thornado-transport: Anderson- and GPU-accelerated nonlinear solvers for neutrino-matter coupling[1]

Journal of Physics: Conference Series **1623** (2020) 012013 doi:10.1088/1742-6596/1623/1/012013

thornado-transport: Anderson- and GPU-accelerated nonlinear solvers for neutrino-matter coupling[1]

M Paul Laiu[1], J Austin Harris[2], Ran Chu[3] and Eirik Endeve[1,3]

[1]Computer Science and Mathematics Division, Oak Ridge National Laboratory, TN 37831
[2]National Center for Computational Sciences, Oak Ridge National Laboratory, TN 37831
[3]Department of Physics and Astronomy, University of Tennessee Knoxville, TN 37996

E-mail: laiump@ornl.gov; harrisja@ornl.gov; rchu@vols.utk.edu; endevee@ornl.gov

Abstract. Algorithms for neutrino-matter coupling in core-collapse supernovae (CCSNe) are investigated in the context of a spectral two-moment model, which is discretized in space with the discontinuous Galerkin method, integrated in time with implicit-explicit (IMEX) methods, and implemented in the **t**oolkit for **h**igh-**or**der **n**eutrino-r**ad**iation hydr**o**dynamics (`thornado`). The model considers electron neutrinos and antineutrinos and tabulated opacities from Bruenn (1985), which includes neutrino-electron scattering and pair processes. The nonlinear system arising from implicit time discretization of the equations governing neutrino-matter coupling is iterated to convergence using Anderson-accelerated fixed-point methods, which avoid formation of Jacobians and inversion of dense linear systems. Numerical experiments show that, for a given tolerance, a nested iteration scheme which aims to reduce opacity evaluations can lower the computational cost. Our initial port to GPUs, using both `OpenMP` and `OpenACC`, shows an overall speedup of up to $\sim$ 100x when compared to results using a single CPU core. These results indicate that the algorithms implemented in `thornado` are well-suited to GPU acceleration.

1. Introduction

Neutrinos play a critical role in the core-collapse supernova (CCSN) explosion mechanism [34, 27, 9, 37]. In a CCSN event, most ($\sim$ 99%) of the gravitational binding energy liberated during iron core collapse ($\sim 10^{46}$ J) is released as neutrino radiation, and it is thought that a fraction of this energy is deposited back into the stellar fluid to help drive the explosion. Although much remains to be elucidated, this basic picture is supported by simulations (e.g., [33, 31, 10]).

The 'supernova problem' has been with us since the 1960s [20, 4], and any perceived slow progress in understanding the explosion mechanism(s) is in part due to the complexity of the physical processes known to play a role, combined with a lacking availability of computational resources and robust and efficient algorithms needed to include these processes in models. The

[1] This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

inclusion of neutrino transport is a major reason for the high computational cost of CCSN models. Since neutrinos interact weakly with matter in the neutrino heating region, a kinetic description based on the phase space distribution function is warranted. The need for sufficient phase space resolution to faithfully capture the physics quickly adds to the computational cost. For example, it has been demonstrated (e.g., [30, 36]) that the inclusion of neutrino-matter interactions that couple across momentum space and/or across neutrino species (e.g., scattering and pair processes) can have a decisive impact on the global dynamics. However, the computational cost of including these interactions in simulations typically grows with the number of momentum space degrees of freedom to a power of 2-3 (depending on the algorithm), which can render them prohibitively expensive. The delivery of exascale computing systems (capable of 10^{18} double-precision floating-point operations per second) is expected to enable unprecedented realism in CCSN simulations. However, the path to the exascale — through heterogeneous computer architectures — has opened up unchartered territory for exploration of suitable algorithms, implementations, and programming models.

The **t**oolkit for **h**igh-**or**der **n**eutrino-r**ad**iation hydr**o**dynamics[2] (`thornado`) [13, 23] is being developed for modeling CCSNe. Current capabilities in `thornado` include hydrodynamics (approximated by the Euler equations with ideal and tabulated equations of state) and spectral neutrino transport (approximated by a two-moment model with algebraic closures). A distinguishing feature of `thornado`, when compared to mature supernova codes (e.g., [35, 8, 41]), is the use of high-order discontinuous Galerkin (DG) methods [17] for both hydrodynamics and neutrino transport. DG methods combine elements of spectral and finite volume methods, achieve high-order accuracy on a compact stencil, and favorable parallel scalability on heterogeneous architectures has been demonstrated [28]. They can be used in combination with *hp*-adaptivity [39], where, in addition to mesh refinement, the local polynomial degree can be chosen differently in different cells. DG methods can easily be adapted to problems using curvilinear coordinates (e.g., beneficial in numerical relativity [42]). Important for neutrino transport, DG methods recover without modification the correct asymptotic behavior in the diffusion limit, characterized by frequent collisions (e.g., [29, 1]). `thornado` is not developed as a supernova simulation code with all the utilities needed to run on large scale, distributed-memory systems, but rather as a collection of modules that can be incorporated into such simulation codes (e.g., `FLASH` [24] and `CASTRO` [2]). For this reason, a primary focus of the `thornado` development is node-level performance on heterogeneous computing systems.

In this paper we provide an initial overview of algorithms and implementations for neutrino-matter coupling in `thornado`. We discuss two iterative, nonlinear solvers embedded in our DG framework with implicit-explicit (IMEX) time integration: (1) a coupled algorithm, which iterates matter and neutrinos on an equal footing; and (2) a nested algorithm, which iterates matter and neutrinos in a hierarchical fashion with the aim of reducing expensive opacity evaluations. Both algorithms are based on Anderson-accelerated fixed-point iteration [3]. Numerical experiments using a single CPU core indicate that the nested algorithm can reduce the computational cost. Moreover, initial results from porting the coupled algorithm to GPUs show that a substantial reduction in time to solution can be achieved for all components of the neutrino transport scheme. Importantly, the implicit part of the DG-IMEX scheme achieves a $\sim$ 100x GPU speedup when compared to results obtained with a single CPU core. In the results presented, we considered electron neutrinos and antineutrinos, and used opacities from Bruenn (1985; [7]), which include emission/absorption, iso-energetic scattering, neutrino-electron scattering, and electron-positron pair processes. Tabulated equations of state and neutrino opacities, with associated subroutines, are provided by the `WeakLib`[3] library.

[2] github.com/endeve/thornado
[3] github.com/starkiller-astro/weaklib

2. Mathematical model

Here we present a model for electron neutrinos and antineutrinos interacting with a (fluid) background. Notable assumptions in our model are: (1) the fluid velocity is zero everywhere, (2) the rest mass density does not change with time, and (3) only the isotropic part of the scattering and pair process kernels are included. We consider a non-relativistic two-moment model for neutrino transport, where the spectral neutrino density $\mathcal{J}$, flux density $\boldsymbol{\mathcal{H}}$, and stress $\boldsymbol{\mathcal{K}}$ are defined as angular moments

$$\left\{ \mathcal{J}, \boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{K}} \right\}(\varepsilon, \boldsymbol{x}, t) = \frac{1}{4\pi} \int_{\mathbb{S}^2} f(\omega, \varepsilon, \boldsymbol{x}, t) \left\{ 1, \boldsymbol{\ell}, \boldsymbol{\ell} \otimes \boldsymbol{\ell} \right\} d\omega, \tag{1}$$

where the distribution function $f : (\omega, \varepsilon, \boldsymbol{x}, t) \in \mathbb{S}^2 \times \mathbb{R}^+ \times \mathbb{R}^3 \times \mathbb{R}^+ \to \mathbb{R}^+$ gives the number of particles propagating in the direction $\omega \in \mathbb{S}^2$, with energy $\varepsilon \in \mathbb{R}^+$, at position $\boldsymbol{x} \in \mathbb{R}^3$ and time $t \in \mathbb{R}^+$, and $\boldsymbol{\ell}(\omega) \in \mathbb{R}^3$ is a unit vector parallel to the particle propagation direction. (We employ units where the speed of light is $c = 1$.) Angular moments of the antineutrino distribution $\bar{f}(\omega, \varepsilon, \boldsymbol{x}, t)$ are defined analogously and annotated with a bar; i.e., $\bar{\mathcal{J}}$, $\bar{\boldsymbol{\mathcal{H}}}$, and $\bar{\boldsymbol{\mathcal{K}}}$.

For electron neutrinos, the evolution equations for the spectral density and flux are given by

$$\partial_t \mathcal{J} + \nabla \cdot \boldsymbol{\mathcal{H}} = \eta(\mathcal{J}, \bar{\mathcal{J}}) - \chi(\mathcal{J}, \bar{\mathcal{J}})\, \mathcal{J}, \tag{2}$$

$$\partial_t \boldsymbol{\mathcal{H}} + \nabla \cdot \boldsymbol{\mathcal{K}} = -\kappa(\mathcal{J}, \bar{\mathcal{J}})\, \boldsymbol{\mathcal{H}}, \tag{3}$$

where, following [32], we write the stress tensor in terms of the lower order moments as

$$\boldsymbol{\mathcal{K}} = \frac{1}{2} \left[\left(1 - \psi\right) \boldsymbol{I} + \left(3\,\psi - 1\right) \widehat{\boldsymbol{h}} \otimes \widehat{\boldsymbol{h}} \right] \mathcal{J}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix, $\widehat{\boldsymbol{h}} = \boldsymbol{\mathcal{H}}/|\boldsymbol{\mathcal{H}}|$, and $\psi(\mathcal{J}, h)$, with $h = |\boldsymbol{\mathcal{H}}|/\mathcal{J}$, is the Eddington factor. Here we use the maximum entropy Eddington factor from Cernohorsky & Bludman [11].

The collision terms on the right-hand side of Eqs. (2) and (3) model neutrino-matter interactions due to electron capture and iso-energetic scattering on nucleons and nuclei, inelastic neutrino-electron scattering (NES), and electron-positron pair production and annihilation. The total emissivity is given by

$$\eta(\mathcal{J}, \bar{\mathcal{J}}) = \chi_{\text{Ec}}\, \mathcal{J}_0 + \int_{\mathbb{R}^+} \Phi^{\text{In}}_{\text{NES}}(\varepsilon, \varepsilon')\, \mathcal{J}(\varepsilon')\, dV_{\varepsilon'} + \int_{\mathbb{R}^+} \Phi^{\text{Pr}}_{\text{Pair}}(\varepsilon, \varepsilon') \left(1 - \bar{\mathcal{J}}(\varepsilon')\right) dV_{\varepsilon'}, \tag{5}$$

where χ_{Ec} is the electron capture opacity, $\mathcal{J}_0$ is the equilibrium (Fermi-Dirac) spectral distribution, $\Phi^{\text{In}}_{\text{NES}}$ is the NES in-scattering kernel, and $\Phi^{\text{Pr}}_{\text{Pair}}$ is the pair production kernel. The infinitesimal energy volume element is $dV_\varepsilon = \varepsilon^2 d\varepsilon$. The total opacity is given by

$$\begin{aligned} \chi(\mathcal{J}, \bar{\mathcal{J}}) = \chi_{\text{Ec}} &+ \int_{\mathbb{R}^+} \left[\Phi^{\text{In}}_{\text{NES}}(\varepsilon, \varepsilon')\, \mathcal{J}(\varepsilon') + \Phi^{\text{Out}}_{\text{NES}}(\varepsilon, \varepsilon') \left(1 - \mathcal{J}(\varepsilon')\right) \right] dV_{\varepsilon'} \\ &+ \int_{\mathbb{R}^+} \left[\Phi^{\text{Pr}}_{\text{Pair}}(\varepsilon, \varepsilon') \left(1 - \bar{\mathcal{J}}(\varepsilon')\right) + \Phi^{\text{An}}_{\text{Pair}}(\varepsilon, \varepsilon')\, \bar{\mathcal{J}}(\varepsilon') \right] dV_{\varepsilon'}, \end{aligned} \tag{6}$$

where $\Phi^{\text{Out}}_{\text{NES}}$ is the NES out-scattering kernel, and $\Phi^{\text{An}}_{\text{Pair}}$ is the pair annihilation kernel. In Eq. (3), $\kappa = \chi + \sigma$, where σ — independent of $\mathcal{J}$ and $\bar{\mathcal{J}}$ — is the scattering opacity due to iso-energetic scattering on nucleons and nuclei. (For brevity we omit listing of analogous equations for antineutrinos.) Here we represent the iso-energetic and NES scattering, and the pair kernels by their isotropic approximations (zeroth order Legendre expansion of the full kernels; cf. [7]). The opacities χ_{Ec} and σ, the equilibrium distribution $\mathcal{J}_0$, and the kernels $\Phi^{\text{In}}_{\text{NES}}$, $\Phi^{\text{Out}}_{\text{NES}}$, $\Phi^{\text{Pr}}_{\text{Pair}}$, and $\Phi^{\text{An}}_{\text{Pair}}$ depend nonlinearly on the neutrino energy ε and local matter conditions; e.g., rest mass

density ρ, temperature T, and electron fraction Y_e. The integral operators in Eqs. (5) and (6) couple neutrinos of all energies. In addition, pair production and annihilation couple neutrinos and antineutrinos.

We denote the evolved quantities and fluxes for neutrinos and antineutrinos compactly as

$$\boldsymbol{\mathcal{M}} = \left(\mathcal{J}, \boldsymbol{\mathcal{H}}, \bar{\mathcal{J}}, \bar{\boldsymbol{\mathcal{H}}} \right) \quad \text{and} \quad \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{M}}) = \left(\boldsymbol{\mathcal{H}}, \boldsymbol{\mathcal{K}}, \bar{\boldsymbol{\mathcal{H}}}, \bar{\boldsymbol{\mathcal{K}}} \right), \tag{7}$$

respectively, and the collision term as

$$\boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{M}}) = \left(\eta - \chi\, \mathcal{J}, -\kappa\, \boldsymbol{\mathcal{H}}, \bar{\eta} - \bar{\chi}\, \bar{\mathcal{J}}, -\bar{\kappa}\, \bar{\boldsymbol{\mathcal{H}}} \right), \tag{8}$$

and write the full system of moment equations as

$$\partial_t \boldsymbol{\mathcal{M}} + \nabla \cdot \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{M}}) = \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{M}}). \tag{9}$$

Neutrino-matter interactions result in lepton and four-momentum exchange between neutrinos and matter. The resulting change in the electron fraction is given by

$$\partial_t Y_e = -\left(\frac{m_{\mathrm{B}}}{\rho} \right) \left(\frac{4\pi}{h^3} \right) \int_{\mathbb{R}^+} \left[\left(\eta - \chi\, \mathcal{J} \right) - \left(\bar{\eta} - \bar{\chi}\, \bar{\mathcal{J}} \right) \right] dV_\varepsilon, \tag{10}$$

where m_{B} is the average baryon mass and h is the Planck constant. The change in the specific internal energy ϵ is given by

$$\partial_t \epsilon = -\left(\frac{1}{\rho} \right) \left(\frac{4\pi}{h^3} \right) \int_{\mathbb{R}^+} \left[\left(\eta - \chi\, \mathcal{J} \right) + \left(\bar{\eta} - \bar{\chi}\, \bar{\mathcal{J}} \right) \right] \varepsilon\, dV_\varepsilon. \tag{11}$$

Combining evolution equations for neutrinos and antineutrinos (cf. Eq. (2)) with Eq. (10), lepton number conservation is expressed as

$$\int_D \left[\left(\frac{\rho}{m_{\mathrm{B}}} \right) \partial_t Y_e + \frac{4\pi}{h^3} \int_{\mathbb{R}^+} \left(\partial_t \mathcal{J} - \partial_t \bar{\mathcal{J}} \right) dV_\varepsilon \right] d\boldsymbol{x} = - \oint_{\partial D} \left[\frac{4\pi}{h^3} \int_{\mathbb{R}^+} \left(\boldsymbol{\mathcal{H}} - \bar{\boldsymbol{\mathcal{H}}} \right) dV_\varepsilon \right] \cdot d\boldsymbol{S}, \tag{12}$$

where the change in the total lepton number in the domain D (left-hand side of Eq. (12)) is only due to fluxes through the boundary ∂D. Similarly, the change in the total energy is

$$\int_D \left[\rho \partial_t \epsilon + \frac{4\pi}{h^3} \int_{\mathbb{R}^+} \left(\partial_t \mathcal{J} + \partial_t \bar{\mathcal{J}} \right) \varepsilon\, dV_\varepsilon \right] d\boldsymbol{x} = - \oint_{\partial D} \left[\frac{4\pi}{h^3} \int_{\mathbb{R}^+} \left(\boldsymbol{\mathcal{H}} + \bar{\boldsymbol{\mathcal{H}}} \right) \varepsilon\, dV_\varepsilon \right] \cdot d\boldsymbol{S}. \tag{13}$$

(Our model imposes a static background, therefore the three-momentum is not conserved.)

3. DG-IMEX scheme

We use the DG method [14, 15, 16, 17, 18, 19] to discretize Eq. (9) in $\boldsymbol{z} = (\varepsilon, \boldsymbol{x})$ combined with IMEX time discretization [5, 38]. Here we summarize the main components of this DG-IMEX scheme. While we defer the full exposition of the current scheme to a future publication, details on our scheme for a single energy, single particle system, using a simplified collision term can be found in [12],

We divide the $d_{\boldsymbol{z}}$-dimensional[4] phase-space domain D into a disjoint union $\mathcal{T}$ of open elements $\boldsymbol{K}_g = \boldsymbol{K}^{\boldsymbol{x}} \otimes K_g^\varepsilon$, so that $D = \cup_{\boldsymbol{K}_g \in \mathcal{T}} \boldsymbol{K}_g$. The spatial elements are denoted $\boldsymbol{K}^{\boldsymbol{x}} = \left\{ \boldsymbol{x} : x^i \in K^i := (x^i_{\mathrm{L}}, x^i_{\mathrm{H}}) | i = 1, \ldots, d_{\boldsymbol{x}} \right\}$, while the finite energy space D^ε is divided into N_g elements (groups) denoted $K_g^\varepsilon := (\varepsilon_{g-1/2}, \varepsilon_{g+1/2})$, $g = 1, \ldots, N_g$, so that $D^\varepsilon = \cup_{g=1}^{N_g} K_g^\varepsilon$.

[4] We let $d_{\boldsymbol{z}} = d_{\boldsymbol{x}} + 1$ denote the dimension of energy-position space and $d_{\boldsymbol{x}}$ the dimension of position space.

We denote the approximation space for the DG method as $\mathbb{V}_{\mathtt{h}}^k$, which is consist of functions that, on each element in $\boldsymbol{K}_g$, take values from tensor products of one-dimensional polynomials of maximal degree k. The semi-discrete DG problem is then to find $\boldsymbol{\mathcal{M}}_{\mathtt{h}} \in \mathbb{V}_{\mathtt{h}}^k$ such that

$$\partial_t \int_{\boldsymbol{K}_g} \boldsymbol{\mathcal{M}}_{\mathtt{h}}\, v\, dV = -\oint_{\partial \boldsymbol{K}_g} v\, \widehat{\boldsymbol{\mathcal{F}}}(\boldsymbol{\mathcal{M}}_{\mathtt{h}}) \cdot \boldsymbol{n}\, dA + \int_{\boldsymbol{K}_g} \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{M}}_{\mathtt{h}}) \cdot \nabla v\, dV + \int_{\boldsymbol{K}_g} \boldsymbol{\mathcal{C}}(\boldsymbol{\mathcal{M}}_{\mathtt{h}})\, v\, dV, \quad (14)$$

for all $v \in \mathbb{V}_{\mathtt{h}}^k$ and all $\boldsymbol{K}_g \in \mathcal{T}$. Here the subscript $\mathtt{h}$ denotes that $\boldsymbol{\mathcal{M}}_{\mathtt{h}}$ is a discretized approximation of $\boldsymbol{\mathcal{M}}$, since functions in $\mathbb{V}_{\mathtt{h}}^k$ are allowed to be discontinuous at edges of the elements. In Eq. (14), $\boldsymbol{n}$ is the outward normal on the boundary $\partial \boldsymbol{K}_g$ (area element dA) of element $\boldsymbol{K}_g$ (volume element dV), and $\widehat{\boldsymbol{\mathcal{F}}}$ is the numerical flux. We use the global Lax-Friedrichs flux with unit wave speed in the numerical experiments presented in Section 5.

In Eq. (14), we use Lagrange polynomials (i.e., nodal DG [25]) based on the tensor product of one-dimensional Legendre-Gauss (LG) quadrature points to represent $\boldsymbol{\mathcal{M}}_{\mathtt{h}}$ and v in $\boldsymbol{K}_g$. We denote the N-point one-dimensional LG point set by $S_N^i = \{\eta_q^i\}_{q=1}^N$ so that the point set for the Lagrange polynomial basis is $S_N = \otimes_{i=1}^{d_{\boldsymbol{z}}} S_N^i$. We define the point set on the element face with normal in the ith dimension as $\tilde{S}_N^i = \otimes_{j\neq i}^{d_{\boldsymbol{z}}} S_N^j$. When evaluating volume and surface integrals in Eq. (14) we use the LG quadrature rule with the S_N and $\tilde{S}_N^i$ points (and associated weights), respectively; i.e., similar to the spectral-type nodal collocation DG method in [6].

The semi-discretization of the moment equations in Eq. (14) results in a system of ordinary differential equations (ODEs) of the form

$$d_t \mathbf{M} = \mathbf{T}_{\mathbf{M}}(\mathbf{M}) + \mathbf{C}_{\mathbf{M}}(\mathbf{M}), \quad (15)$$

where $\mathbf{M} = \left\{\mathbf{M}_{\boldsymbol{K}_g}\right\}_{\boldsymbol{K}_g \in \mathcal{T}}$ represent all the degrees of freedom evolved with the DG method, and $\mathbf{M}_{\boldsymbol{K}_g}$ are the unknowns in element $\boldsymbol{K}_g$ (i.e., $\boldsymbol{\mathcal{M}}_{\mathtt{h}}$ evaluated in the point set S_N). On the right-hand side of Eq. (15), the transport operator $\mathbf{T}_{\mathbf{M}}$ corresponds to the first two terms on the right-hand side of Eq. (14), while the collision operator $\mathbf{C}_{\mathbf{M}}$ corresponds to the last term. Coupling Eq. (15) with the electron fraction and internal energy equations, Eqs. (10) and (11), gives

$$d_t \mathbf{U} = \mathbf{T}(\mathbf{U}) + \mathbf{C}(\mathbf{U}), \quad (16)$$

where $\mathbf{U} = (\mathbf{u}, \mathbf{M})$, with $\mathbf{u}$ denoting (Y_e, ϵ) in Eqs. (10)–(11). Here the transport operator $\mathbf{T}(\mathbf{U}) = (\mathbf{0}, \mathbf{T}_{\mathbf{M}}(\mathbf{M}))$, and the collision operator $\mathbf{C}$ is given by the right-hand sides of Eqs. (10)–(11) and $\mathbf{C}_{\mathbf{M}}$.

Since neutrino-matter interactions are frequent and introduce stiffness in the dense regions of the computational domain, we use IMEX time integration to stably integrate with time steps that can greatly exceed the shortest time scales associated with the collision operator using implicit methods, while we integrate the transport operator using explicit methods. s-stage IMEX schemes can be written in the following general form [38]

$$\mathbf{U}^{(i)} = \mathbf{U}^n + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{ij}\, \mathbf{T}(\mathbf{U}^{(j)}) + \Delta t \sum_{j=1}^{i} a_{ij}\, \mathbf{C}(\mathbf{U}^{(j)}), \quad i = 1, \ldots, s, \quad (17)$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t \sum_{i=1}^{s} \tilde{w}_i\, \mathbf{T}(\mathbf{U}^{(i)}) + \Delta t \sum_{i=1}^{s} w_i\, \mathbf{C}(\mathbf{U}^{(i)}), \quad (18)$$

where the matrices $(\tilde{a}_{ij}), (a_{ij}) \in \mathbb{R}^{s\times s}$ and vectors $(\tilde{w}_i), (w_i) \in \mathbb{R}^s$ must satisfy certain order conditions for accuracy. The explicit matrix $(\tilde{a}_{ij})$ is strictly lower triangular, while the implicit matrix (a_{ij}) is lower triangular, i.e., the diagonal entries a_{ii} can be nonzero. We use the PD-ARS scheme from [12] (their Appendix A with $\epsilon = 0$), which combines two evaluations of $\mathbf{T}$

and $\mathbf{C}$ (i.e., two implicit solves) per time step. The scheme is formally only first-order accurate in time, but reduces to the optimal second-order accurate explicit Runge-Kutta scheme of [40] in the streaming limit ($\mathbf{C} = 0$), performs well in the diffusion limit, and is convex-invariant. Since neutrinos obey Fermi-Dirac statistics, their distribution function is governed by lower and upper bounds (e.g., $f, \bar{f} \in [0, 1]$; a convex set). As a consequence, the moments of the neutrino distribution function must satisfy certain algebraic constraints, which also form a convex set. Given sufficient conditions, a convex-invariant time integration scheme preserves the constraints of a model if the set of admissible states satisfying the constraints forms a convex set. When applying the PD-ARS scheme from [12] to the two-moment model, sufficient conditions to ensure the cell-averaged moments satisfy the constraints at t^{n+1} are: (1) a restriction on the time step, which comes from the explicit part; (2) that the moments satisfy the constraints in a discrete set of points in each element; and (3) that the moment closure is consistent with Fermi-Dirac statistics. The realizability-enforcing limiter described in [12] is applied in the stages of the IMEX scheme to enforce the constraints point-wise in each element (see [12] for complete details).

Note that each stage of the IMEX scheme in Eq. (17) can be written as

$$\mathbf{U}^{(i)} = \mathbf{U}^{(*)} + a_{ii}\,\Delta t\,\mathbf{C}(\mathbf{U}^{(i)}), \quad \text{where} \quad \mathbf{U}^{(*)} = \mathbf{U}^{n} + \Delta t \sum_{j=1}^{i-1} \Big(\tilde{a}_{ij}\,\mathbf{T}(\mathbf{U}^{(j)}) + a_{ij}\,\mathbf{C}(\mathbf{U}^{(j)}) \Big), \tag{19}$$

which is equivalent to a backward Euler solve with time step $a_{ii}\,\Delta t$ ($a_{ii} \geq 0$). We proceed to discuss approaches to the implicit solve in Eq. (19), and the coupling to the matter equations.

4. Nonlinear solvers

In this section, considering two neutrino species (electron neutrinos and antineutrinos), we write the implicit solve in Eq. (19) as

$$\mathbf{U}^{(+)} = \mathbf{U}^{(*)} + \tau\,\mathbf{C}(\mathbf{U}^{(+)})\,, \tag{20}$$

where $\mathbf{U}^{(*)}$ is given by the explicit part in Eq. (19), τ denotes the effective time step, and $\mathbf{U}^{(+)}$ represents the new state from the implicit update. In Eq. (20), $\mathbf{U}^{(*)}$ is considered a known constant vector, while $\mathbf{U}^{(*)}$ is the unknown to be solved. Since the collision operator $\mathbf{C}$ does not introduce coupling in position $\boldsymbol{x}$, Eq. (20) can be split into $(N \times N_{\boldsymbol{x}})^{d_{\boldsymbol{x}}}$ decoupled systems of equations, each of which governs $2 + 2 \times N \times N_g$ unknowns in a given spatial node. Let $(\mathsf{Y}_e, \epsilon, \mathsf{J}, \mathbf{H}, \bar{\mathsf{J}}, \bar{\mathbf{H}})$ denote the states $\mathbf{U}$ at one spatial node, i.e., $(\mathsf{Y}_e, \epsilon) \in \mathbb{R}^2$ denotes the values of (Y_e, ϵ) at the given spatial node, and $(\mathsf{J}, \mathbf{H}, \bar{\mathsf{J}}, \bar{\mathbf{H}}) \in \mathbb{R}^{2NN_g}$ collects values of moments $(\mathcal{J}, \boldsymbol{\mathcal{H}}, \bar{\mathcal{J}}, \bar{\boldsymbol{\mathcal{H}}})$ on all energy nodes at the given spatial node. Then, at each spatial node, Eq. (20) can be explicitly written as

$$\mathsf{Y}_e^{(+)} = \mathsf{Y}_e^{(*)} - \tau\big(\tfrac{m_{\mathrm{B}}}{\rho}\big)\big(\tfrac{4\pi}{h^3}\big)\textstyle\sum_q W_q^{(2)}\big((\eta - \chi\,\mathsf{J}_q^{(+)}) - (\bar{\eta} - \bar{\chi}\,\bar{\mathsf{J}}_q^{(+)})\big)\,, \tag{21a}$$

$$\epsilon^{(+)} = \epsilon^{(*)} - \tau\big(\tfrac{1}{\rho}\big)\big(\tfrac{4\pi}{h^3}\big)\textstyle\sum_q W_q^{(3)}\big((\eta - \chi\,\mathsf{J}_q^{(+)}) + (\bar{\eta} - \bar{\chi}\,\bar{\mathsf{J}}_q^{(+)})\big)\,, \tag{21b}$$

$$\mathsf{J}^{(+)} = \mathsf{J}^{(*)} + \tau\big(\eta - \chi\,\mathsf{J}^{(+)}\big)\,, \tag{21c}$$

$$\bar{\mathsf{J}}^{(+)} = \bar{\mathsf{J}}^{(*)} + \tau\big(\bar{\eta} - \bar{\chi}\,\bar{\mathsf{J}}^{(+)}\big)\,, \tag{21d}$$

$$\mathbf{H}^{(+)} = \mathbf{H}^{(*)} - \tau\,\kappa\,\mathbf{H}^{(+)}\,, \tag{21e}$$

$$\bar{\mathbf{H}}^{(+)} = \bar{\mathbf{H}}^{(*)} - \tau\,\bar{\kappa}\,\bar{\mathbf{H}}^{(+)}\,, \tag{21f}$$

where $(\mathsf{J}_q, \bar{\mathsf{J}}_q)$ denotes the number densities at energy node $\varepsilon_q \in D^{\varepsilon}$, given by the N-point LG quadrature on each energy element (group) K_g^{ε} as defined in section 3. The weights $W_q^{(2)}$ and

$W_q^{(3)}$ correspond to the quadrature weights used to evaluate the energy integrals in Eqs. (10)–(11). Here the discrete opacities η and χ are computed by evaluating the energy integrals in Eqs. (5) and (6) using the energy quadrature, and $\kappa := \chi + \sigma$, with σ evaluated in the same quadrature points as χ. The antineutrino opacities $\bar{\eta}$, $\bar{\chi}$, and $\bar{\kappa}$ are defined analogously.

We take a two-step approach to solving Eq. (21), which first solves the fully coupled nonlinear system (21a)–(21d), plug the solution $(\mathsf{Y}_e^{(+)}, \epsilon^{(+)}, \mathsf{J}^{(+)}, \bar{\mathsf{J}}^{(+)})$ into (21e)–(21f) to compute κ, $\bar{\kappa}$, and then update $(\mathbf{H}^{(+)}, \bar{\mathbf{H}}^{(+)})$. Since solving (21e)–(21f) is straightforward once κ and $\bar{\kappa}$ are known, the main difficulty in solving (21) is to solve (21a)–(21d), where the opacities $(\eta, \chi, \bar{\eta}, \bar{\chi})$ are functions of $(\mathsf{Y}_e, \epsilon, \mathsf{J}, \bar{\mathsf{J}})$ (see Eqs. (5), (6)) and these opacities need to be updated in the solution procedure in order to remain consistent with $(\mathsf{Y}_e^{(+)}, \epsilon^{(+)}, \mathsf{J}^{(+)}, \bar{\mathsf{J}}^{(+)})$.

In the following subsections, we investigate two approaches for solving the coupled nonlinear system (21a)–(21d). To simplify the notation, we denote the matter states by $\mathbf{u} := (\mathsf{Y}_e^{(+)}, \epsilon^{(+)})$, the discretized neutrino (antineutrino) distributions by $\boldsymbol{\mathcal{U}} := (\mathsf{J}^{(+)}, \bar{\mathsf{J}}^{(+)})$, and the total variables by $\mathbf{U} := (\mathbf{u}, \boldsymbol{\mathcal{U}})$. We then write (21a)–(21d) as

$$\mathbf{U} = \mathbf{G}(\mathbf{U}) = \begin{pmatrix} \mathbf{g}(\boldsymbol{\mathcal{U}}) \\ \boldsymbol{\mathcal{G}}(\mathbf{u}, \boldsymbol{\mathcal{U}}) \end{pmatrix}, \tag{22}$$

where

$$\mathbf{g}(\boldsymbol{\mathcal{U}}) := \begin{pmatrix} -\left(\frac{m_{\mathrm{B}}}{\rho}\right)\left(\frac{4\pi}{h^3}\right)\sum_q W_q^{(2)}\left(\mathsf{J}_q^{(+)} - \bar{\mathsf{J}}_q^{(+)}\right) + C_{\mathsf{Y}_e} \\ -\left(\frac{1}{\rho}\right)\left(\frac{4\pi}{h^3}\right)\sum_q W_q^{(3)}\left(\mathsf{J}_q^{(+)} + \bar{\mathsf{J}}_q^{(+)}\right) + C_{\epsilon} \end{pmatrix}, \tag{23}$$

and

$$\boldsymbol{\mathcal{G}}(\mathbf{u}, \boldsymbol{\mathcal{U}}) := \begin{pmatrix} \left(\mathsf{J}^{(*)} + \tau\,\eta(\mathbf{u}, \boldsymbol{\mathcal{U}})\right) / \left(1 + \tau\,\chi(\mathbf{u}, \boldsymbol{\mathcal{U}})\right) \\ \left(\bar{\mathsf{J}}^{(*)} + \tau\,\bar{\eta}(\mathbf{u}, \boldsymbol{\mathcal{U}})\right) / \left(1 + \tau\,\bar{\chi}(\mathbf{u}, \boldsymbol{\mathcal{U}})\right) \end{pmatrix}. \tag{24}$$

Here $\mathbf{g}$ is derived by substituting Eqs. (21c) and (21d) into the right-hand sides of Eqs. (21a) and (21b), and C_{Y_e} and C_{ϵ} collect the known constants from the explicit state $\mathbf{U}^{(*)}$. Specifically, $C_{\mathsf{Y}_e} = \mathsf{Y}_e^{(*)} + \left(\frac{m_{\mathrm{B}}}{\rho}\right)\left(\frac{4\pi}{h^3}\right)\sum_q W_q^{(2)}\left(\mathsf{J}_q^{(*)} - \bar{\mathsf{J}}_q^{(*)}\right)$ and $C_{\epsilon} = \epsilon^{(*)} + \left(\frac{1}{\rho}\right)\left(\frac{4\pi}{h^3}\right)\sum_q W_q^{(3)}\left(\mathsf{J}_q^{(*)} + \bar{\mathsf{J}}_q^{(*)}\right)$. This substitution simplifies the exposition since the resulting $\mathbf{g}$ only depends on $\boldsymbol{\mathcal{U}}$. On the other hand, $\boldsymbol{\mathcal{G}}$ comes from reorganizing (21c)–(21d), which ensures that $\mathbf{G}$ is a contraction map, i.e., the Lipschitz constant of $\mathbf{G}$ is strictly less than one. In addition, for the sake of clarity, the fact that the opacities $(\eta, \chi, \bar{\eta}, \bar{\chi})$ depend on $\mathbf{u} = (\mathsf{Y}_e^{(+)}, \epsilon^{(+)})$ (through opacity kernels Φ) and $\boldsymbol{\mathcal{U}} = (\mathsf{J}^{(+)}, \bar{\mathsf{J}}^{(+)})$ is expressed explicitly.

We consider two fixed-point approaches for solving (22) – the coupled approach and the nested approach. When solving the nonlinear system (22), fixed-point methods are more attractive than Newton's method because they (1) do not require the Jacobian matrix, which can be difficult to compute accurately with tabulated opacities; and (2) avoid inversion of dense linear systems. On the other hand, the rate of convergence can be slower for fixed-point methods than that of Newton-based methods. We intend to investigate this further in a future study.

4.1. Coupled fixed-point algorithm

The coupled approach considers Eq. (22) as a fixed-point problem with unknowns $\mathbf{U}$, e.g., applying Picard iteration on Eq. (22) leads to

$$\mathbf{U}^{(k+1)} = \mathbf{G}(\mathbf{U}^{(k)}). \tag{25}$$

Here the opacities $(\eta, \chi, \bar{\eta}, \bar{\chi})$ in $\mathbf{G}$ are updated at each iteration k using $\mathbf{U}^{(k)}$ and thus are consistent with the solution. The Picard iteration guarantees convergence when $\mathbf{G}$ is a contraction map, however, the convergence could be slow. To achieve faster convergence,

we implement Anderson acceleration [3, 44] to solve Eq. (22). Anderson acceleration utilizes information from previous iterations to update the unknowns, which is expected to give faster convergence than Picard iteration, but at a cost of additional memory usage. Specifically, in iteration $k+1$, Anderson acceleration on the coupled problem first computes the residual

$$\mathbf{r}^{(k)} := \mathbf{G}(\mathbf{U}^{(k)}) - \mathbf{U}^{(k)} , \tag{26}$$

then solves a least-squares problem $\alpha^* := \underset{\alpha\in\mathbb{R}^{m_k+1}}{\operatorname{argmin}} \left\{ \left\| \sum_{i=0}^{m_k} \alpha_i \, \mathbf{r}^{(k-i)} \right\|_2^2 : \sum_{i=0}^{m_k} \alpha_i = 1 \right\}$ with $m_k := \min\{m, k\}$, and finally updates

$$\mathbf{U}^{(k+1)} = \sum_{i=0}^{m_k} \alpha_i^* \, \mathbf{G}(\mathbf{U}^{(k-i)}) . \tag{27}$$

Here the truncation parameter $m \geq 0$ is an integer that indicates the "memory" of Anderson acceleration, i.e., the maximum number of residuals kept in memory. When $m = 0$, the solver reduces to Picard iteration. For $m > 0$, Anderson acceleration updates $\mathbf{U}$ using a linear combination of the last m_k iterates that leads to the minimum residual. The convergence properties of Anderson acceleration were shown in [43], which include (i) global convergence on linear problems under the standard contraction assumption, and (ii) local convergence on nonlinear problems under assumptions similar to the standard assumptions for local convergence of Newton's method. In the numerical examples in Section 5, we use $m = 2$, which we have found to significantly reduce the number of iterations when compared to Picard. For $m = 2$, the additional memory required for Anderson acceleration is small since each implicit solve is local in space. In addition, the least squares problem for α_i^* is small, and can be written out explicitly or solved using an optimized linear algebra library (e.g., LAPACK).

4.2. Nested fixed-point algorithm

The second approach we consider is a nested algorithm, which formulates Eq. (22) as a nested fixed-point problem with two layers

$$\mathbf{u} = \mathbf{g}(\hat{\boldsymbol{\mathcal{U}}}(\mathbf{u})) , \tag{28a}$$

$$\hat{\boldsymbol{\mathcal{U}}}(\mathbf{u}) = \boldsymbol{\mathcal{G}}(\mathbf{u}, \hat{\boldsymbol{\mathcal{U}}}(\mathbf{u})) , \tag{28b}$$

where the outer layer, Eq. (28a), is a fixed-point problem on the matter states $\mathbf{u}$, and the inner, Eq. (28b), is on the distributions $\hat{\boldsymbol{\mathcal{U}}}$ for fixed matter states $\mathbf{u}$. These two problems are nested in the sense that evaluating the right-hand side of Eq. (28a) at a given $\mathbf{u}$ requires solving Eq. (28b). For example, applying Picard iteration on both Eqs. (28a) and (28b) gives the following iterative scheme

$$\mathbf{u}^{(k+1)} = \mathbf{g}(\hat{\boldsymbol{\mathcal{U}}}(\mathbf{u}^{(k)})) , \tag{29a}$$

where $\hat{\boldsymbol{\mathcal{U}}}(\mathbf{u}^{(k)}) = \boldsymbol{\mathcal{U}}^{(k,*)}$, the limit point of the inner Picard iteration

$$\boldsymbol{\mathcal{U}}^{(k,\ell+1)} = \boldsymbol{\mathcal{G}}(\mathbf{u}^{(k)}, \boldsymbol{\mathcal{U}}^{(k,\ell)}) . \tag{29b}$$

In practice, we use Anderson acceleration with $m = 2$, as described in Section 4.1, to accelerate both the inner and outer solves separately.

The nested approach is motivated by the fact that in solving Eq. (22), the most costly part is evaluating the opacity kernels Φ at a given matter state $\mathbf{u}$, which is performed in $\boldsymbol{\mathcal{G}}$ whenever $\mathbf{u}$ is updated. Therefore, while the coupled approach seems simple and straightforward, the nested structure in Eq. (28) justifies the additional complexity by reducing the number of updates (on $\mathbf{u}$) in Eq. (28a) via a more accurate distribution update given by solving Eq. (28b) at the current matter state. Note that the matter state $\mathbf{u}$ is fixed in the solution procedure of the inner problem in Eq. (28b) and does not require opacity kernel reevaluations.

4.3. Fixed-point convergence criteria

Here we introduce the convergence criteria applied on the two fixed-point algorithms. For fixed-point problems of the form $\mathbf{U} = \mathbf{G}(\mathbf{U})$, commonly used convergence criteria are either on the error $\|\mathbf{U}^{(k)} - \mathbf{U}^{(k-1)}\|$ or the residual $\|\mathbf{U}^{(k)} - \mathbf{G}(\mathbf{U}^{(k)})\|$. These criteria do not directly apply to the problems considered here, since the unknowns $\mathbf{U} = (\mathsf{Y}_e, \mathsf{\epsilon}, \mathsf{J}, \bar{\mathsf{J}})$ in these problems are physical quantities of different scales. To address this, we impose a separate convergence criterion on each physical quantity of interest, i.e., the convergence criteria take the form

$$|\mathsf{Y}_e^{(k+1)} - \mathsf{Y}_e^{(k)}| \le \mathtt{tol}\,|\mathsf{Y}_e^{(0)}|\,, \quad |\mathsf{\epsilon}^{(k+1)} - \mathsf{\epsilon}^{(k)}| \le \mathtt{tol}\,|\mathsf{\epsilon}^{(0)}|\,, \tag{30a}$$

$$\|\mathsf{J}^{(k+1)} - \mathsf{J}^{(k)}\| \le \mathtt{tol}\,\|\mathsf{J}^{(0)}\|\,, \quad \|\bar{\mathsf{J}}^{(k+1)} - \bar{\mathsf{J}}^{(k)}\| \le \mathtt{tol}\,\|\bar{\mathsf{J}}^{(0)}\|\,, \tag{30b}$$

where $\mathtt{tol} > 0$ is a constant relative tolerance. In the coupled algorithm in Section 4.1, all four criteria in (30) are applied. On the other hand, in the nested algorithm in Section 4.2, the outer layer (Eq. (28a)) and inner layer (Eq. (28b)) use the convergence criteria (30a) and (30b), respectively. In the numerical experiments, we set $\mathtt{tol}$ to be identical in each criterion in (30).

5. Numerical experiments

We compare the performance of the nonlinear solvers discussed in Section 4 using a benchmark problem, *Deleptonization Wave*, that mimics the conditions in a collapsed stellar core. The initial matter profiles (for ρ, T, and Y_e) are given by the analytic expressions

$$\rho(r) = 0.5 \times \left\{D_{\max} \times \left[1 - \tanh\left((r - R_D)/H_D\right)\right] + D_{\min} \times \left[1 - \tanh\left((R_D - r)/H_D\right)\right]\right\}, \tag{31}$$

$$T(r) = 0.5 \times \left\{T_{\max} \times \left[1 - \tanh\left((r - R_T)/H_T\right)\right] + T_{\min} \times \left[1 - \tanh\left((R_T - r)/H_T\right)\right]\right\}, \tag{32}$$

$$Y_e(r) = 0.5 \times \left\{Y_{\min} \times \left[1 - \tanh\left((r - R_Y)/H_Y\right)\right] + Y_{\max} \times \left[1 - \tanh\left((R_Y - r)/H_Y\right)\right]\right\}, \tag{33}$$

where, for the density profile we set $D_{\min} = 10^8$ g cm^{-3}, $D_{\max} = 4 \times 10^{14}$ g cm^{-3}, $R_D = 20$ km, and $H_D = 10$ km. Thus, in the inner core the maximum density is consistent with nuclear matter, and neutrinos are trapped. As the density decreases with radius, the neutrino mean free path increases, allowing neutrinos to escape the computational domain (deleptonization). For the temperature profile we set $T_{\min} = 5 \times 10^9$ K, $T_{\max} = 2.6 \times 10^{11}$ K, $R_T = 25$ km, and $H_T = 20$ km, while for the electron fraction profile we set $Y_{\min} = 0.3$, $Y_{\max} = 0.46$, $R_Y = 45$ km, and $H_Y = 10$ km. The initial density, temperature, and electron fraction profiles are plotted in Figure 1. The radiation field is initialized by setting the distribution function equal to the local Fermi-Dirac distribution; i.e., $\mathcal{J} = \mathcal{J}_0$ and $\bar{\mathcal{J}} = \bar{\mathcal{J}}_0$, and $\boldsymbol{\mathcal{H}} = \bar{\boldsymbol{\mathcal{H}}} = 0$.

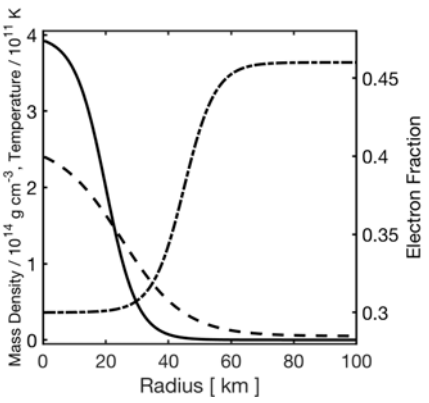

Figure 1. Initial matter profiles versus radius for the Deleptonization Wave problem: rest mass density (solid), temperature (dashed), and electron fraction (dash-dot).

We first solve this problem in one spatial dimension (imposing spherical symmetry) with linear elements ($k = 1$) on a spatial domain $r \in [0, 100]$ km divided into 64 uniform elements, and an energy domain covering $\varepsilon \in [0, 300]$ MeV covered by 16 geometrically progressing elements where the first element has $\Delta\varepsilon = 4$ MeV and the last element has $\Delta\varepsilon \approx 50$ MeV. We evolve until $t = 5$ ms using a fixed time step $\Delta t \approx 10^{-3}$ ms.

Figure 2 provides an overview of the results from the coupled algorithm in Section 4.1 (see caption for additional details). In the top panel we

plot the electron fraction versus radius for select times. Neutrinos are trapped in the central core ($r \lesssim 30$ km) and the electron fraction remains relatively unchanged. For larger radii, as the density decreases with radius, neutrinos decouple from matter and electron capture on protons ($e^- + p \rightarrow n + \nu_e$) produce neutrinos that are transported radially to decrease the electron fraction around the middle of the domain (30 km $\lesssim r \lesssim$ 60 km). Some reabsorption ($n + \nu_e \rightarrow e^- + p$) results in an increase in the electron fraction in the outer part of the domain ($r \gtrsim 60$ km), but overall the matter is deleptonized. In the lower left panel of Figure 2 we plot the flux factor h (solid) and the mean energy $\langle \varepsilon \rangle$ (dashed) versus radius at $t = 5$ ms, defined respectively as

$$\left\{ \mathsf{h}, \langle \varepsilon \rangle \right\} = J^{-1} \int_{\mathbb{R}^+} \left\{ \mathcal{H}, \mathcal{J}\, \varepsilon \right\} dV_\varepsilon, \quad \text{where} \quad J = \int_{\mathbb{R}^+} \mathcal{J}\, dV_\varepsilon. \tag{34}$$

In the inner regions, the neutrino distribution is essentially isotropic ($\mathsf{h} \approx 0$). As neutrinos decouple from matter and transition to free-streaming, the flux factor increases monotonically with radius towards its maximum value of unity. The mean energy is largest in the center (around 150 MeV for $r = 0$), decreases to about 12 MeV at $r = 60$ km, and then remains fairly constant with radius. We plot the neutrino number density $\mathcal{J}$ (black lines) and the equilibrium distribution $\mathcal{J}_0$ (blue lines) versus energy ε at $t = 5$ ms for various radii in the lower right panel of Figure 2. For $r = 20$ km (solid lines), neutrinos are strongly coupled to matter and $\mathcal{J} \approx \mathcal{J}_0$. At a larger radius, when $r = 50$ km (dashed lines), the neutrino-matter coupling weakens and $\mathcal{J} \approx \mathcal{J}_0$ for $\varepsilon \gtrsim 15$ MeV (note that the neutrino opacity increases roughly as ε^2). Neutrinos are further decoupled from matter at $r = 80$ km (dash-dot lines), and the spectral density deviates substantially from the equilibrium distribution for all energies.

In Figure 3 we compare the coupled (Section 4.1) and nested (Section 4.2) algorithms in terms of iteration counts needed to reach a specified tolerance $\texttt{tol} = 10^{-8}$ with convergence criteria (30) as discussed in Section 4.3. (At $t = 5$ ms, the relative difference in temperature and electron fraction obtained with the two algorithms is less than 10^{-3}). We plot the total number of iterations versus radius and time. The coupled algorithm (left panel in Figure 3) requires more iterations involving opacity evaluations to converge. Most iterations (up to ten) are needed in the high density ($> 10^{14}$ g cm^{-3}; cf. vertical dashed lines) part of the domain. Overall, the number of iterations decreases with increasing radius, varying between four and six for $\rho \in [10^{11}, 10^{14}]$ g cm^{-3}. Only two iterations are needed at lower densities ($< 10^{11}$ g cm^{-3}). The total number of iterations at a given radius is not very sensitive to time. For the nested algorithm we plot both the inner and outer iteration counts (middle and right panels in Figure 3, respectively), where the inner iteration count is averaged over the number of outer iterations per solve. The nested algorithm requires fewer (at most four) outer iterations, and hence fewer opacity evaluations, than the coupled algorithm. Between one and three outer iterations are needed in the high density regions ($> 10^{14}$ g cm^{-3}), three to four are needed in the regions with $\rho \in [10^{11}, 10^{14}]$ g cm^{-3}, while two are needed at lower densities. The number of inner iterations is highest (at most eight) at high density, and decreases with increasing radius, almost in a monotonic fashion, to one iteration outside $r = 60$ km. At $t = 5$ ms, the total number of iterations for the coupled algorithm is 169500, versus 88770 for the nested algorithm.

We compare the performance of the nonlinear solvers in the DG-IMEX scheme in terms of elapsed wall-time per time step in Figure 4. For this purpose we ran the Deleptonization Wave problem in 3D for four time steps (with $\Delta t \approx 3 \times 10^{-3}$ ms) using Cartesian coordinates, a cubic spatial domain with $x^i \in [0, 100]$ km, covered by 12^3 linear ($k = 1$) elements. The energy domain is the same as for the spherically symmetric problem described above.

In Figure 4 we plot the time consumed by the primary components of the scheme for the coupled (left columns) and nested (right columns) solvers. The total time per step t_{Tot} is represented by the blue columns, while the time spent by the explicit updates (t_{Ex}), implicit updates (t_{Im}), and the positivity limiter (t_{PL}) are represented by the red, purple, and orange

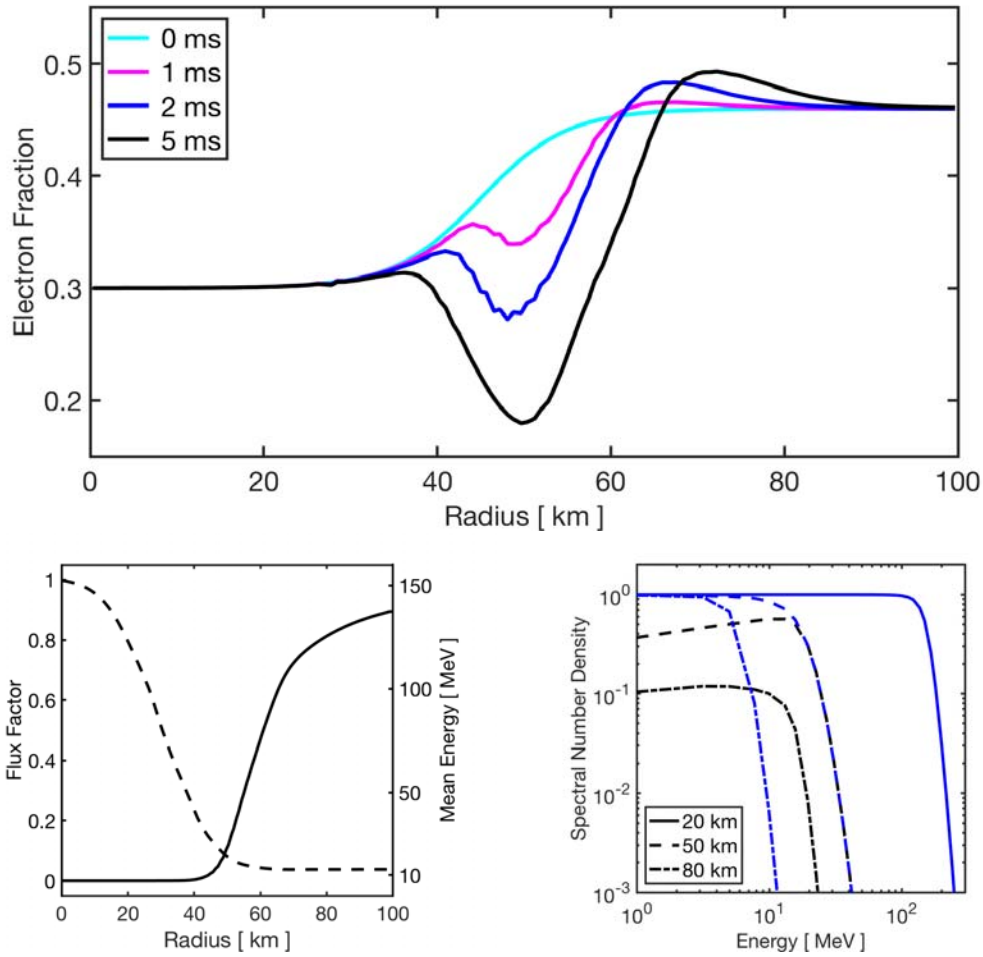

Figure 2. Results from the spherically symmetric Deleptonization Wave problem. In the upper panel we plot the electron fraction versus radius for various times: $t = 0$ ms (cyan), $t = 1$ ms (magenta), $t = 2$ ms (blue), and $t = 5$ ms (black). In the lower left panel we plot the flux factor h (solid) and the mean energy $\langle\varepsilon\rangle$ (dashed) versus radius for electron neutrinos at $t = 5$ ms. In the lower right panel we plot the electron neutrino number density $\mathcal{J}$ versus energy at $t = 5$ ms for various radii: $r = 20$ km (solid black), $r = 50$ km (dashed black), and $r = 80$ km (dash-dot black). We also plot the local equilibrium number density $\mathcal{J}_0$ for the same radii (blue lines with matching line style). Note that $\mathcal{J} \approx \mathcal{J}_0$ for $r = 20$ km.

columns, respectively ($t_{\rm Tot} \approx t_{\rm Ex} + t_{\rm Im} + t_{\rm PL}$). We also plot the time spent doing opacity interpolations ($t_{\rm Op}$; green columns), which is part of the implicit update. For both solvers, the implicit solve dominates the overall computational cost of the DG-IMEX scheme, with $t_{\rm Im}/t_{\rm Tot} \approx 0.88$ and $t_{\rm Im}/t_{\rm Tot} \approx 0.86$ for the coupled and nested algorithms, respectively. We also find that opacity interpolations dominate the cost of the implicit solve, with $t_{\rm Op}/t_{\rm Im} \approx 0.91$ and $t_{\rm Op}/t_{\rm Im} \approx 0.84$ for the coupled and nested algorithms, respectively. The nested algorithm spends less time doing opacity interpolations ($t_{\rm Op} = 9.3$ s, versus $t_{\rm Op} = 12.3$ s for the coupled algorithm), and as a result it is about 15 percent faster than the coupled algorithm).

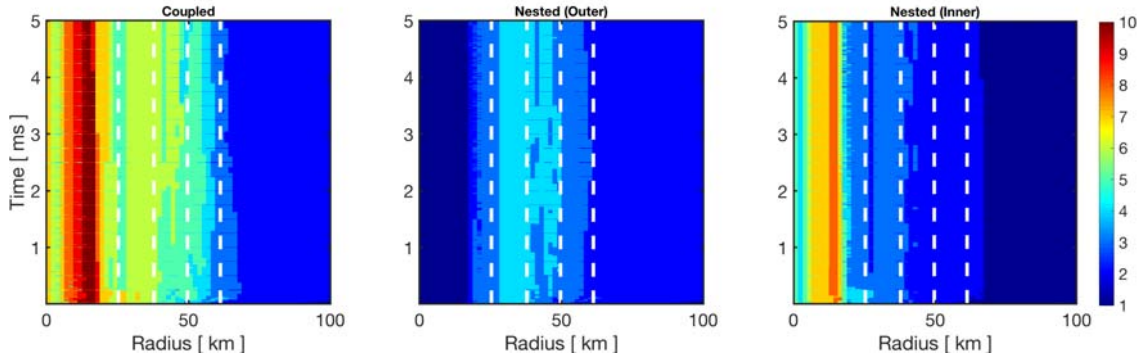

Figure 3. Plot of iteration count versus radius and time for the coupled algorithm (left panel) and the nested algorithm (middle and right panels). Vertical white dashed lines (starting from the left) indicate where the rest mass density equals 10^{14}, 10^{13}, 10^{12}, and 10^{11} g cm^{-3}.

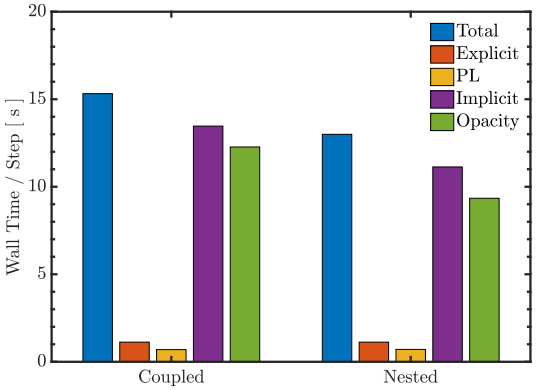

Figure 4. Elapsed wall-time per step for the 3D *Deleptonization Wave* problem using the coupled (Section 4.1) and nested (Section 4.2) solvers. The total execution of the DG-IMEX scheme is further divided into its primary components: explicit update, implicit update, positivity limiter (PL), and opacity interpolations, which are part of the implicit update. Runs were performed on a single core of an IBM POWER9 CPU and used the PGI Fortran compiler (version 19.4).

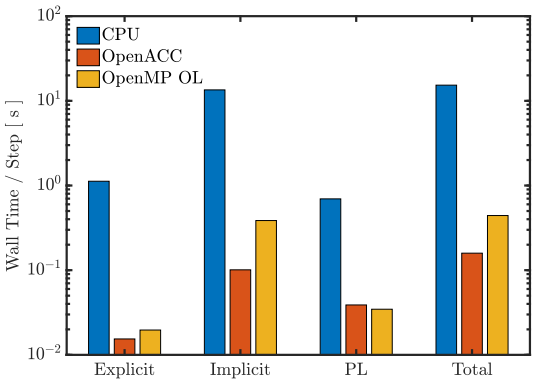

Figure 5. Comparison of different programming models' elapsed wall-time per step for the 3D *Deleptonization Wave* problem using the coupled nonlinear solver. The total execution of the DG-IMEX scheme is further divided into its primary components: explicit update, implicit update, and positivity limiter (PL). `OpenMP` offload results used the IBM XL Fortran compiler (version 16.1.1-3). CPU and `OpenACC` results used the PGI Fortran compiler (version 19.4).

6. Performance Portability

High-performance computing has entered an era of increased heterogeneity and diversity in computer architectures. As part of the Exascale Computing Project, `thornado` will be incorporated into large scale simulations of CCSN with a parallel adaptive-mesh refinement (AMR) code like `FLASH` [24]. Effectively utilizing the variety of architectures these codes may run on will be critical to our ability to incorporate new physics capabilities in these types of simulations on realistic timescales and with limited computational resources. In this context, we are mainly concerned with the performance of `thornado` at the scale of a single MPI rank since optimizations at larger scales will be managed by the encompassing AMR framework [45].

In this section, we demonstrate our approach to performance portability in `thornado`, which we define simply as the ability of the code to take full advantage of a target architecture. For the purpose of this paper, we constrain the scope of our discussion to only include GPUs, but the approach could apply to other novel architectures.

Programming models based on abstractions for parallel execution and data management across a variety of architectures have been very successful for C++ (e.g., `Kokkos` [22] or `RAJA` [26]). However, these types of abstractions do not conform to `thornado` as a Fortran code. Instead, we rely on a combination of preprocessor macros, GPU compiler directives (`OpenMP` or `OpenACC`), and generic interfaces to optimized linear algebra libraries (e.g., `MAGMA` [21]) to fully accelerate all components of the DG-IMEX scheme. For example, consider the code in Listing 1 that performs an array permutation and the subsequent interpolation to a cell faces in Listing 2, needed in the implicit part of the scheme described in Section 3. With this approach, depending on the available hardware and programming environment, we are able to choose at compile time whether to use `OpenMP` with GPU offload, `OpenACC`, or more traditional `OpenMP` for CPU multi-core to execute the nested loop in parallel.

```
#if defined(THORNADO_OMP_OL)
!$OMP TARGET TEAMS DISTRIBUTE PARALLEL DO SIMD COLLAPSE(7)
#elif defined(THORNADO_OACC)
!$ACC PARALLEL LOOP GANG VECTOR COLLAPSE(7)
#elif defined(THORNADO_OMP)
!$OMP PARALLEL DO SIMD COLLAPSE(7)
#endif
DO iZ4 = iZB4-1, iZE4+1 ; ... ; DO iNode = 1, nDOF
  uCR_K(iNode,iZ1,iZ2,iZ3,iM,iS,iZ4) = U(iNode,iZ1,iZ2,iZ3,iZ4,iM,iS)
END DO ; ... ; END DO
```

Listing 1. Example of array permutation using tightly nested loops that are common throughout `thornado`.

```
SUBROUTINE MatrixMatrixMultiply(...)
...
#if defined(THORNADO_OMP_OL)
!$OMP TARGET DATA USE_DEVICE_PTR( pa, pb, pc )
#elif defined(THORNADO_OACC)
!$ACC HOST_DATA USE_DEVICE( pa, pb, pc )
#endif
da = C_LOC( pa ) ; db = C_LOC( pb ) ; dc = C_LOC( pc )
#if defined(THORNADO_OMP_OL)
!$OMP END TARGET DATA
#elif defined(THORNADO_OACC)
!$ACC END HOST_DATA
#endif
#if defined(THORNADO_LA_CUBLAS)
ierr = cublasDgemm_v2( ..., da, lda, ... )
#elif defined(THORNADO_LA_MAGMA)
CALL magma_dgemm( ..., da, lda, ... )
#else
CALL DGEMM( ..., a, lda, ... )
#endif
END SUBROUTINE
INTERFACE
  SUBROUTINE MatrixMatrixMultiply &
    ( transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c, ldc )
  CHARACTER                          :: transa, transb
  INTEGER                            :: m, n, k, lda, ldb, ldc
  REAL(DP)                           :: alpha, beta
  REAL(DP), DIMENSION(lda,*), TARGET :: a
  REAL(DP), DIMENSION(ldb,*), TARGET :: b
  REAL(DP), DIMENSION(ldc,*), TARGET :: c
  END SUBROUTINE
END INTERFACE
...
```

```
CALL MatrixMatrixMultiply &
  ( 'N', 'N', nDOF_X3, nF, nDOF, One, L_X3_Up, nDOF_X3, &
    uCR_K(1,iZB1,iZB2,iZB3,1,1,iZB4-1), nDOF, Zero, &
    uCR_L(1,iZB1,iZB2,iZB3,1,1,iZB4   ), nDOF_X3 )
```

Listing 2. Example of interface for matrix-matrix multiplication to be performed by an optimized linear algebra library. The `TARGET` attribute is added to the dummy arguments so the C address can be ascertained and passed to C routines with the `ISO_C_BINDING` interface.

At this time, we only show results using the coupled algorithm described in Section 4.1 and defer discussion detailing comparisons among the different GPU implementations of the nonlinear solvers to a future paper. In Figure 5, we show some preliminary profiling results for the deleptonization wave described in Section 5 in a three-dimensional (3D) setting, where we have adapted the benchmark to be more representative of workloads that one might encounter in a large scale simulation with `FLASH`. Specifically, we solve the problem in a 3D box on a Cartesian mesh with 12 elements in each dimension but otherwise retain the previously described problem setup; this is a fairly good representation of a single AMR "block" in a `FLASH` simulation that would be local to one MPI rank. This benchmark was run on a single node of the Summit computer at the Oak Ridge Leadership Computing Facility (OLCF). Each node is comprised of 2 IBM POWER9 CPUs and 6 NVIDIA Volta GPUs, but for the reasons described earlier in this section we limit our tests to a single CPU core and one GPU.

The most notable result from this benchmark is a ∼100x overall speedup of `OpenACC` relative to using only the CPU, bringing the time per time step down from $t_{\text{Tot}} \approx 15.3$ s to $t_{\text{Tot}} \approx 0.16$ s. Furthermore, we find that the implementation of `OpenMP` offloading by the IBM XL compiler was about three times slower than the `OpenACC` implementation by PGI. Preliminary analysis suggests this may be accounted for by different choices for the GPU kernel launch configurations (i.e. fewer thread blocks and more registers), but this warrants more study before we can draw any definitive conclusions. Overall, these results show that `thornado` is well-suited to GPU acceleration. We suspect this is due to a high-degree of SIMD-like parallelism and high arithmetic intensity, particularly in operations such as neutrino opacity interpolations, but more analysis (e.g., roofline model) is left for the subject of future study.

7. Summary

We have described algorithms for neutrino-matter coupling as implemented in `thornado`. The transport of electron neutrinos and antineutrinos in conditions expected in core-collapse supernovae is modeled with a two-moment model using the algebraic fermion closure from [11]. We use tabulated neutrino-matter interactions from [7], which include emission and absorption, iso-energetic scattering, neutrino-electron scattering, and pair processes. The two-moment model is discretized in position-energy space using the discontinuous Galerkin method, and we use implicit-explicit time integration, where the neutrino-matter interactions are treated implicitly. The nonlinear system governing the neutrino-matter coupling is solved iteratively using Anderson-accelerated fixed-point iteration, and we have investigated two approaches: (1) a fully coupled approach, and (2) a nested approach, where the nested approach aims to reduce expensive opacity evaluations. We have found that the nested approach can reduce the computational cost of the implicit solve. Furthermore, we have discussed our approach to porting the neutrino transport in `thornado` to GPUs. Results obtained with the fully coupled algorithms on GPUs, using `OpenACC` and `OpenMP` offloading, show speedups of ∼100x can be achieved when compared to running with a single CPU core. These results indicate that the algorithms implemented in `thornado` are well-suited for GPU acceleration. This is certainly true for the benchmarking experiments performed here with linear DG, but in lieu of further experimentation, we speculate only modest quantitative implications for the performance at higher orders. Future investigations will include comparison of the algorithms presented here

with algorithms based on Newton's method, and comparison of the relative performance of these algorithms on GPUs.

Acknowledgments
This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research was sponsored, in part, by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. `thornado` uses equation of state and opacity tables and subroutines developed under the `WeakLib` project. We acknowledge the contributions to `WeakLib` of Ryan Landfield and Eric J. Lentz.

References
[1] Adams M 2001 *Nuclear Science and Engineering* **137** 298
[2] Almgren A, Beckner V E, Bell J B, et al. 2010 *ApJS* **715** 1221
[3] Anderson D G 1965 *J. ACM* **12** 547
[4] Arnett W D *Canadian Journal of Physics* **44** 2553
[5] Ascher U M, Ruuth S J and Spiteri R J 1997 *Applied Numerical Mathematics* **23** 151
[6] Bassi F, Franchina N, Ghidoni A and Rebay S 2013 *Int. J. Numer. Meth. Fluids* **71** 1322
[7] Bruenn S W 1985 *ApJS* **58** 771
[8] Bruenn S W, Blondin J M, Hix W R, et al. 2018 *arXiv:1809.05608v1*
[9] Burrows A 2013 *Rev. Mod. Phys.* **85** 245
[10] Burrows A, Radice D and Vartanyan D 2019 *MNRAS* **485** 3153
[11] Cernohorsky J and Bludman J A 1994 *ApJ* **433** 250
[12] Chu R, Endeve E, Hauck C D and Mezzacappa A 2019 *JCP* **389** 62
[13] Chu R, Endeve E, Hauck C D, et al. (2019) *J. Phys.: Conf. Ser.* **1225** 012013
[14] Cockburn B and Shu C-W 1989 *Math. Comput.* **52** 411
[15] Cockburn B and Shu C-W 1991 *M2AN* **25** 337
[16] Cockburn B and Shu C-W 1998 *JCP* **141** 199
[17] Cockburn B and Shu C-W 2001 *Journal of Scientific Computing* **16** 173
[18] Cockburn B, Lin S-Y and Shu C-W 1989 *JCP* **84** 90
[19] Cockburn B, Hou S and Shu C-W 1990 *Math. Comput.* **54** 545
[20] Colgate S A and White R H 1966 *ApJ* **143** 626
[21] Dongarra J, Gates M, Haidar A et al. 2014 *Numerical Computations with GPUs* Springer
[22] Edwards H C, Trott C R and Sunderland D 2014 *Journal of Parallel and Distributed Computing* **74** 3302
[23] Endeve E, Buffaloe J, Dunham S J, et al. (2019) *J. Phys.: Conf. Ser.* **1225** 012014
[24] Fryxell B, Olson K, Ricker P et al. 2000 *ApJS* **131** 273
[25] Hesthaven J S and Warburton T 2008 *Nodal discontinuous Galerkin methods: Algorithms, analysis and applications* Springer
[26] Hornung R, Jones H, Keasler J et al. 2015 *ASC Tri-lab Co-design Level 2 Milestone Report* Technical Report LLNL-TR-677453
[27] Janka H-Th 2012 *Annu. Rev. Nucl. Part. Sci.* **62** 407
[28] Klöckner A, Warburton T, Bridge J and Hesthaven J S 2009 *JCP* **228** 7863
[29] Larsen E W and Morel J E 1989 *JCP* **83** 212
[30] Lentz E J, Mezzacappa A, Messer O E B, et al. 2012 *ApJ* **747** 73
[31] Lentz E J, Bruenn S W, Hix W R, et al. 2015 *ApJL* **807** L31
[32] Levermore C D 1984 *JQSRT* **31** 149
[33] Melson T, Janka H-Th and Marek A 2015 *ApJL* **801** L24
[34] Mezzacappa A 2005 *Annu. Rev. Nucl. Part. Sci.* **55** 467
[35] Müller B, Janka H-Th and Dimmelmeier H 2010 *ApJS* **189** 104
[36] Müller B, Janka H-Th and Marek A 2012 *ApJ* **756** 84
[37] Müller B *PASA* **33** 1
[38] Pareschi L and Russo G 2005 *Journal of Scientific Computing* **25** 129
[39] Remacle J-F, Flaherty J E and Shephard M S 2003 *SIAM Review* **45** 53
[40] Shu C-W and Osher S 1988 *JCP* **77** 439

[41] Skinner M A, Dolence J C, Burrows A, et al. 2019 *ApJS* **241** 7
[42] Teukolsky S A 2016 *JCP* **312** 333
[43] Toth A and Kelley C 2015 *SIAM Journal on Numerical Analysis* **53** 805
[44] Walker H and Ni P 2011 *SIAM Journal on Numerical Analysis* **49** 1715
[45] Zhang W, Almgren A, Beckner V et al. 2019 *The Open Journal* **4** 1370