcal{I}(\hat{T}) \quad (27)$$

Riportiamo quindi in figura 8 il l'andamento di $\mathcal{P}(r, \hat{T} = 1)$ per le stelle più massive descritte in 1.

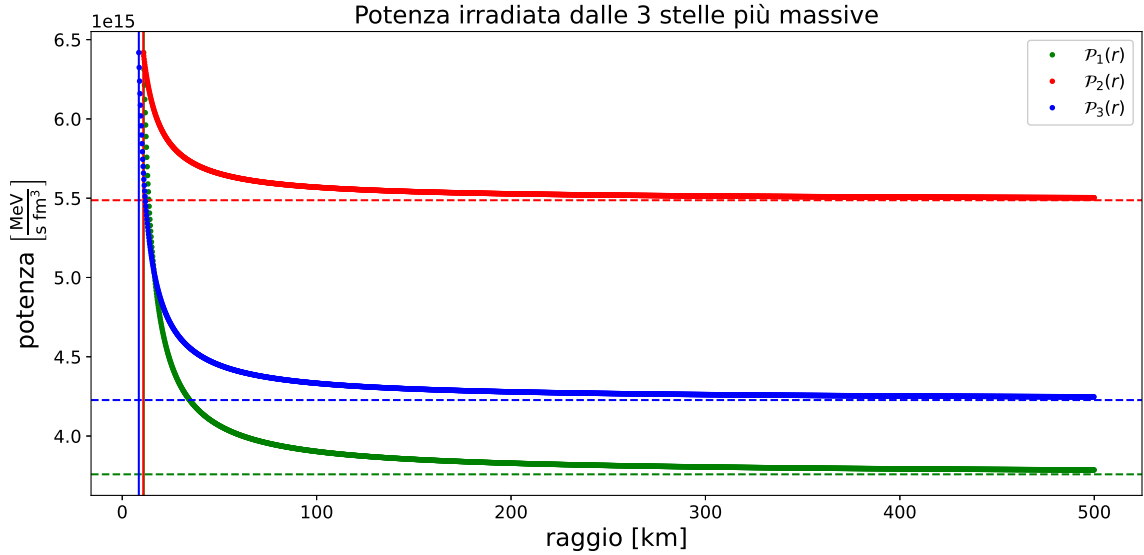


Figura 8: Potenza in funzione della distanza per le 3 stelle: $(R_1 = 59.0\text{km}, M_1 = 14.3M_\odot)$, $(R_2 = 10.9\text{km}, M_2 = 0.926M_\odot)$ e $(R_3 = 8.56\text{km}, M_3 = 1.53M_\odot)$. La linea continua verticale rappresenta il raggio della stella, la linea tratteggiata orizzontale il valore di \mathcal{P} a $r = \infty$.

Con linea tratteggiata e linea continua sono rispettivamente rappresentati raggio e valore della potenza all'infinito per ogni stella analizzata.

7 Temperatura Percepita

Una volta ottenuta la potenza totale irradiata da una stella è possibile calcolare la sua temperatura, grazie alla relazione

$$k_B T_{eff} = \left(\mathcal{P}(T) \frac{15c^2 h^3}{2\pi^5} \right)^{1/4} \quad (28)$$

Dove T_{eff} è la temperatura percepita a una certa distanza dalla stella e T la temperatura reale, che si misurerebbe a $r = R$.

Mettendo le variabili adimensionali, sostituendo a $\hat{\mathcal{P}}$ l'espressione in 27 e ricordando le espressioni di $\hat{\mathcal{P}}_0$ (eq. 22), B_0 (eq. 14) ν_0 (eq. 15) e che $T_0 = \text{MeV}$, otteniamo

$$T_0 \hat{T}_{eff} = \left(\mathcal{P}_0 \frac{15c^2 h^3}{2\pi^5} \right)^{1/4} \hat{\mathcal{P}}^{1/4} = \frac{15^{1/4}}{\pi} \text{MeV} \left(1 - \frac{2\hat{M}}{\hat{R}} \right)^{1/8} \left(1 - \frac{2\hat{M}}{\hat{r}} \right)^{-1/8} \mathcal{I}(\hat{T})^{1/4} \quad (29)$$

$$\hat{T}_{eff}(\hat{r}, \hat{T}) = \frac{15^{1/4}}{\pi} \left(1 - \frac{2\hat{M}}{\hat{R}} \right)^{1/8} \left(1 - \frac{2\hat{M}}{\hat{r}} \right)^{-1/8} \mathcal{I}(\hat{T})^{1/4} \quad (30)$$

Possiamo quindi studiare l'equazione 30 per le 3 stelle studiate in precedenza, questa volta però fissiamo $r = \infty$ e studiamo la T_{eff} in funzione di T , la temperatura propria della stella, facendo il limite

$$\hat{T}_{eff}(\hat{r}, \hat{T}) = \frac{15^{1/4}}{\pi} \left(1 - \frac{2\hat{M}}{\hat{R}} \right)^{1/8} \mathcal{I}(\hat{T})^{1/4} \quad (31)$$

Presentiamo il grafico in figura 9, il codice si trova nell'appendice E.

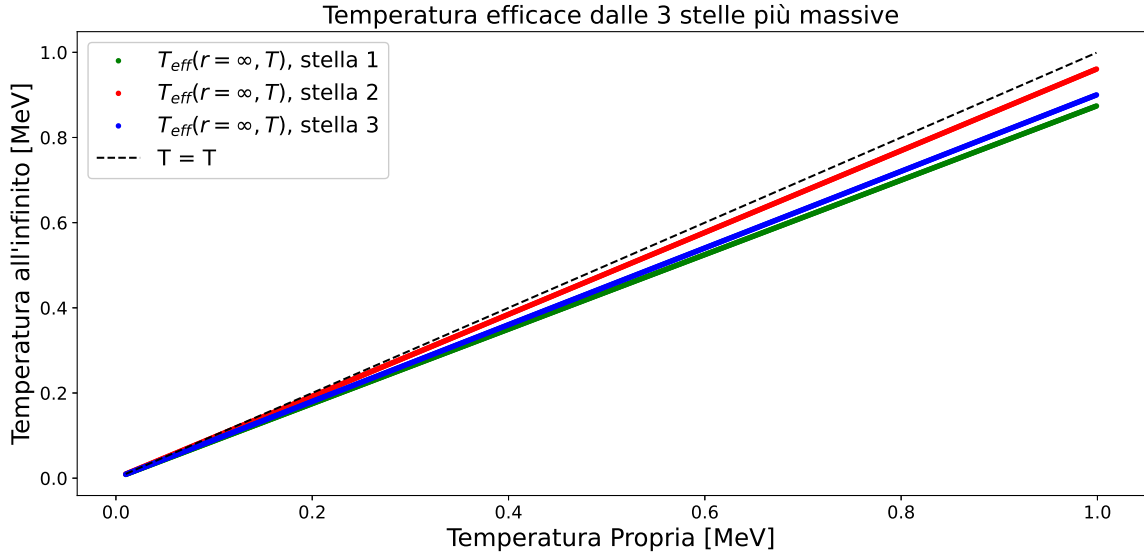


Figura 9: Temperatura percepita all'infinito delle 3 stelle con masse $M_1 = 2.46M_\odot$, $M_2 = 0.99M_\odot$ e $M_3 = 1.64M_\odot$. Come ci aspettavamo la temperatura misurata all'infinito è sempre minore di quella propria della stella.

Come ci aspettavamo la temperatura percepita all'infinito è sempre minore di quella propria della stella. Il grafico mostra anche un'incidenza diversa del redshift per le 3 stelle. In accordo con quello osservato in figura 2 e 8 i fotoni subiscono il redshift maggiore per la stella più massiva $M_1 = 2.46M_\odot$ e seguono in ordine (sempre di massa) effetti minori per $M_3 = 1.64M_\odot$ e $M_2 = 0.995M_\odot$.

8 Temperatura efficace in funzione della pressione centrale

Si presentano infine i grafici della temperatura efficace in funzione della pressione centrale della stella. Vengono mostrati i grafici per tutte e 3 le politropiche. Viene inoltre mostrato il caso per la temperatura della stella $\hat{T} = 1$ a sinistra e quello per $\hat{T} = 0.01$ a destra. Come ci si poteva aspettare una pressione centrale maggiore porta a una stella più massiva, il cui redshift gravitazionale è più forte e quindi la temperatura efficace più bassa.

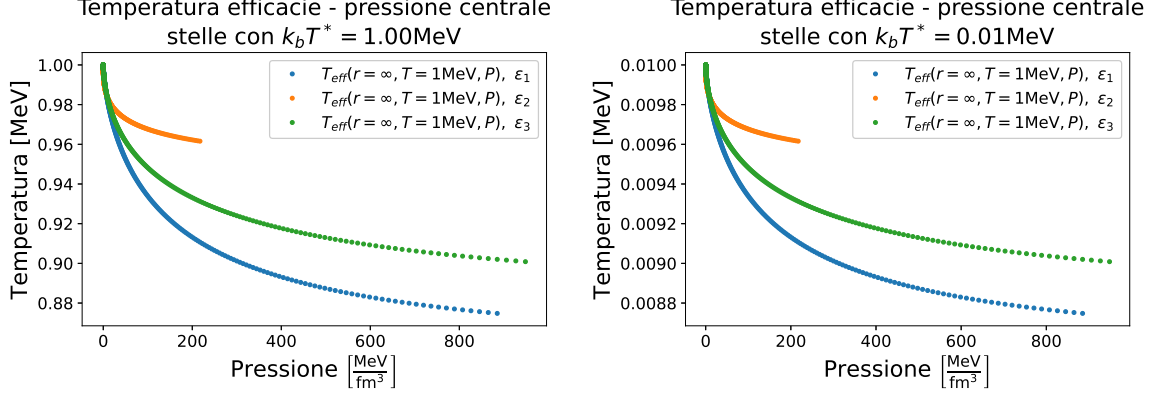


Figura 10: Fissata la temperatura della stella ($\hat{T} = 1$ a sinistra e $\hat{T} = 0.01$ a destra) è possibile esprimere la temperatura misurata all'infinito in funzione della pressione centrale che, data una politropica, ne determina la massa.

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.

```
1 void rungeKutta4(double h, double r, double *P, double *m, int tipo_politropica){
2
3     double k1, k2, k3, k4, l1, l2, l3, l4;
4
5     k1 = h * fun_m(r, *P, tipo_politropica);
6     l1 = h * fun_P(r, *P, *m, tipo_politropica);
7
8     k2 = h * fun_m(r + h / 2, *P + l1 / 2, tipo_politropica);
9     l2 = h * fun_P(r + h / 2, *P + l1 / 2, *m + k1 / 2, tipo_politropica);
10
11    k3 = h * fun_m(r + h / 2, *P + l2 / 2, tipo_politropica);
12    l3 = h * fun_P(r + h / 2, *P + l2 / 2, *m + k2 / 2, tipo_politropica);
13
14    k4 = h * fun_m(r + h, *P + l3, tipo_politropica);
15    l4 = h * fun_P(r + h, *P + l3, *m + k3, tipo_politropica);
16
17    *m += (k1 + 2 * k2 + 2 * k3 + k4) / 6;
18    *P += (l1 + 2 * l2 + 2 * l3 + l4) / 6;
19 }
```

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

```
1 // f_m = r^2 E
2 double fun_m(double r, double P, int tipo_politropica){
3     return r * r * fun_E(P, tipo_politropica);
4 }
5
6
7 // f_P = - (P + E)(m + r^3 P)/(r^2 - 2mr)
8 double fun_P(double r, double P, double m, int tipo_politropica){
9     if (m == 0)
10         return 0;
11     return (P + fun_E(P, tipo_politropica)) * (m + pow(r, 3) * P)
12         / ((2 * m - r) * r);
13 }
```

Dove la funzione `fun_E()` è definita come

```
1 double fun_E(double P, int tipo_politropica){
2
3     // Politropica quasi realistica (rho*mc^2 + a*rho^alpha + b*rho^beta)
4     if (tipo_politropica == 1){
5         double rho = findRho(P);
6         return rho + A * pow(rho, ALPHA + 1.) + B * pow(rho, BETA + 1.);
7     }
8
9     double lambda, K;
10
11    // Materia fermionica non relativistica
12    if (tipo_politropica == 2){
13        lambda = 5. / 3.;
14        K = 0.05;
15    }
16    else if (tipo_politropica == 3){
17        lambda = 2.54;
```

```

18     K = 0.01;
19 }
20 else {
21     printf("Tipo politropica non riconosciuto\n");
22     return 0;
23 }
24
25 double a1 = P / (lambda - 1.);
26 return a1 + pow(a1 / K, 1. / lambda);
27 }

```

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

$$P = (\alpha - 1)a \left(\frac{n}{n_0} \right)^\alpha + (\beta - 1)b \left(\frac{n}{n_0} \right)^\beta. \quad (32)$$

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

```

1 double findRho(double P){
2
3     // Newton-Raphson method to find rho
4     double rho = pow(P / 0.15, 1. / 3.); // buona approssimazione iniziale
5
6     while (fabs(P - P_of_rho(rho)) > 1e-8){
7         rho = (P_of_rho(rho) - P) / DP_of_rho(rho);
8     }
9     return rho;
10 }

```

dove `P_of_rho()` e `DP_of_rho()` sono rispettivamente la funzione 32 e la sua derivata. In questo modo possiamo sempre trasformare $\epsilon(\rho)$ in $\epsilon(P)$.

C Risoluzione dell'integrale del potenziale gravitazionale

Per ognuna delle 3 stelle trovate (la più massiva per ogni diversa equazione di stato) salviamo ogni valore di r , m e P in un file che successivamente importiamo (riga 3) e utilizziamo per calcolare l'integrale

```

1 for (int tipo_politropica = 1; tipo_politropica < 4; tipo_politropica++){
2     int lenfile = len_files[tipo_politropica - 1];
3
4     double r[lenfile], P[lenfile], m[lenfile], Phi[lenfile];
5
6     read_maxM_data(tipo_politropica, lenfile, r, P, m);
7
8     double R = r[lenfile - 1];
9     double M = m[lenfile - 1];
10    double Phi_ext = fun_Phi_ext(R, M);
11    double integral = 0;
12    double h = 1e-5;
13
14    // Partiamo a calcolare Phi dalla fine (r = R) perche' e' quando
15    // l'integrale e' piu' piccolo
16    for (int i = lenfile - 1; i > 0; i--){
17        integral += h / 2 * (fun_to_integrate(r[i], m[i], P[i])
18            + fun_to_integrate(r[i - 1], m[i - 1], P[i - 1]));
19        Phi[i] = Phi_ext + integral;
20    }

```

```

21
22 char Phi_int_filename[50];
23 sprintf(Pi_int_filename, "../data/Phi_int_%d.csv", tipo_politropica);
24 FILE *f_Phi_int = fopen(Pi_int_filename, "w");
25 fprintf(f_Phi_int, "r,Phi\n");
26 for (int i = 0; i < lenfile; i++)
27     fprintf(f_Phi_int, "%.10e,%.10e\n", r[i] * R0, Phi[i]);
28 fclose(f_Phi_int);
29
30 // Calcoliamo anche Phi_ext(r) per r > R
31 double r_ext = R;
32
33 char Phi_ext_filename[50];
34 sprintf(Pi_ext_filename, "../data/Phi_ext_%d.csv", tipo_politropica);
35 FILE *f_Phi_ext = fopen(Pi_ext_filename, "w");
36 fprintf(f_Phi_ext, "r,Phi\n");
37
38 while (r_ext < 150 / R0){
39     fprintf(f_Phi_ext, "%.10e,%.10e\n",
40         r_ext * R0, fun_Phi_ext(r_ext, M));
41     r_ext += h;
42 }
43
44 fclose(f_Phi_ext);
45 }

```

L'integrale (calcolato esplicitamente nella righe 17-18) può essere valutato solo nei punti r che sono stati utilizzati durante la risoluzione delle equazioni di stabilità della stella.

Per ottimizzare il codice, invece che calcolare l'intero integrale per ogni punto r del grafico di $\Phi(r)$, partiamo da $r = R$ (ovvero quando l'integrale è nullo) e aggiungiamo il valore di 1 trapezio per volta alla variabile `integral`. In contemporanea, durante una iterazione del ciclo `for` possiamo utilizzare il valore (parziale) di `integral` per calcolare Φ_{int} in r .

Infine dalla riga 30 calcoliamo Φ_{ext} utilizzando la funzione analitica.

D Integrali con trapezi e simpson

Le funzioni che fanno l'integrale con i due metodi sono

```

1 // Metodo Trapezi
2 double integrale_trapezio(double a, double b, int N, double T,
3                             double (*fun)(double, double)){
4     // assume a < b
5     double h = (b - a) / (double)N;
6     double integral = ((*fun)(a, T) + (*fun)(b, T)) * h / 2;
7
8     for (int i = 1; i < N; i++) {
9         integral += h * (*fun)(a + i * h, T);
10    }
11
12    return integral;
13 }
14
15 // Metodo Simpson
16 double integrale_simpson(double a, double b, int N, double T,
17                             double (*fun)(double, double)){
18     // assume a < b
19     double h = (b - a) / N;
20     double integral = ((*fun)(a, T) + (*fun)(b, T)) * h / 3;
21
22     // assume N pari
23     if (N % 2 != 0){
24         printf("N non e' pari");
25     }
26     for (int i = 1; i <= N/2 - 1; i++) {
27         integral += (2 * h / 3) * (*fun)(a + 2 * i * h, T);

```

```

28     integral += (4 * h / 3) * (*fun)(a + (2 * i - 1) * h, T);
29 }
30
31     integral += (4 * h / 3) * (*fun)(a + (N - 1) * h, T);
32
33     return integral;
34 }

```

La funzione da integrare (la radianza) è stata definita come

```

1 double funB(double nu, double T){
2     return pow(nu, 3) / (exp(nu / T) - 1.);
3 }

```

Per controllare la convergenza del valore della potenza usiamo inizialmente `nu_max` (ovvero $\hat{\nu}_{\max}$) uguale a 20. Prendiamo il caso in cui l'integrale è più piccolo e difficile da valutare con precisione, ovvero per $\hat{T} = 0.01$. Verifichiamo per quale `N` si ottiene un errore minore di quello voluto prima per i trapezi e poi per Simpson. I risultati vengono stampati sul terminale.

```

1 double T = 0.01;
2 int N = 10;
3 int N_trap, N_simp;
4 double Pot, nu_max = 20.;
5 double Pot_cvg = integrale_trapezio(1e-12, nu_max, 1e8, T, &funB);
6 double errore_max = 1e-7;
7 printf("Errore massimo scelto = %.0e\n", errore_max);
8 int kk = 0;
9
10 // Dati per grafico cvg per N trapezi
11 FILE *f0 = fopen("../data/potenza/test_cvg_N_trap.csv", "w");
12 fprintf(f0, "I,N,nu_max\n");
13 while(N <= 1e6){
14     Pot = integrale_trapezio(1e-12, nu_max, N, T, &funB);
15     fprintf(f0, "%.13e,%d,%.13e\n", Pot, N, nu_max * nu0);
16
17     if ((fabs(Pot - Pot_cvg) / Pot_cvg) < errore_max && kk == 0){
18         printf("Per trapezi N = %d e' sufficiente\n", N);
19         N_trap = N;
20         kk++;
21     }
22
23     N *= 1.1;
24     if (N % 2 != 0) N += 1;
25 }
26 fprintf(f0, "%.13e,%d,%.13e\n", Pot_cvg, (int)1e8, nu_max * nu0);
27 fclose(f0);
28
29 N = 10;
30 kk = 0;
31 // Dati per grafico cvg per N simpson
32 FILE *f1 = fopen("../data/potenza/test_cvg_N_simp.csv", "w");
33 fprintf(f1, "Pot,N,nu_max\n");
34 while(N <= 1e6){
35     Pot = integrale_simpson(1e-12, nu_max, N, T, &funB);
36     fprintf(f1, "%.13e,%d,%.13e\n", Pot, N, nu_max * nu0);
37
38     if ((fabs(Pot - Pot_cvg) / Pot_cvg) < errore_max && kk == 0){
39         printf("Per Simpson N = %d e' sufficiente\n", N);
40         N_simp = N;
41         kk++;
42     }
43 }
44
45 N *= 1.1;
46 if (N % 2 != 0) N += 1;
47 }

```

```

48 fprintf(f1, "%.13e,%d,%.13e\n", Pot_cvg, (int)1e8, nu_max * nu0);
49 fclose(f1);
50
51 // Decidiamo che va bene N_trap e N_simp risultati del codice sopra
52 // Ora verifichiamo la convergenza di nu_max nel caso peggiore, ovvero
53 T = 1;
54 Pot_cvg = integrale_trapezio(1e-12, 200, 1e8, T, &funB);
55 nu_max = 20.;
56 int Nrel_trap = (double)N_trap / nu_max; // Teniamo la stessa densita'
57 int Nrel_simp = (double)N_simp / nu_max; // di trapezi: N / nu_max
58
59 // partiamo da
60 nu_max = 10.;
61 kk = 0;
62 // Test cvg per A (ovvero nu_max)
63 FILE *f2 = fopen("../data/potenza/test_cvg_A_trap.csv", "w");
64 fprintf(f2, "Pot,N,nu_max[ad]\n");
65 while(nu_max <= 1e2){
66     N = Nrel_trap * nu_max;
67     if (N % 2 != 0) N += 1;
68
69     Pot = integrale_trapezio(1e-12, nu_max, N, T, &funB);
70     fprintf(f2, "%.13e,%d,%.13e\n", Pot, N, nu_max);
71
72     if ((fabs(Pot - Pot_cvg) / Pot_cvg) < errore_max && kk == 0){
73         printf("Per trapezi N = %d, nu_max = %.7f sono sufficienti\n",
74               N, nu_max);
75         kk++;
76     }
77
78     nu_max *= 1.05;
79 }
80 fprintf(f2, "%.13e,%d,%.13e\n", Pot_cvg, (int)1e8, 200.);
81 fclose(f2);
82
83 nu_max = 10.;
84 kk = 0;
85 // Test cvg per A (ovvero nu_max)
86 FILE *f3 = fopen("../data/potenza/test_cvg_A_simp.csv", "w");
87 fprintf(f2, "Pot,N,nu_max[ad]\n");
88 while(nu_max <= 1e2){
89     N = Nrel_simp * nu_max;
90     if (N % 2 != 0) N += 1;
91
92     Pot = integrale_simpson(1e-12, nu_max, N, T, &funB);
93     fprintf(f3, "%.13e,%d,%.13e\n", Pot, N, nu_max);
94
95     if ((fabs(Pot - Pot_cvg) / Pot_cvg) < errore_max && kk == 0){
96         printf("Per Simpson N = %d, nu_max = %.7f sono sufficienti\n",
97               N, nu_max);
98         kk++;
99     }
100
101     nu_max *= 1.05;
102 }
103 fprintf(f3, "%.13e,%d,%.13e\n", Pot_cvg, (int)1e8, 200.);
104 fclose(f3);

```

In stdout otteniamo

```

Errore massimo scelto = 1e-07
Per trapezi N = 19060 e' sufficiente
Per Simpson N = 25368 e' sufficiente
Per trapezi N = 22936, nu_max = 24.0661923 sono sufficienti
Per Simpson N = 30516, nu_max = 24.0661923 sono sufficienti

```

Dalla riga 51 viene utilizzato un codice molto simile a quello sopra controllare la convergenza per diversi valori di `nu_max`. L'unica differenza sta nel fatto che questa volta usiamo $\hat{T} = 1$, che è il caso in cui la funzione è più spostata verso destra e ci assicuriamo che la densità di trapezi N / nu_max rimanga costante al variare di `nu_max`.

E Calcolo Temperatura efficace

Per calcolare la temperatura efficace si riutilizzano molte delle funzioni scritte per l'appendice D. Per fare gli integrali possiamo affidarci ai parametri `N_trap` e `nu_max` trovati in D poiché sono stati studiati apposta per valere nel intervallo di temperature che va da 0.1 a 1.

Vista l'equivalenza dei due metodi di integrazione in quanto a precisione, ma non in quanto a costo computazionale, per questo calcolo utilizziamo solamente il metodo dei trapezi.

Il codice è il seguente:

```

1 #define PI 3.1415926535 // \pi
2
3 int N_trap = 22936;
4 double nu_max = 24.0661923;
5 double T_min = 0.01;
6 double T_max = 1;
7
8 double R[3] = {11.04289 / R0, 10.86752 / R0, 8.531525 / R0}; // Raggi stelle
9 double M[3] = {2.456841 / M0, 0.990100 / M0, 1.635845 / M0}; // Masse stelle
10
11
12 // ciclo sulle stelle
13 for (int i = 0; i < 3; i++){
14     char filename[50]; sprintf(filename, "../data/potenza/Teff_%d.csv", i + 1);
15     FILE *f = fopen(filename, "w");
16     fprintf(f, "T,Teff\n");
17
18     double T = T_min;
19     double Integrale, Teff;
20
21     while (T <= T_max){
22         Integrale = integrale_trapezio(1e-12, nu_max, N_trap, T, &funB);
23         Teff = pow(15, 1. / 4.) / PI * pow(1. - 2. * M[i] / R[i], 1. / 8.);
24         Teff *= pow(Integrale, 1. / 4.);
25         fprintf(f, "%.7e,%.7e\n", T, Teff);
26         T += 0.001;
27     }
28     fclose(f);
29 }

```