Numerical implementation of Neutron star cooling

Tae Geun Kim

Table of Contents

Table of Contents

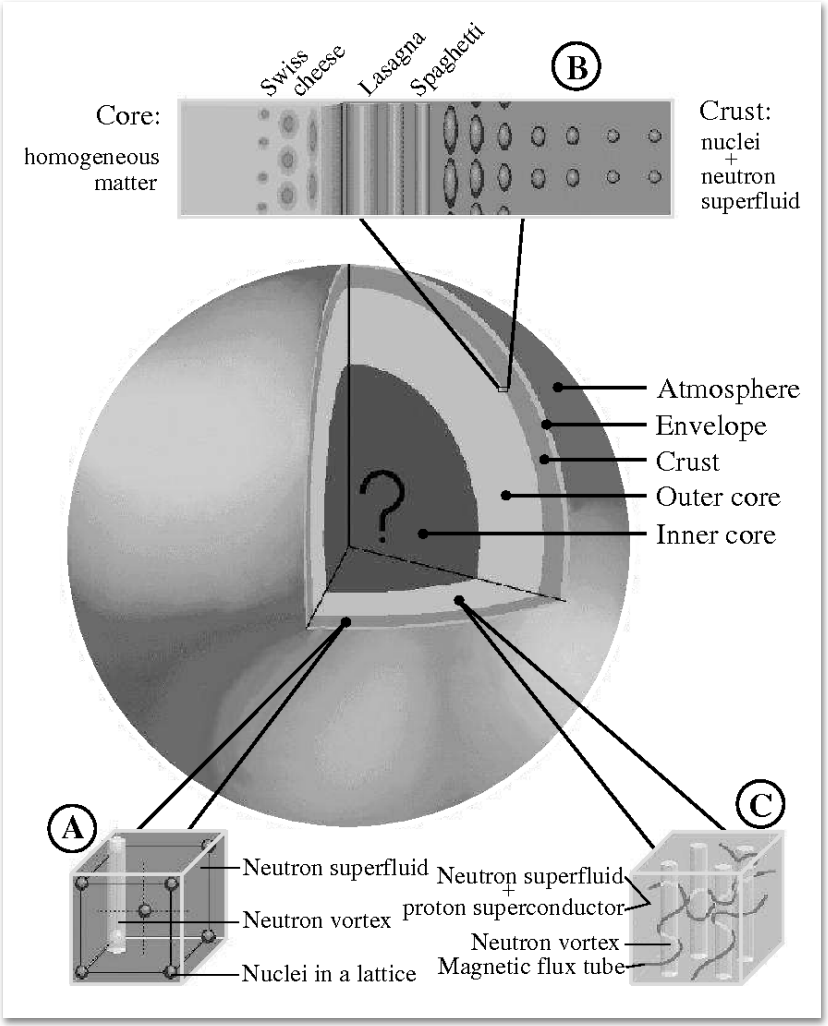
Table of Contents

Table of Contents

Table of Contents

Basic structure of Neutron star

Basic structure of Neutron star

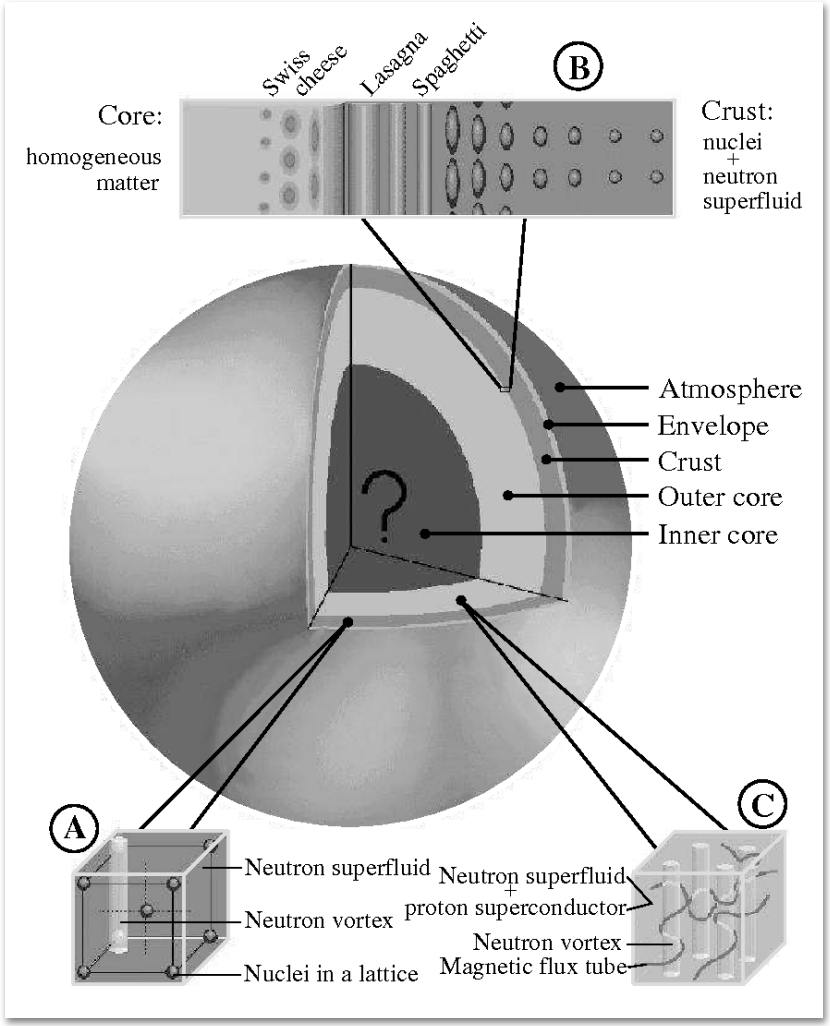


- **Atmosphere (10cm)**
  Determines the thermal radiation (the spectrum). Of upmost importance for interpretation of X-ray (and optical) observation. However it as NO effect on the thermal evolution of the star.

- **Envelope (100m)**
  Contains a huge temperature gradient - extremly important for the cooling, strongly affected by magnetic fields and the presence of *polluting* light elements

- **Crust (1km)**
  Composed of nuclei, quasi-free neutrons. The nuclei are arranged into a crystalline lattice. At bottom of crust, *pasta phases* of nuclear matter forms mantle.

Basic structure of Neutron star



- **Outer Core (10km)**

  Contains most of the stellar mass, composed of neutrons with an admixture of protons & leptons - electrons, muons ($npe\mu$ matter)

- **Inner Core (? km)**

  Exists for massive neutron star, compositions & properties are not well known. There are some proposed theoretical models:

1. **hyperonization of matter** – the appearance of various hyperons (first of all, Λ^- and Σ^- hyperons — ($npe\mu\Lambda\Sigma$)matter)
2. **pion condensation** – formation of a Bose condensate of collective interactions with the properties of π-mesons.
3. **kaon condensation** – formation of a similar condensate of K-mesons.

Basic structure of neutron star

- Before handling neutron star cooling directly, we should solve two sets of equations.

Basic structure of neutron star

- Before handling neutron star cooling directly, we should solve two sets of equations.
  - **Tolman-Oppenheimer-Volkoff equations**
    - Describe relativistic stellar structure

Basic structure of neutron star

- Before handling neutron star cooling directly, we should solve two sets of equations.
  - **Tolman-Oppenheimer-Volkoff equations**
    - Describe relativistic stellar structure
  - **Thermal evolution equations**
    - Describe stellar evolution

Relativistic stellar structure

Tolman-Oppenheimer-Volkoff Equation

- For static, non-rotating and spherical symmetric star, metric is given as

$$ds^2 = -e^{2\Phi(r)}dt^2 + e^{2\Lambda(r)}dr^2 + r^2 d\Omega^2$$

Tolman-Oppenheimer-Volkoff Equation

- For static, non-rotating and spherical symmetric star, metric is given as

$$ds^2 = -e^{2\Phi(r)}dt^2 + e^{2\Lambda(r)}dr^2 + r^2 d\Omega^2$$

- For isolated star, the metric must reduce to the Schwarzschild metric at outside of the star

$$ds^2 = -\left(1 - \frac{2M}{r}\right)dt^2 + \frac{1}{1 - \frac{2M}{r}}dr^2 + r^2 d\Omega^2$$

Tolman-Oppenheimer-Volkoff Equation

- For static, non-rotating and spherical symmetric star, metric is given as

$$ds^2 = -e^{2\Phi(r)}dt^2 + e^{2\Lambda(r)}dr^2 + r^2 d\Omega^2$$

- For isolated star, the metric must reduce to the Schwarzschild metric at outside of the star

$$ds^2 = -\left(1 - \frac{2M}{r}\right)dt^2 + \frac{1}{1 - \frac{2M}{r}}dr^2 + r^2 d\Omega^2$$

- At interior of the star, let define new metric function for convenience

$$e^{-2\Lambda(r)} = 1 - \frac{2m(r)}{r}$$

  The quantity $m(r)$ can be interpreted as the **mass interior to radius** r.

Tolman-Oppenheimer-Volkoff Equation

- Now, consider perfect fluid matter

$$T_{\mu\nu} = (\rho + P)u_\mu u_\nu + P g_{\mu\nu}, \quad u = e^{-\Phi}\frac{\partial}{\partial t}$$

Tolman-Oppenheimer-Volkoff Equation

- Now, consider perfect fluid matter

$$T_{\mu\nu} = (\rho + P)u_\mu u_\nu + P g_{\mu\nu}, \quad u = e^{-\Phi}\frac{\partial}{\partial t}$$

- Rewrite: (index-free notation)

$$T = \rho^{2\Phi} dt^2 + \frac{P}{1 - \frac{2m(r)}{r}} dr^2 + P r^2 d\Omega^2$$

Tolman-Oppenheimer-Volkoff Equation

- Now, consider perfect fluid matter

$$T_{\mu\nu} = (\rho + P)u_\mu u_\nu + P g_{\mu\nu}, \quad u = e^{-\Phi}\frac{\partial}{\partial t}$$

- Rewrite: (index-free notation)

$$T = \rho^{2\Phi} dt^2 + \frac{P}{1 - \frac{2m(r)}{r}} dr^2 + P r^2 d\Omega^2$$

- From Einstein equation, we can get two equations: Specific calculation with SageManifolds

$$(0,0) \Rightarrow \frac{dm(r)}{dr} = 4\pi r^2 \rho$$

$$(1,1) \Rightarrow \frac{d\Phi}{dr} = \frac{m(r) + 4\pi r^3 P}{r^2\left(1 - \frac{2m(r)}{r}\right)}$$

Tolman-Oppenheimer-Volkoff Equation

- From energy conservation, we can get one more equation:

$$\nabla_\mu T^\mu{}_\nu = 0 \;\Rightarrow\; (\rho + P)\frac{d\Phi}{dr} + \frac{dP}{dr} = 0$$

Tolman-Oppenheimer-Volkoff Equation

- From energy conservation, we can get one more equation:

$$\nabla_\mu T^\mu{}_\nu = 0 \;\Rightarrow\; (\rho + P)\frac{d\Phi}{dr} + \frac{dP}{dr} = 0$$

- These three fundamental equations are called **Tolman-Oppenheimer-Volkoff Equation**

$$\begin{aligned}
\frac{dm}{dr} &= 4\pi r^2 \rho \\
\frac{d\Phi}{dr} &= \frac{m + 4\pi r^3 P}{r^2\left(1 - \frac{2m}{r}\right)} \\
\frac{dP}{dr} &= -\frac{(\rho + P)(m + 4\pi r^3 P)}{r^2\left(1 - \frac{2m}{r}\right)}
\end{aligned}$$

Polytropic Equation of State

- Neutron stars are well modeled by polytropes

$$P = K\rho_0^{\Gamma}$$

where

K : polytropic gas constant
ρ_0 : rest mass density
$\rho_0 = m_u n_B$ (for neutron star)
m_u : nucleon mass - atomic mass unit
n_B : baryonic number density
$\rho = \rho_0(1+\epsilon)$
ϵ : internal energy density per unit mass
$\Gamma \equiv 1 + \frac{1}{n}$
n : polytropic index

Polytropic Equation of State

- By 1st law of thermodynamics with isentropic assumption, we can get

$$P = n_B^2 \frac{\partial(\rho/n_B)}{\partial n_B}$$

Polytropic Equation of State

- By 1st law of thermodynamics with isentropic assumption, we can get

$$P = n_B^2 \frac{\partial(\rho/n_B)}{\partial n_B}$$

- Substitute polytropic EoS:

$$\rho = \rho_0 + \frac{P}{\Gamma - 1} = \left(\frac{P}{K}\right)^{\frac{1}{\Gamma}} + \frac{\rho}{\Gamma - 1}$$

Solving TOV equation

- Total running terms

$$\frac{dm}{dr} = 4\pi r^2 \rho$$
$$\frac{dP}{dr} = -\frac{(\rho + P)(m + 4\pi r^3 P)}{r^2\left(1 - \frac{2m}{r}\right)}$$
$$\frac{d\Phi}{dr} = \frac{m + 4\pi r^3 P}{r^2\left(1 - \frac{2m}{r}\right)}$$
$$\frac{dN_B}{dr} = \frac{4\pi r^2 n_B}{\sqrt{1 - \frac{2m}{r}}}$$

$$\rho_0 = \left(\frac{P}{K}\right)^{\frac{1}{\Gamma}}$$
$$\rho = \rho_0 + \frac{P}{\Gamma - 1} = \left(\frac{P}{K}\right)^{\frac{1}{\Gamma}} + \frac{P}{\Gamma - 1}$$
$$n_B = \frac{\rho_0}{m_u} = \frac{1}{m_u}\left(\frac{P}{K}\right)^{\frac{1}{\Gamma}}$$

Solving TOV equation

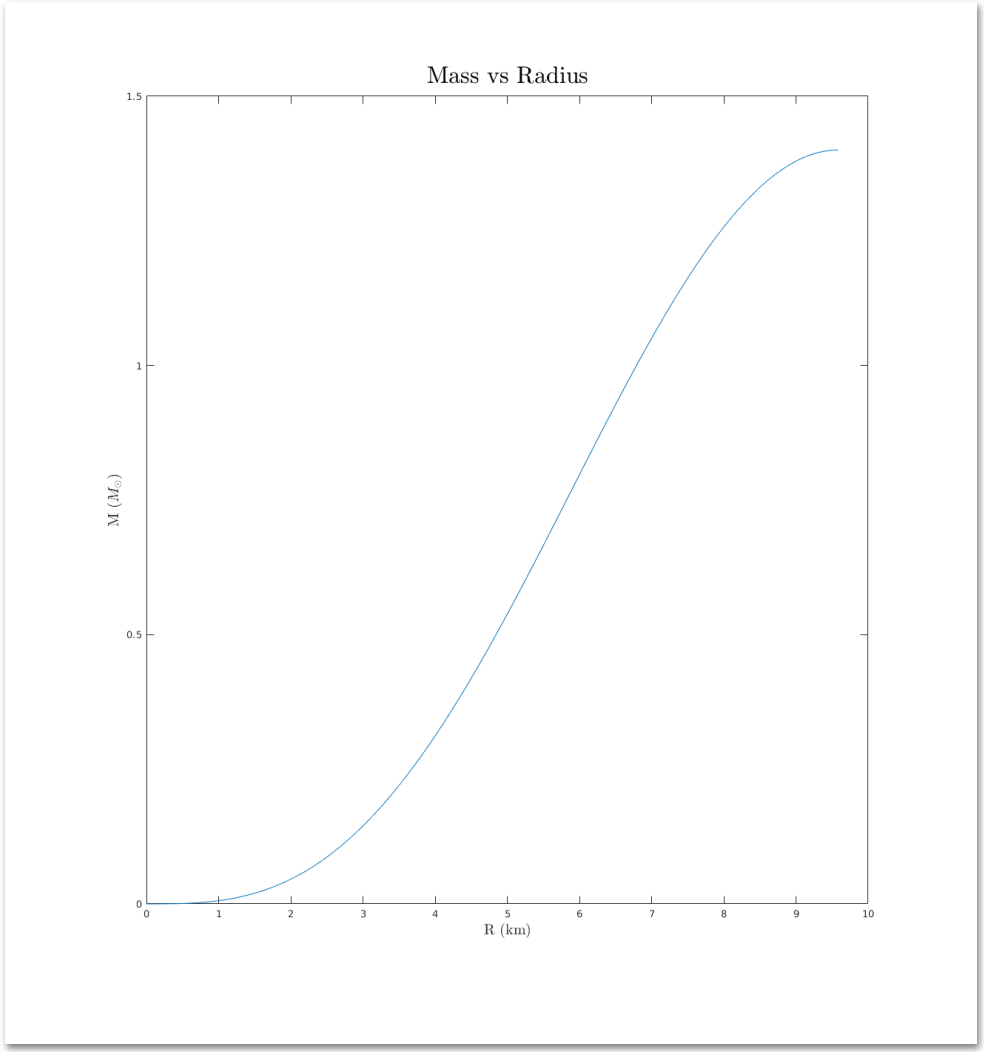
- Total boundary conditions

$$
\begin{aligned}
m(0) &= 0 \\
\rho_0(0) &= \rho_c \\
P(0) &= P_c = K\rho_c^{\Gamma} \\
n_B(0) &= n_c = \frac{\rho_c}{m_u}
\end{aligned}
$$

$$
\begin{aligned}
P(R) &= 0 \\
\Phi(R) &= \frac{1}{2}\ln\left(1 - \frac{2M}{R}\right)
\end{aligned}
$$

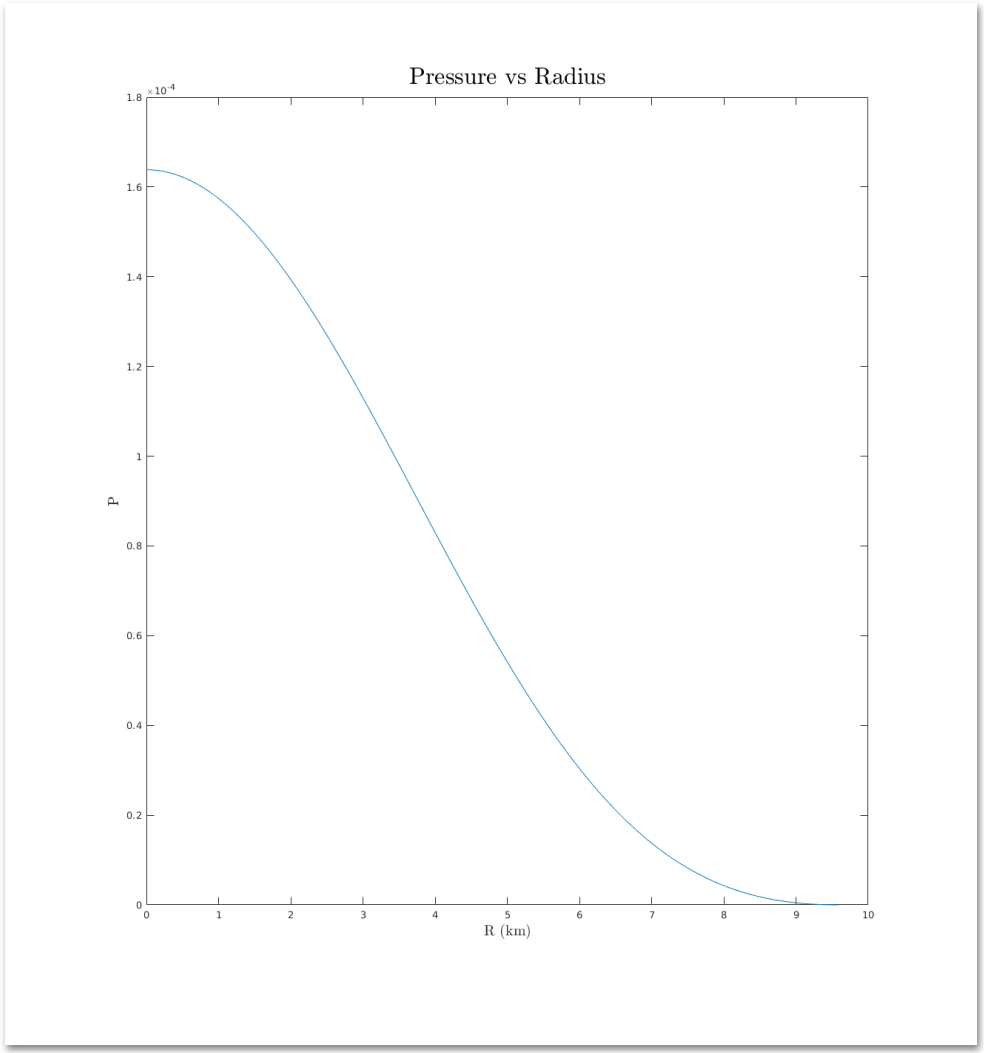
Results ($K = 100, \Gamma = 2, \rho_c = 1.28 \times 10^{-3}$)

$$R = 9.5850 km$$
$$M = 1.4002 M_\odot$$



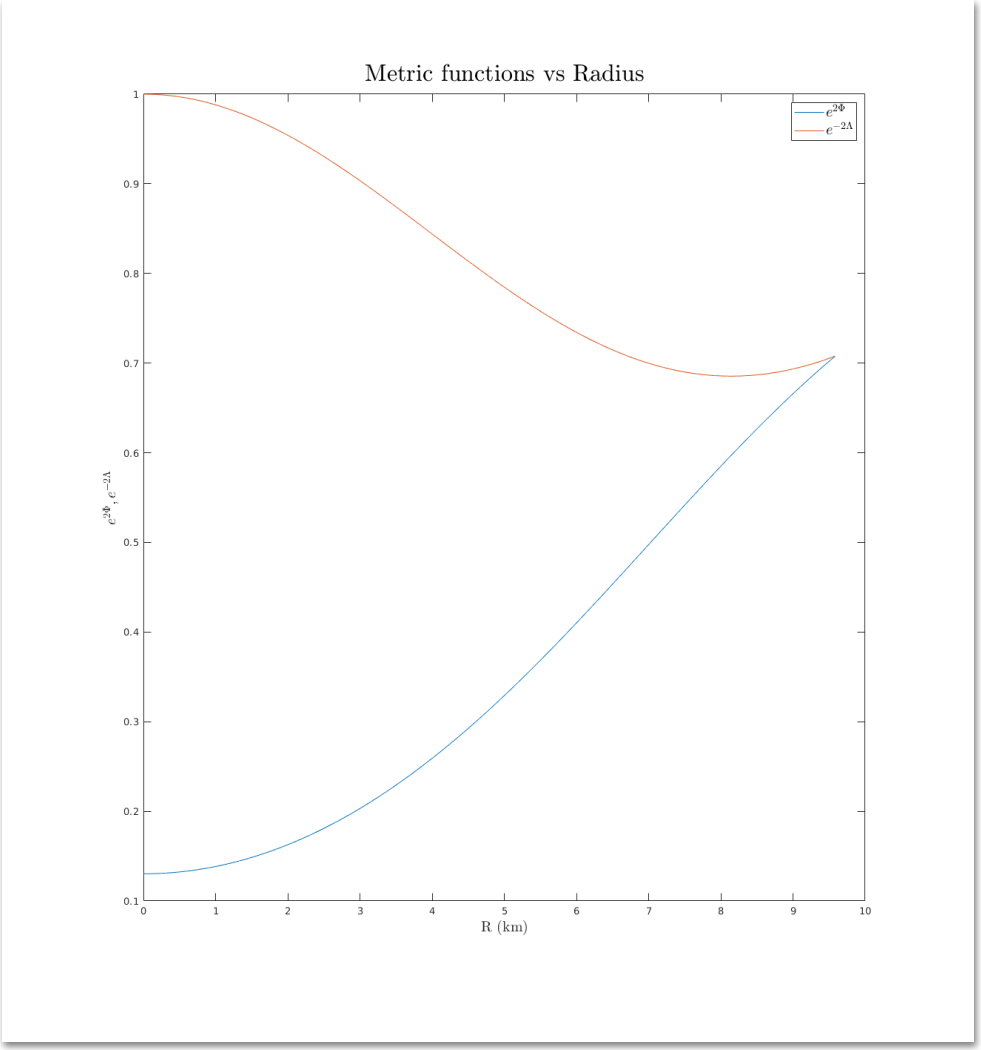
Results ($K = 100, \Gamma = 2, \rho_c = 1.28 \times 10^{-3}$)

$$P(0) = P_c, \quad P(R) = 0$$



Results ($K = 100, \Gamma = 2, \rho_c = 1.28 \times 10^{-3}$)

$$e^{2\Phi(R)} = e^{-2\Lambda(R)}$$



Implicit scheme to solve stiff ODE

Brief view - Thermal evolution equations

- Energy balance

$$\frac{d(Le^{2\Phi})}{dr} = -\frac{4\pi r^2 e^{\Phi}}{\sqrt{1-2m/r}}\left(C_v\frac{dT}{dt} + e^{\Phi}(Q_\nu - Q_h)\right)$$

Brief view - Thermal evolution equations

- Energy balance

$$\frac{d(Le^{2\Phi})}{dr} = -\frac{4\pi r^2 e^{\Phi}}{\sqrt{1-2m/r}}\left(C_v\frac{dT}{dt} + e^{\Phi}(Q_\nu - Q_h)\right)$$

- Energy transport

$$\frac{d(Te^{\Phi})}{dr} = -\frac{1}{\lambda}\frac{Le^{\Phi}}{4\pi r^2\sqrt{1-2m/r}}$$

Brief view - Thermal evolution equations

- Energy balance

$$\frac{d(Le^{2\Phi})}{dr} = -\frac{4\pi r^2 e^{\Phi}}{\sqrt{1-2m/r}}\left(C_v\frac{dT}{dt} + e^{\Phi}(Q_\nu - Q_h)\right)$$

- Energy transport

$$\frac{d(Te^{\Phi})}{dr} = -\frac{1}{\lambda}\frac{Le^{\Phi}}{4\pi r^2\sqrt{1-2m/r}}$$

where

- C_v: Specific heat
- L: Diffusive luminosity at r
- λ: Thermal conductivity
- Q_ν: Neutrino emissivity
- Q_h: Heating late

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{da}\right) \qquad \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{da}\right)$$

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{da}\right) \qquad \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{da}\right)$$

- But there is serious problem - **Numerically unstable**(stiff) unless step size is very small

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{da}\right) \qquad \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{da}\right)$$

- But there is serious problem - **Numerically unstable**(stiff) unless step size is very small
- We can't use explicit scheme - Euler, RK4, Verlet and etc.

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{da}\right) \qquad \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{da}\right)$$

- But there is serious problem - **Numerically unstable**(stiff) unless step size is very small
- We can't use explicit scheme - Euler, RK4, Verlet and etc.
- We should use implicit scheme - Backward Euler, DIRK, Rosenbrock-Krylov and etc.

Brief view - Thermal evolution equations

- Let's rewrite these equations with red-shifted functions & Lagrangian coordinate a

$$\mathcal{T} = e^{\Phi} T,\ \mathcal{L} = e^{2\Phi} L, \quad da = \frac{4\pi r^2 n_B dr}{\sqrt{1 - 2m/r^2}}$$

- Then we can get next two implicit equations

$$\frac{d\mathcal{T}}{dt} = F\left(\mathcal{T}, \frac{d\mathcal{L}}{da}\right) \qquad \mathcal{L} = G\left(\mathcal{T}, \frac{d\mathcal{T}}{da}\right)$$

- But there is serious problem - **Numerically unstable**(stiff) unless step size is very small
- We can't use explicit scheme - Euler, RK4, Verlet and etc.
- We should use implicit scheme - Backward Euler, DIRK, Rosenbrock-Krylov and etc.
- Our choice is **Backward Euler** - Most robust method (but high cost)

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.
  - Numerical method - Errors can do snowballing. (e.g. Approximate Jacobian)

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.
  - Numerical method - Errors can do snowballing. (e.g. Approximate Jacobian)
  - Modern regime - **Automatic Differentiation**

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.
  - Numerical method - Errors can do snowballing. (e.g. Approximate Jacobian)
  - Modern regime - **Automatic Differentiation**
    - Can find **exact** Jacobian very **fast**. - No error snowballing!

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.
  - Numerical method - Errors can do snowballing. (e.g. Approximate Jacobian)
  - Modern regime - **Automatic Differentiation**
    - Can find **exact** Jacobian very **fast**. - No error snowballing!
    - Actively used in field of Machine Learning

Brief view - Backward Euler

- Suppose that there is non-autonomous ODE system

$$\mathbf{Y}' = f(t, \mathbf{Y})$$

- One step of Backward Euler is as follow:

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + hf(t_n + h, \mathbf{Y}_{n+1})$$

- To find $\mathbf{Y}_{n+1}$, we should find root of above equation. There are some solutions.
  - Use CAS (like Mathematica) - If there are no algebraic roots, require very high costs.
  - Numerical method - Errors can do snowballing. (e.g. Approximate Jacobian)
  - Modern regime - **Automatic Differentiation**
    - Can find **exact** Jacobian very **fast**. - No error snowballing!
    - Actively used in field of Machine Learning

- I have achieved to reduce high cost of Backward Euler with AD & Rust (High performance programming language)

TODO

Current Progress

Symbolic

- Solve TOV from spherical symmetric metric & polytropic EM Tensor

Current Progress

Symbolic

- Solve TOV from spherical symmetric metric & polytropic EM Tensor

Numeric

- Implement Explicit RK4
- Apply ERK4 to TOV eq
- Implement Implicit Method
  - Backward Euler Method

TODO

Numeric

- Implement more implicit methods
  - 4th order Gauss-Legendre (GL4)
  - Diagnoal Implicit Runge-Kutta (DIRK)
  - Ronsenbrok-Krylov
- Provide precise and various benchmark

TODO

Numeric

- Implement more implicit methods
  - 4th order Gauss-Legendre (GL4)
  - Diagnoal Implicit Runge-Kutta (DIRK)
  - Ronsenbrok-Krylov
- Provide precise and various benchmark

Physics

- To formulate neutrino emissivity for several Equation of States & Core model
  - Focus on Cas A (Cassiopeia A)
    - Modified Urca
    - Bremstrahlung
    - Cooper pair breaking and formation (PBF)

TODO

Numeric

- Implement more implicit methods
  - 4th order Gauss-Legendre (GL4)
  - Diagnoal Implicit Runge-Kutta (DIRK)
  - Ronsenbrok-Krylov
- Provide precise and various benchmark

Physics

- To formulate neutrino emissivity for several Equation of States & Core model
  - Focus on Cas A (Cassiopeia A)
    - Modified Urca
    - Bremstrahlung
    - Cooper pair breaking and formation (PBF)

Interface

- Implement one-click (or type) installation for any OS
- Various language API (Python or even Mathematica?)

Thank you