DRAFT VERSION NOVEMBER 17, 2021
Typeset using LATEX **modern** style in AASTeX63

An Improved Method for Coupling Hydrodynamics with Reaction Networks and Nuclear Statistical Equilibrium

M. ZINGALE,[1] M. P. KATZ,[2] A. J. NONAKA,[3] AND ALICE HARPOLE[1]

[1]*Dept. of Physics and Astronomy, Stony Brook University, Stony Brook, NY 11794-3800*
[2]*Nvidia Corp*
[3]*Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

ABSTRACT

Reacting astrophysical flows can be challenging to model because of the difficulty in accurately coupling hydrodynamics and reactions. This can be particularly acute during explosive burning or at high temperatures where nuclear statistical equilibrium is established. We develop a simplified approach based on the ideas of spectral deferred corrections (SDC) coupling of explicit hydrodynamics and stiff reaction sources as an alternative to operator splitting or the more comprehensive SDC approach we demonstrated previously. We apply the new method to some example problems and show how to modify it to work with a hybrid network consisting of a reaction ODE system and a table for nuclear statistical equilibrium. This is all done in the framework of the Castro hydrodynamics code, and all algorithm implementations are freely available.

Keywords: hydrodynamics—methods: numerical

1. INTRODUCTION

Modeling astrophysical reacting flows can be challenging because of the disparity better the nuclear and hydrodynamics timescales. Reaction networks tend to be stiff, requiring implicit integration techniques to stably integrate the system. In contrast, compressible hydrodynamics flows are limited by the (often much longer) sound-crossing time over a computational cell and can be solved using explicit time integration. Traditional methods of coupling hydrodynamics and reactions used in astrophysics use operator splitting—each physical process acts on the output of the previous process in alternating fashion. This makes it easy to use different time-integration methods for the different physics, and to build a simulation code in a modular way. However, competition between the different

Corresponding author: Michael Zingale
michael.zingale@stonybrook.edu

physical processes can cause the coupling to breakdown. These splitting errors can lead to loss of accuracy and further time step limitations.

A particularly difficult phase of evolution to model is the nuclear statistical equilibrium that sets in for temperatures in excess of few $\times 10^9$ K. Physically, the forward and reverse rates of reaction should balance leading to an equilibrium. With operator splitting, an NSE region will have a large positive flow through the network in a zone in one step followed by a large negative flow over the next timestep, as the code struggles to produce an equilibrium. These large changes in abundances (and large alternately positive and negative energy generation rates) can be a challenge for a code. The easiest way to improve the coupling is to cut the timestep, but this makes simulations prohibitively expensive. Sometimes the burning is simply halted on a zone-by-zone basis when NSE conditions are reached (e.g., as in Zingale et al. 2001). Alternately, at high temperatures, a reaction network can be replaced with a table of NSE abundances and the zone's composition set through table look-ups (e.g. Ma et al. 2013)

need more here

The Castro hydrodynamics code (Almgren et al. 2010; Almgren et al. 2020) is used for all of our numerical experiments. Castro is a compressible (magneto-, radiation-) hydrodynamics code built on the AMReX adaptive mesh refinement (AMR) framework (Zhang et al. 2019). Castro has been designed to be performance portable and runs on massively parallel CPU and GPU architectures (Katz et al. 2020). For hydrodynamics, the corner transport upwind (CTU) (Colella 1990) method with the piecewise parabolic method (PPM) (Colella & Woodward 1984; Miller & Colella 2002) is used. Castro includes self-gravity, rotation, arbitrary equations of state and reaction networks, and has been used for modeling X-ray bursts and different models of thermonuclear, core-collapse, and pair-instability supernovae. Recently in Zingale et al. (2019), we developed second- and fourth-order accurate methods in space and time for coupling hydrodynamics and nuclear reaction networks based on spectral deferred corrections (SDC) methods, and demonstrated the method in a variety of test problems.

The time-integration approach presented here is considerably simpler than the SDC method of Zingale et al. (2019), but allows us to reuse the main CTU hydrodynamics construction and a largely similar ODE integration scheme, making this method easier to add to existing simulation codes. Furthermore, it also extends to adaptive mesh refinement with subcycling in a straightforward manner, avoiding the complications described in (McCorquodale & Colella 2011) needed to fill ghost cells when using method-of-lines integration. However, it is restricted to second-order accuracy overall. We term this algorithm the "simplified SDC method". In this paper we describe the overall method, show its extension to NSE, and demonstrate it on several test problems using Castro. All of the code to reproduce the results in this paper are in the Castro github repository[1].

2. NUMERICAL METHODOLOGY

[1] https://github.com/amrex-astro/Castro/

We solve the Euler equations for compressible, reacting flow. For ease of exposition we describe the one-dimensional case; multidimensional extensions are a straightforward modification to include the CTU hydrodynamics scheme. Our conserved variables are

$$\boldsymbol{\mathcal{U}} = \begin{pmatrix} \rho \\ \rho X_k \\ \rho \alpha_l \\ \rho u \\ \rho E \\ \rho e \end{pmatrix} \tag{1}$$

where ρ is the mass density, u is velocity, E is specific total energy, p is the pressure, and we carry nuclear species mass fractions, X_k, and auxiliary composition variables, α_l. The specific total energy relates to the specific internal energy, e, as $E = e + u^2/2$, and we also separately evolve e, as part of a dual energy formulation (see Bryan et al. 1995; Katz et al. 2016). The mass fractions are constrained to sum to 1, $\sum_k X_k = 1$, but no such constraint exists on the auxiliary variables. Defining the hydrodynamical fluxes:

$$\mathbf{F}(\boldsymbol{\mathcal{U}}) = \begin{pmatrix} \rho u \\ \rho X_k u \\ \rho \alpha_l u \\ \rho u^2 p \\ (\rho E + p) u \\ \rho e u \end{pmatrix} \tag{2}$$

We can write the system in conservative form for all state variables aside from (ρe) as:

$$\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t} + \frac{\partial \mathbf{F}(\boldsymbol{\mathcal{U}})}{\partial x} = \mathbf{S}(\boldsymbol{\mathcal{U}}) \tag{3}$$

where for the special case of (ρe), we have an additional "pdV" term:

$$\frac{\partial (\rho e)}{\partial t} + \frac{\partial F(\rho e)}{\partial x} + p \frac{\partial u}{\partial x} = S(\rho e) \tag{4}$$

We'll split the source term, $\mathbf{S}(\boldsymbol{\mathcal{U}})$ into pure hydrodynamic and reactive parts:

$$\mathbf{H}(\boldsymbol{\mathcal{U}}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \rho g \\ \rho u g \\ 0 \end{pmatrix} \qquad \mathbf{R}(\boldsymbol{\mathcal{U}}) = \begin{pmatrix} 0 \\ \rho \dot{\omega}(X_k) \\ \rho \dot{\omega}(\alpha_l) \\ 0 \\ \rho \dot{S} \\ \rho \dot{S} \end{pmatrix} \tag{5}$$

with

$$\mathbf{S}(\boldsymbol{\mathcal{U}}) = \mathbf{H}(\boldsymbol{\mathcal{U}}) + \mathbf{R}(\boldsymbol{\mathcal{U}}). \tag{6}$$

Here, gravity, represented by the gravitational acceleration g appears as a pure hydrodynamical source term. From reactions, $\dot{\omega}(X_k)$ is the creation rate for species k, $\dot{\omega}(\alpha_l)$ is the creation rate for auxiliary composition variable l, and $\dot{S}$ is the energy generation rate per unit mass. We note that the internal energy "pdV" work is not treated as a source term but is instead constructed with the hydrodynamical fluxes are computed in the CTU method.

Aside from the $p\partial u/\partial x$ ("pdV") term in the internal energy equation, this system is in conservative form with source terms.

We'll define the advective terms with hydrodynamic sources, $\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})$ as

$$\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) = -\frac{\partial \mathbf{F}(\boldsymbol{\mathcal{U}})}{\partial x} + \mathbf{H}(\boldsymbol{\mathcal{U}}) \tag{7}$$

for the general case, with the (ρe) component again having the extra "pdV" term:

$$\mathcal{A}(\rho e) = -\frac{\partial F(\rho e)}{\partial x} - p\frac{\partial u}{\partial x} + H(\rho e) \tag{8}$$

To close the system, we need the equation of state. For a system where the composition is completely specified by the mass fractions, X_k, the equation of state would take the form:

$$p = p(\rho, X_k, e) \tag{9}$$

However, as we'll see shortly, when we use the NSE table, we are using the results of a much larger network then can be represented by the mass fractions, and in this case, we rely on the auxiliary composition variables, α_l, to describe the state of the composition, and our EOS has the form:

$$p = p(\rho, \alpha_l, e) \tag{10}$$

Sometimes it is preferable to work with the primitive variables,

$$\mathbf{q} = \begin{pmatrix} \rho \\ X_k \\ \alpha_l \\ u \\ p \\ (\rho e) \end{pmatrix} \tag{11}$$

Here, the system appears as:

$$\mathbf{q}_t + \mathbf{A}^{(x)}(\mathbf{q})\mathbf{q}_x = \mathbf{S}(\mathbf{q}) \tag{12}$$

with the matrix $\mathbf{A}^{(x)}$ giving the coefficients of the spatial derivatives of the primitive variables:

$$\mathbf{A}^{(x)}(\mathbf{q}) = \begin{pmatrix} u & 0 & 0 & \rho & 0 & 0 \\ 0 & u & 0 & 0 & 0 & 0 \\ 0 & 0 & u & 0 & 0 & 0 \\ 0 & 0 & 0 & u & 1/\rho & 0 \\ 0 & 0 & 0 & \Gamma_1 p & u & 0 \\ 0 & 0 & 0 & \rho h & 0 & u \end{pmatrix} \tag{13}$$

where h is the specific enthalpy and Γ_1 is an adiabatic index, $\Gamma_1 = d\log p/d\log\rho|_s$ at constant entropy. The CTU+PPM algorithm uses the characteristic wave structure of $\mathbf{A}$ to collect the information that makes it to an interface over a timestep in order to compute the fluxes through the interface. Note, the primitive state has two thermodynamic quantities, p and (ρe), to more efficiently handle the general equation of state in the Riemann solver, as described in Almgren et al. (2010), but alternate formulations are possible (Colella & Glaz 1985). The source term vector, $\mathbf{S}(\mathbf{q})$, can again be decomposed into hydrodynamic sources (now in terms of the primitive variables) and reaction terms,

$$\mathbf{S}(\mathbf{q}) = \mathbf{H}(\mathbf{q}) + \mathbf{R}(\mathbf{q}) \tag{14}$$

with

$$\mathbf{H}(\mathbf{q}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ g \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{R}(\mathbf{q}) = \begin{pmatrix} 0 \\ \dot\omega(X_k) \\ \dot\omega(\alpha_l) \\ 0 \\ \Gamma_1 p\sigma\dot S \\ \rho\dot S \end{pmatrix} \tag{15}$$

where

$$\sigma \equiv \frac{\partial p/\partial T|_\rho}{\rho c_p \partial p/\partial\rho|_T} \tag{16}$$

and c_p is the specific heat at constant pressure, $c_p = \partial h/\partial T|_p$. A derivation of this source for the pressure equation can be found in Almgren et al. (2008). We note that this source is algebraically identical to that shown in Eq. 25 of Almgren et al. (2010).

The CTU+PPM method for hydrodynamics is second-order accurate in space and time. We want to couple the reactive sources to the hydrodynamics to be second-order in time as well. As discussed above, nuclear reaction sources are stiff, and need to be integrated using implicit methods for stabilty. Operator splitting (e.g., Strang) is traditionally employed here, and is used as a benchmark for comparison in this paper. We'll discuss this traditional approach next before moving on to our new time-coupling method.

2.1. *Strang Splitting*

In Strang splitting, we first integrate the system with reactions terms only (no advection) over $\Delta t/2$, then integrate the advection terms only (no reactions) over Δt, and finally integrate the reaction terms only over $\Delta t/2$.

In the absence of advective terms, our reaction system appears as just $d\boldsymbol{\mathcal{U}}/dt = \mathbf{R}(\boldsymbol{\mathcal{U}})$, or:

$$\frac{d\rho}{dt} = 0 \tag{17}$$
$$\frac{d(\rho X_k)}{dt} = \rho\dot{\omega}(X_k) \tag{18}$$
$$\frac{d(\rho\alpha_l)}{dt} = \rho\dot{\omega}(\alpha_l) \tag{19}$$
$$\frac{d(\rho u)}{dt} = 0 \tag{20}$$
$$\frac{d(\rho E)}{dt} = \rho\dot{S} \tag{21}$$

We can write the energy equation as:

$$\frac{d(\rho E)}{dt} = \frac{d(\rho e)}{dt} + \frac{dK}{dt} = \rho\dot{S} \tag{22}$$

where K is the kinetic energy, $K = \rho|u|^2/2$. Since the density and velocity are unchanged by reactions (when Strang splitting), our energy equation becomes:

$$\frac{d(\rho e)}{dt} = \rho\frac{de}{dt} = \rho\dot{S} \tag{23}$$

The reaction rates are typically expressed as $\dot{\omega}_k(\rho, T, X_k)$ when we evolve this system, which requires us to get temperature from the equation of state each time we need to evaluate the reactive terms.

We also typically integrate mass fraction itself, instead of partial densities:

$$\frac{dX_k}{dt} = \dot{\omega}_k \tag{24}$$

We integrate Equations (23) and (24) using an implicit ODE solver designed for stiff systems of equations, VODE (Brown et al. 1989).

We explored this and other approaches (including not evolving an energy / temperature equation during the reaction step) in Zingale et al. (2021) and showed that the above formulation gets second order convergence. However, for very strong reactions, when using Strang splitting the state can drift significantly off of the smooth solution to the coupled reactive hydrodynamics equations, as shown graphically in Zingale et al. (2019) using an earlier version of the present algorithm.

A variation on Strang splitting called (re-)balanced splitting was developed in Speth et al. (2013).

2.2. *Timestep Limiters and Retry Mechanism*

Since this method is based off of the CTU hydrodynamics scheme, it benefits from the larger timestep that method can take (when done with full corner coupling, the advective CFL condition is unity) as compared to a method-of-lines approach (see Colella 1990). In addition to the standard CFL timestep limiter for explicit hydrodynamics, Castro can also enforce timestep limiters based on the energy generation or abundance changes over a timestep:

do we enforce this in the NSE region?

$$\Delta t \le f_e \min_i \left\{ \frac{e_i}{\dot{S}_i} \right\} \tag{25}$$

$$\Delta t \le f_X \min_i \left\{ \min_{k, X_k > \epsilon_X} \frac{X_{ki}}{\dot{\omega}(X_k)_i} \right\} \tag{26}$$

where i is the zone index and f_e and f_X are runtime parameters used to control the allowed change, and only species for which $X_k > \epsilon_X$ are considered.

Castro has the ability to reject a timestep if it detects a failure and retry with smaller timesteps (subcycling to make up the original required timestep). Among the conditions that can trigger this are density falling below zero during advection, the ODE integration failing to converge in the implicit solve, or violation of one of the timestep limiters during the step. This means that equations (25) and (26) are not reactive, but instead guaranteed to be met throughout the simulation, because the step is rejected if they are violated. This contrasts to similar approaches used in other codes where conditions 25 and 26 are used to restrict the next timestep size, but the update over the current step is kept, which already violated the constraints. The retry mechanism in Castro works with both the Strang and simplified-SDC integration scheme.

would be nice to have a reference here

2.3. *Spectral Deferred Corrections*

ref

The basic idea of spectral deferred corrections is to express the update as an integral and divide the time-update into a number of discrete time nodes. The integral is then approximated using a quadrature rule over these time nodes and low order approximations are used to update the state from one time node to the next. The method uses iteration to successively improve the solution and the ultimate accuracy is determined by the quadrature method used to evaluate the integral, which is done using an iteratively-lagged solution. This is the classic approach that we implemented in Zingale et al. (2019).

Our simplified-SDC approach is based on the method described in Nonaka et al. (2012) and a similar implementation in the Maestro simulation code. We again start by considering the update in integral form:

$$\boldsymbol{\mathcal{U}}^{n+1} = \boldsymbol{\mathcal{U}}^n + \int_t^{t+\Delta t} \left[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}}) + \mathbf{R}(\boldsymbol{\mathcal{U}})\right] dt, \tag{27}$$

In this approach, we approximate the advective term piecewise constant in time, using the value at the midpoint in time, to achieve second-order accuracy, giving us:

$$\boldsymbol{\mathcal{U}}^{n+1} = \boldsymbol{\mathcal{U}}^n + \int_t^{t+\Delta t} \left\{ \left[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})\right]^{n+1/2} + \mathbf{R}(\boldsymbol{\mathcal{U}}) \right\} dt \tag{28}$$

The Castro version of the simplified-SDC algorithm differs from the previous versions due to the need to do some operations on the conserved variable state and some on the primitive variable state.

The basic time update algorithm proceeds as:

- *Initialization*
  - We need an approximation of how much the reactions alone changed the primitive variable state over the timestep, which we will denote $\boldsymbol{\mathcal{I}}_q$ (this is basically an approximation of $\mathbf{R}(q)$).

    Since we do not have any information about the current timestep in the first iteration, we use the value from the last iteration of the previous timestep:

$$\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{n+1/2,(0)} = \boldsymbol{\mathcal{I}}_{\mathbf{q}}^{n-1/2,(s_{\max})} \tag{29}$$

- *Iterate*

  Iterate from $k = 1, \ldots, s_{\max}$. For second-order accuracy, $s_{\max} = 2$ is sufficient. In addition to denoting the time-level with a superscript (like n or $n+1$), we'll use a second subscript in parentheses to keep track of the iteration. A single iteration starts with $\boldsymbol{\mathcal{U}}^n$ and results in the new time-level state for that iteration, $\boldsymbol{\mathcal{U}}^{n+1,(k)}$.
  - *Create the advective update term,* $[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})]^{n+1/2,(k)}$
    * convert $\boldsymbol{\mathcal{U}} \rightarrow \mathbf{q}$. This is an algebraic transformation, but will require the EOS.
    * predict $\mathbf{q}$ to the interfaces at $t^{n+1/2}$ using the CTU PPM method. The source terms, $\mathbf{S_q}$, used in the prediction are:

$$\mathbf{S}(\mathbf{q}) = \mathbf{H}(\mathbf{q}) + \boldsymbol{\mathcal{I}}_{\mathbf{q}}^{n+1/2,(k-1)} \tag{30}$$

      Here we use the iteratively lagged integrals of the primitive variable terms accounting only for reactions, $\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{n+1/2,(k-1)}$, as the reactive source. This is in contrast to Strang-splitting, where no explicit reactive source terms are seen by the hydrodynamics update.

      In the unsplit CTU method (Colella 1990), the interface states used for the final Riemann problem through the zone interface consist of a normal predictor and a transverse flux correction. We can add the source terms either to the normal predictor (for example, doing characteristic tracing as described in Colella & Woodward 1984) or after all of the transverse flux corrections are made. Both are second-order accurate. For the hydrodynamics sources, we do those in the normal predictor. However, for the reactive sources, we found that it is most reliable to add them at the end of the interface state construction, after the transverse flux corrections. This is because we want to ensure that sum over species of $\boldsymbol{\mathcal{I}}_{\mathbf{q}}$ is zero, and characteristic tracing or the various flux corrections may not preserve this. We enforce that the species interface states remain in $[0, 1]$ after adding the reactive source.

- solve the Riemann problem at each interface to get a unique conserved state on each interface, $\boldsymbol{\mathcal{U}}_{i+1/2}^{n+1/2,(k)}$
- construct the advective update terms, $\left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]_i^{n+1/2,(k)}$, by first ignoring the hydrodynamics sources,

$$\left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]_i^{n+1/2,(k)} = -\frac{\mathbf{F}^{(x)}(\boldsymbol{\mathcal{U}}_{i+1/2}^{n+1/2,(k)}) - \mathbf{F}^{(x)}(\boldsymbol{\mathcal{U}}_{i-1/2}^{n+1/2,(k)})}{\Delta x} \tag{31}$$

for the general case, and

$$\left[\tilde{\mathcal{A}}(\rho e)\right]_i^{n+1/2,(k)} = -\frac{F\left((\rho e)_{i+1/2}^{n+1/2,(k)}\right) - F\left((\rho e)_{i-1/2}^{n+1/2,(k)}\right)}{\Delta x} \tag{32}$$

$$- \left(\frac{p_{i+1/2}^{n+1/2,(k)} + p_{i-1/2}^{n+1/2,(k)}}{2}\right) \frac{u_{i+1/2}^{n+1/2,(k)} - u_{i-1/2}^{n+1/2,(k)}}{\Delta x} \tag{33}$$

for (ρe).

Now the conservative hydrodynamics source terms are computed by first updating to the new state with advection and the old-time source term applied for the full Δt as:

$$\boldsymbol{\mathcal{U}}^{\star\star} = \boldsymbol{\mathcal{U}}^n + \Delta t\left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]^{n+1/2,(k)} + \Delta t\mathbf{H}(\boldsymbol{\mathcal{U}}^n) \tag{34}$$

We then evaluate the source terms with $\boldsymbol{\mathcal{U}}^{\star\star}$ and correct the advective term so that we have a time-centered source. The final advective update term is then[2]:

$$[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})]_i^{n+1/2,(k)} = \left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]_i^{n+1/2,(k)} + \frac{\Delta t}{2}\left(\mathbf{H}(\boldsymbol{\mathcal{U}}^n) + \mathbf{H}(\boldsymbol{\mathcal{U}}^{\star\star})\right) \tag{35}$$

We note that this procedure works because the source terms here do not depend on the outcome of the reactions.

– *Update the System Using a Method of Lines Integration*

We update the state by doing the integral in equation (28). Since we are approximating the advective term as piecewise constant in time, we can simply use an ODE integrator to integrate this, just as we do with the reaction system

[2] For a source like gravity, this update can be done first for ρ and then define the new momentum source using $\rho^{\star\star}$ and likewise for energy

in Strang splitting. The difference here being that we are integrating the conserved variables and the state sees the effect of advection as we integrate the reactions. So rather than use $d\boldsymbol{\mathcal{U}}/dt = \mathbf{R}(\boldsymbol{\mathcal{U}})$, the ODE form we use is

$$\frac{d\boldsymbol{\mathcal{U}}}{dt} = \left[\boldsymbol{\mathcal{A}}\left(\boldsymbol{\mathcal{U}}\right)\right]^{n+1/2,(k)} + \mathbf{R}(\boldsymbol{\mathcal{U}}) \tag{36}$$

Looking at the form of Eq. **??**, we see that only the species, auxiliary quantities, and energies have source terms. Since we are reacting, we will use the X_k as the primary composition variable and compute α_l as needed from them, so we don't directly integrate the auxiliary quantities. Likewise, the total and internal energies both provide the same information, and integrating both overconstrains the system, so we just integrate the internal energy. We define the subset of variables that are directly integrated as:

$$\boldsymbol{\mathcal{U}}' = \begin{pmatrix} \rho X_k \\ \rho e \end{pmatrix} \tag{37}$$

and we integrate

$$\frac{d\boldsymbol{\mathcal{U}}'}{dt} = \left[\boldsymbol{\mathcal{A}}\left(\boldsymbol{\mathcal{U}}'\right)\right]^{n+1/2,(k)} + \mathbf{R}(\boldsymbol{\mathcal{U}}') \tag{38}$$

This integration begins with $\boldsymbol{\mathcal{U}}'^{,n}$ and results in $\boldsymbol{\mathcal{U}}'^{,n+1,(k)}$.
We will need the density at times during the integration, which we construct as:

$$\rho(t) = \rho^n + \left[\mathcal{A}\left(\rho\right)\right]^{n+1/2,(k)} t \tag{39}$$

As we are integrating this system we need to get the temperature, T, for the rate evaluations. We obtain this by directly from internal energy, composition, and density using the equation of state.
Our integrator also needs the Jacobian of the system, in terms of the $\boldsymbol{\mathcal{U}}'$. This is different than the form of the Jacobian usually used in reaction networks. We describe the form of the Jacobian in appendix A.
At the end of the integration, we can do the conservative update of momentum and energy. Momentum is straightforward, since there are no reactive sources:

$$(\rho u)^{n+1} = (\rho u)^n + \Delta t[\mathcal{A}\left(\rho u\right)]^{n+1/2,(k)} \tag{40}$$

For total energy, we first need to isolate the reactive source for energy:

$$(\rho\dot{S})^{n+1/2} = \frac{(\rho e)^{n+1} - (\rho e)^n}{\Delta t} - \left[\mathcal{A}\left(\rho e\right)\right]^{n+1/2,(k)} \tag{41}$$

Then the update is

$$(\rho E)^{n+1} = (\rho E)^n + \Delta t[\mathcal{A}\left(\rho E\right)]^{n+1/2,(k)} + \Delta t(\rho\dot{S})^{n+1/2} \tag{42}$$

– *Compute the Reactive Source Terms.*

We now seek the $\boldsymbol{\mathcal{I}}$'s that capture the effect of just the reaction sources on the state variables for the next iteration. For the conserved quantities, these would simply be:

$$\boldsymbol{\mathcal{I}}_{\boldsymbol{\mathcal{U}}}^{(k)} = \frac{\boldsymbol{\mathcal{U}}^{n+1,(k)} - \boldsymbol{\mathcal{U}}^n}{\Delta t} - \left[\boldsymbol{\mathcal{A}}\left(\boldsymbol{\mathcal{U}}\right)\right]^{n+1/2,(k)} \tag{43}$$

However, for our primitive variables, which are used in the prediction, we need to construct the required source terms more carefully. We want:

$$\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)} = \frac{\mathbf{q}^{n+1,(k)} - \mathbf{q}^n}{\Delta t} - \left[\boldsymbol{\mathcal{A}}\left(\mathbf{q}\right)\right]^{n+1/2,(k)} \tag{44}$$

but we need the advective update for $\mathbf{q}$, which we have not constructed. We note that $\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)}$ is an approximation to the integral of Eq. 15 over the timestep. Additionally, we cannot simply use the equation of state on $\boldsymbol{\mathcal{I}}_{\boldsymbol{\mathcal{U}}}^{(k)}$ since this is a time-derivative and does not represent a well-defined state in itself. Instead, we derive $\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)}$ via a multi-step process. We first find the conservative state as if it were updated only with advection:

$$\boldsymbol{\mathcal{U}}^{\star} = \boldsymbol{\mathcal{U}}^n + \Delta t\left[\boldsymbol{\mathcal{A}}\left(\boldsymbol{\mathcal{U}}\right)\right]^{n+1/2,(k)} \tag{45}$$

and then construct the corresponding primitive variable state via an algebraic transform, $\boldsymbol{\mathcal{U}}^{\star} \rightarrow \mathbf{q}^{\star}$. This allows us to define the advective update for $\mathbf{q}$ as:

$$\left[\boldsymbol{\mathcal{A}}\left(\mathbf{q}\right)\right]^{n+1/2,(k)} = \frac{\mathbf{q}^{\star} - \mathbf{q}^n}{\Delta t} \tag{46}$$

Defining the primitive state corresponding to the fully-updated conserved state via an algebraic transform, $\boldsymbol{\mathcal{U}}^{n+1,(k)} \rightarrow \mathbf{q}^{n+1,(k)}$, we can construct $\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)}$ as given in Equation (44). Putting all of this together, we see:

$$\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)} = \frac{\mathbf{q}^{n+1,(k)} - \mathbf{q}^{\star}}{\Delta t} \tag{47}$$

3. REACTION NETWORKS AND NUCLEAR STATISTICAL EQUILIBRIUM

For this study we will use the 19 nuclei network containing ^{1}H, ^{3}He, ^{4}He, ^{12}C, ^{14}N, ^{16}O, ^{20}Ne, ^{24}Mg, ^{28}Si, ^{32}S, ^{36}Ar, ^{40}Ca, ^{44}Ti, ^{48}Cr, ^{52}Fe, ^{54}Fe, ^{56}Ni, protons (from photodisintegration), and neutrons. This is based on the "aprox19" network from **?** and originally described in Weaver et al. (1978). We use a modified version of the VODE (Brown et al. 1989) integrator—ported to C++ with checks added to the timestep rejection logic that ensure that the species mass fractions stay between 0 and 1[3]. We combine this with a table that gives the nuclear statistical equilibrium (NSE) abundances in regions where the system is in NSE. The NSE table was generated using a 127 nuclei reaction network and is the

we need to be clear that there is different logic for SDC

discuss tolerances

[3] This modified version of VODE is available in our Microphysics repo: https://github.com/starkiller-astro/Microphysics.

same as described in Ma et al. (2013). In our simulations, we carry all 19 isotopes in the main network in each zone and advect them in the hydrodynamics portion of the algorithm. The composition of the larger 127 nuclei network is mapped into the 19 isotopes we carry according to Table **??**.

need a statement about how the table was constructed and what NSE means

The NSE table requires:

$$Y_e = \sum_k \frac{Z_k X_k}{A_k} \tag{48}$$

(where A_k and Z_k are the atomic weight and atomic number of nucleus k) and provides

$$\bar{A} = \left[\sum_k \frac{X_k}{A_k}\right]^{-1} \tag{49}$$

$$\left(\frac{B}{A}\right) = \sum_k \frac{B_k X_k}{A_k} \tag{50}$$

where B_k is the binding energy of nucleus k. In our simulations, we store these 3 quantities as auxiliary data that is carried along with the rest of the fluid state in each zone. The table also returns values of the mass fractions mapped onto the 19-isotopes we carry, X_k, and the time-derivative of Y_e, dY_e/dt. We use the notation:

$$\mathtt{NSE}(\rho, T, Y_e) \rightarrow \bar{A}, X_k, (B/A), dY_e/dt \tag{51}$$

to represent the NSE table call and its inputs and outputs.

We can derive evolution equations for each of these composition quantities as:

$$\frac{DY_e}{Dt} = \sum_k \frac{Z_k}{A_k}\frac{DX_k}{Dt} = \sum_k \frac{Z_k}{A_k}\dot{\omega}_k \tag{52}$$

$$\frac{D\bar{A}}{Dt} = -\bar{A}^2 \sum_k \frac{1}{A_k}\frac{DX_k}{Dt} = -\bar{A}^2 \sum_k \frac{1}{A_k}\dot{\omega}_k \tag{53}$$

$$\frac{D}{Dt}\left(\frac{B}{A}\right) = \sum_k \frac{B_k}{A_k}\frac{DX_k}{Dt} = \sum_k \frac{B_k}{A_k}\dot{\omega}_k \tag{54}$$

For Strang split coupling of hydro and reactions, $DX_k/Dt = 0$, therefore each of these auxillary equations obeys an advection equation in the hydro part of the advancement. In the SDC algorithm, there will be a reactive source (an $\boldsymbol{\mathcal{I}}_q$) for each of these that is computed in the same manner as above. We note that our NSE table provides the evolution of Y_e due to reactions (DY_e/Dt) directly.

The compositional quantities it carries, $\bar{A}$ and Y_e are not representable from the 19 isotopes we carry in the main network. For this reason, when we are using the NSE network, we always provide these two composition quantities as inputs to the EOS rather than using the X_k directly. the EOS directly from the auxiliary state in each zone instead of using the X_k directly. Our equation of state needs the mean charge per nucleus, $\bar{Z}$, in addition to the auxiliary quantities, which is computed as

$$\bar{Z} = \bar{A}\sum_k \frac{Z_k X_k}{A_k} = \bar{A} Y_e \tag{55}$$

(see, e.g., Fryxell et al. 2000).

3.1. *Initialization*

We initialize the mass fractions, X_k, and then compute the electron fraction, Y_e, using (48). For the two other composition quantities we carry, $\bar{A}$, and (B/A), we need values that are consistent with the value of Y_e and the nuclei stored in the NSE table. Therefore, if the thermodynamic conditions put us in NSE (using the conditions defined below), then we obtain $\bar{A}$ and (B/A) from the NSE table. Otherwise, we compute these directly from X_k using (49) and (50).

3.2. *NSE condition*

We treat a zone as being in NSE if the density and temperature exceed some threshold and the composition is mainly α and Fe-group nuclei, where the Fe-group nuclei are ^{48}Cr, ^{52}Fe, ^{54}Fe, and ^{56}Ni. The full condition is:

$$\rho > \rho_{\rm nse} \tag{56}$$
$$T > T_{\rm nse} \tag{57}$$
$$X(^{12}{\rm C}) < A_{\rm nse} \tag{58}$$
$$X(^{4}{\rm He}) + \sum_{k\in {\rm Fe-group}} X_k > B_{\rm nse} \tag{59}$$
$$X(^{26}{\rm Si}) < C_{\rm nse} \tag{60}$$

Typical values are $\rho_{\rm nse} = 2 \times 10^6$ g cm^{-3}, $T_{\rm nse} = 3 \times 10^9$ K, $A_{\rm nse} = 0.12$, $B_{\rm nse} = 0.88$, and $C_{\rm nse} = 0.01$. The values of $A_{\rm nse}$ and $B_{\rm nse}$ are based on Ma et al. (2013). The value of $C_{\rm nse}$ ensures that Si-burning has ended before invoking the table.

3.3. *Strang-split algorithm for NSE*

For Strang splitting, we alternate the reaction and hydrodynamics, with each operation working on the output of the previous operation. If we define an advection operator over a timestep Δt acting on $\boldsymbol{\mathcal{U}}$ as $\mathbb{A}_{\Delta t}(\boldsymbol{\mathcal{U}})$ and a reaction operator over a timestep of $\Delta t/2$ acting on $\boldsymbol{\mathcal{U}}$ as $\mathbb{R}_{\Delta t/2}(\boldsymbol{\mathcal{U}})$, then the advance appears as:

$$\boldsymbol{\mathcal{U}}^{\star} = \mathbb{R}_{\Delta t/2}(\boldsymbol{\mathcal{U}}^n) \tag{61}$$
$$\boldsymbol{\mathcal{U}}^{n+1,\star} = \mathbb{A}_{\Delta t}(\boldsymbol{\mathcal{U}}^{\star}) \tag{62}$$
$$\boldsymbol{\mathcal{U}}^{n+1} = \mathbb{R}_{\Delta t/2}(\boldsymbol{\mathcal{U}}^{n+1,\star}) \tag{63}$$

Tthe hydrodynamic and reactive substeps over the overall time-advancemenet scheme using the aprox19 + NSE network are as follows:

- *Hydrodynamics update*

  At the beginning of each hydrodynamic update we have an input state, $\boldsymbol{\mathcal{U}}_{\rm in}$ that we wish to integrate over Δt to obtain $\boldsymbol{\mathcal{U}}_{\rm out}$ The hydrodynamics update proceeds as normal, but with an advection equation for each of the auxiliary composition variables:

$$\frac{\partial(\rho Y_e)}{\partial t} + \frac{\partial(\rho Y_e u)}{\partial x} = 0 \quad (64)$$

$$\frac{\partial(\rho \bar{A})}{\partial t} + \frac{\partial(\rho \bar{A} u)}{\partial x} = 0 \quad (65)$$

$$\frac{\partial[\rho(B/A)]}{\partial t} + \frac{\partial[\rho(B/A)u]}{\partial x} = 0 \quad (66)$$

- *Reactive update*

  At the beginning of each reactive update we have an input state, $\boldsymbol{\mathcal{U}}_{\rm in}$ that we wish to integrate over $\Delta t/2$ to obtain $\boldsymbol{\mathcal{U}}_{\rm out}$.

  - For a zone that is in NSE:

    The goal is to update the composition and thermodynamics due to the change in the nuclei abundances over the (half-)timestep. Ma et al. (2013) uses a first-order in time difference to get the new composition state and evaluates the energy release from the change in binding energy in the NSE state. They only apply a fraction (0.3) of the energy release to the internal energy in a zone, to avoid a potential instability that can arise if too much energy is added to a zone in a single timestep. We prefer to deal with this issue through the retry mechanism already built into Castro.

    The approach we use begins with calling the NSE table with the input state:

$$\texttt{NSE}(\rho_{\rm in}, T_{\rm in}, (Y_e)_{\rm in}) \rightarrow \bar{A}^\star, (X_k)^\star, (B/A)^\star, dY_e/dt \quad (67)$$

    We indicate with a $\star$ that most of those values are provisional, and we will seek a better value in the corrector step. We only update Y_e from this call:

$$(Y_e)_{\rm out} = (Y_e)_{\rm in} + \Delta t \frac{dY_e}{dt} \quad (68)$$

    and then we use the table again, but with the updated Y_e (note that $\rho_{\rm out} = \rho_{\rm in}$ in the Strang reaction formulation):

$$\texttt{NSE}(\rho_{\rm out}, T_{\rm in}, (Y_e)_{\rm out}) \rightarrow \bar{A}_{\rm out}, (X_k)_{\rm out}, (B/A)_{\rm out}, dY_e/dt \quad (69)$$

    In this formulation, we have not updated the temperature. This corrects the composition so that it is consistent with the updated Y_e. We use these values $\bar{A}_{\rm out}$, $(X_k)_{\rm out}$, and $(B/A)_{\rm out}$ to then complete the update, computing the energy release, $\dot{S}$ as:

$$\dot{S} = \left[\left(\frac{B}{A}\right)_{\rm out} - \left(\frac{B}{A}\right)_{\rm in}\right] N_A \frac{1}{\Delta t} \quad (70)$$

  - For zones not in NSE:

discuss this more, and perhaps switch to a mode where we try to predict T

we could imagine looping over the EOS to get an updated T and then the table to get the updated $\bar{A}$, to see if that converges?

- $*$ Integrate the full reaction network (Eqs. 23 and 24) as usual for Strang splitting
- $*$ Update the aux quantities at the end of the burn using Eqs. 48, 49, and 50 with the new mass fractions, X_k.

3.4. *SDC-NSE Coupling*

With SDC evolution, when we are in NSE, we need to do the advective and reactive updates together. We want to compute:

$$\boldsymbol{\mathcal{U}}^{n+1,(k)} = \boldsymbol{\mathcal{U}}^n + \Delta t[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})]^{n+1/2,(k)} + \Delta t\,[\mathbf{R}(\boldsymbol{\mathcal{U}})]^{n+1/2,(k)} \tag{71}$$

For density and momentum, we can do this update already, since there are no reactive sources. That gives us $\rho^{n+1,(k)}$ and $(\rho\mathbf{U})^{n+1,(k)}$. For the other quantities, the NSE table gives only the instantaneous values. aside from Y_e. We therefore do an iterative update.

Calling the NSE table on the starting state gives us:

$$\mathtt{NSE}(\rho^n, T^n, (Y_e)^n) \rightarrow \bar{A}^n, (X_k)^n, (B/A)^n, (dY_e/dt)^n \tag{72}$$

We now compute a first approximation to the reactive source:

$$R^{(0)}(\rho e) = 0 \tag{73}$$
$$R^{(0)}(\rho Y_e) = \rho^n (dY_e/dt)^n \tag{74}$$
$$R^{(0)}(\rho \bar{A}) = 0 \tag{75}$$

Now we iterate $\xi = 0, \ldots N-1$, doing the following:

- Update the energy as:

$$(\rho e)^{n+1,(\xi)} = (\rho e)^n + \Delta t[\mathcal{A}(\rho e)]^{n+1/2,(k)} + \Delta t R^{(\xi)}(\rho e) \tag{76}$$

- Update the auxiliary data as:

$$(\rho\alpha_l)^{n+1,(\xi)} = (\rho\alpha_l)^n + \Delta t[\mathcal{A}(\rho\alpha_l)]^{n+1/2,(k)} + \Delta t R^{(\xi)}(\rho\alpha_l) \tag{77}$$

- Compute the updated temperature, $T^{n+1,(k)}$ as

$$T^{n+1,(\xi)} = T(\rho^{n+1,(k)}, \alpha^{n+1,(\xi)}, e^{n+1,(\xi)}) \tag{78}$$

  where $\alpha^{n+1,(\xi)} = (\rho\alpha)^{n+1,(\xi)}/\rho^{n+1,(k)}$ and $e^{n+1,(\xi)} = (\rho e)^{n+1,(\xi)}/\rho^{n+1,(k)}$.

- Call the NSE table to get the new NSE state

$$\mathtt{NSE}(\rho^{n+1,(k)}, T^{n+1,(\xi)}, (Y_e)^{n+1,(\xi)}) \rightarrow \bar{A}^{n+1,(\xi)}, (X_k)^{n+1,(\xi)}, (B/A)^{n+1,(\xi)}, (dY_e/dt)^{n+1,(\xi)} \tag{79}$$

- Recompute the reactive source terms for the next iteration.

  The energy generation rate is found as:

$$(\rho\dot{S})^{n+1/2,(\xi)} = \frac{\rho^{n+1,(k)} + \rho^n}{2} \left[\left(\frac{B}{A}\right)^{n+1,(\xi)} - \left(\frac{B}{A}\right)^n \right] N_A \frac{1}{\Delta t} \tag{80}$$

  and our new auxiliary sources are:

$$R^{(\xi+1)}(\rho e) = (\rho\dot{S})^{n+1/2,(\xi)} \tag{81}$$

$$R^{(\xi+1)}(\rho Y_e) = \frac{\rho^{n+1,(k)} + \rho^n}{2} \frac{1}{2} \left[\left(\frac{dY_e}{dt}\right)^n + \left(\frac{dY_e}{dt}\right)^{n+1,(\xi)} \right] \tag{82}$$

$$R^{(\xi+1)}(\rho\bar{A}) = \frac{\rho^{n+1,(k)} + \rho^n}{2} \frac{1}{\Delta t} \left[\bar{A}^{n+1,(\xi)} - \frac{(\rho\bar{A})^n}{\rho^n} \right] \tag{83}$$

With the iteration complete, we can do the final update of the total energy using last iterations prediction of the source (for iteration N):

$$(\rho E)^{n+1,(k)} = (\rho E)^n + \Delta t[\mathcal{A}\,(\rho E)]^{n+1/2} + \Delta t(\rho\dot{S})^{n+1/2,(N)} \tag{84}$$

and we take the values of the auxiliary state and mass fractions from the last call to the NSE table:

$$(\rho Y_e)^{n+1,(k)} = \rho^{n+1,(k)}(Y_e)^{n+1,(N-1)} \tag{85}$$

$$(\rho\bar{A})^{n+1,(k)} = \rho^{n+1,(k)}(\bar{A})^{n+1,(N-1)} \tag{86}$$

$$(\rho(B/A))^{n+1,(k)} = \rho^{n+1,(k)}(B/A)^{n+1,(N-1)} \tag{87}$$

$$(\rho Y_e)^{n+1,(k)} = \rho^{n+1,(k)}(X_k)^{n+1,(N-1)} \tag{88}$$

We note that this update is not fully second order, however, the treatment of Y_e is, and we expect that to have the dominant effect in the thermodynamic evolution through NSE.

3.5. *NSE Bailout*

There is one additional part of the time integration algorithm. Because T evolves during the reactions (for both Strang and SDC), it is possible for a zone to start out not in NSE but evolve into NSE during the reaction update. When this happens, the ODE integrator may fail, because an excessive number of timesteps is required. Rather than trigger our retry mechanism and throw away the entire timestep and start over with a smaller Δt, we first check if the zone evolved into NSE, and if so, we redo the zone update using the NSE prescription. To ensure that this is not triggered at the very start of integration, we require a minimum number of steps (10) to have elapsed before checking for NSE. After we leave the integrator, the state is returned to the burner driver where we finish the integration via NSE if the conditions satisfy the NSE criteria. We optionally apply a relaxation factor, typically, 0.9, to the NSE conditions to allow a state that is close to NSE after an integrator failure enter NSE.

Table 1. Convergence (L_1 norm) for the reacting convergence problem using Strang splitting.

field	$\epsilon_{64\rightarrow128}$	rate	$\epsilon_{128\rightarrow256}$	rate	$\epsilon_{256\rightarrow512}$
ρ	2.794×10^{18}	2.044	6.777×10^{17}	2.554	1.154×10^{17}
ρu	6.796×10^{26}	2.448	1.245×10^{26}	2.889	1.681×10^{25}
ρv	6.796×10^{26}	2.448	1.245×10^{26}	2.889	1.681×10^{25}
ρE	2.451×10^{35}	2.351	4.803×10^{34}	2.742	7.179×10^{33}
ρe	2.261×10^{35}	2.320	4.526×10^{34}	2.821	6.403×10^{33}
T	2.237×10^{21}	1.691	6.927×10^{20}	2.482	1.240×10^{20}
$\rho X(^{4}\mathrm{He})$	2.878×10^{18}	2.020	7.096×10^{17}	2.529	1.229×10^{17}
$\rho X(^{12}\mathrm{C})$	1.698×10^{17}	1.950	4.393×10^{16}	2.232	9.353×10^{15}
$\rho X(^{16}\mathrm{O})$	1.687×10^{14}	1.660	5.338×10^{13}	1.957	1.375×10^{13}
$\rho X(^{56}\mathrm{Fe})$	2.794×10^{8}	2.044	6.777×10^{7}	2.554	1.154×10^{7}

Table 2. Convergence (L_1 norm) for the reacting convergence problem using SDC integration (2 iterations).

field	$\epsilon_{64\rightarrow128}$	rate	$\epsilon_{128\rightarrow256}$	rate	$\epsilon_{256\rightarrow512}$
ρ	2.795×10^{18}	2.044	6.777×10^{17}	2.554	1.154×10^{17}
ρu	6.798×10^{26}	2.449	1.245×10^{26}	2.889	1.680×10^{25}
ρv	6.798×10^{26}	2.449	1.245×10^{26}	2.889	1.680×10^{25}
ρE	2.452×10^{35}	2.352	4.804×10^{34}	2.742	7.179×10^{33}
ρe	2.262×10^{35}	2.321	4.527×10^{34}	2.822	6.404×10^{33}
T	2.236×10^{21}	1.691	6.927×10^{20}	2.481	1.240×10^{20}
$\rho X(^{4}\mathrm{He})$	2.879×10^{18}	2.020	7.096×10^{17}	2.529	1.229×10^{17}
$\rho X(^{12}\mathrm{C})$	1.697×10^{17}	1.950	4.393×10^{16}	2.231	9.357×10^{15}
$\rho X(^{16}\mathrm{O})$	1.686×10^{14}	1.659	5.338×10^{13}	1.957	1.375×10^{13}
$\rho X(^{56}\mathrm{Fe})$	2.795×10^{8}	2.044	6.777×10^{7}	2.554	1.154×10^{7}

4. SIMULATIONS

4.1. *Reacting convergence test problem*

Zingale et al. (2019) introduced a test problem for measuring convergence of a reacting hydrodynamic algorithm. We run that same test here with the simplified SDC algorithm.

Also explore: PLM vs PPM, 1 vs. 2 vs. 3 iterations

4.2. *NSE convergence test problem*

To test the coupling of the NSE table to the hydrodynamics, we run a similar problem as above, except with the thermodynamic conditions appropriate for the matter to be in NSE.

Table 3. NSE convergence test problem.

parameter	value
ρ_0	5×10^8 g cm^{-3}
T_0	4×10^9 K
(δT)	0.2
$(Y_e)_0$	0.5
(δY_e)	-0.05
λ	2×10^7 cm
L	10^8 cm

The initial conditions are:

$$\rho = \rho_0 \tag{89}$$

$$T = T_0 \left[1 + (\delta T) e^{-(r/\lambda)^2} \cos^6(\pi r/L)\right] \tag{90}$$

$$Y_e = (Y_e)_0 \left[1 + (\delta Y_e) e^{-(r/\lambda)^2} \cos^6(\pi r/L)\right] \tag{91}$$

where r is the distance from the center of the domain and the remaining parameters are listed in Table 3. This has Y_e varying between $[0.47, 0.5]$ initially. The mass fractions and remaining composition variables are then initialized from the NSE table, as described in section 2.3.

check

The domain has a size $[0, L]^2$, and the timestep is fixed as:

$$\Delta t = 10^{-3} \left(\frac{64}{N}\right) \text{ s} \tag{92}$$

where N is the number of zones in each direction.

Figure 1 shows the Y_e profile for the Strang evolution case.

4.3. *Detonation*

The purpose here is to look at what timestep is taken when we use the nuclear burning limiters. Run with aprox19 only, Strang and SDC, then aprox19 + NSE, Strang and NSE.

Look at number of SDC iterations too.

How does the timestep limiter work in the middle of SDC iterating?

aprox21

look at number of RHS calls, wallclock time

output every step and then for a single zone plot enuc every timestep to see if it oscillates

4.4. *Reacting Buoyant Bubble*

We next consider a buoyant, reacting bubble in a stratified, plane-parallel atmosphere. This test exercises the treatment of hydrodynamical sources in the SDC integration. We use the set of initial conditions that was first shown in Almgren et al. (2008).

Table 4. Convergence (L_1 norm) for the NSE convergence problem using Strang splitting.

field	$\epsilon_{64\to128}$	rate	$\epsilon_{128\to256}$	rate	$\epsilon_{256\to512}$
ρ	4.522×10^{19}	1.659	1.432×10^{19}	1.542	4.917×10^{18}
ρu	1.883×10^{28}	1.681	5.873×10^{27}	1.524	2.042×10^{27}
ρv	1.883×10^{28}	1.681	5.873×10^{27}	1.524	2.042×10^{27}
ρE	5.933×10^{37}	1.613	1.940×10^{37}	1.411	7.293×10^{36}
ρe	5.931×10^{37}	1.613	1.939×10^{37}	1.411	7.291×10^{36}
T	9.885×10^{20}	1.207	4.281×10^{20}	1.058	2.057×10^{20}
$\rho X(^1\mathrm{H})$	4.522×10^{-11}	1.659	1.432×10^{-11}	1.542	4.917×10^{-12}
$\rho X(^4\mathrm{He})$	5.086×10^{17}	1.430	1.887×10^{17}	1.187	8.288×10^{16}
$\rho X(^{12}\mathrm{C})$	1.585×10^{12}	1.829	4.462×10^{11}	1.433	1.652×10^{11}
$\rho X(^{16}\mathrm{O})$	8.060×10^{12}	1.706	2.470×10^{12}	1.347	9.710×10^{11}
$\rho X(^{48}\mathrm{Cr})$	1.149×10^{19}	1.400	4.353×10^{18}	1.258	1.820×10^{18}
$\rho X(^{52}\mathrm{Fe})$	9.018×10^{19}	1.173	4.001×10^{19}	1.127	1.832×10^{19}
$\rho X(^{54}\mathrm{Fe})$	5.036×10^{20}	1.731	1.517×10^{20}	1.727	4.582×10^{19}
$\rho X(^{56}\mathrm{Ni})$	5.686×10^{20}	1.622	1.847×10^{20}	1.563	6.252×10^{19}
ρY_e	2.379×10^{19}	1.678	7.431×10^{18}	1.578	2.489×10^{18}
$\rho\bar{A}$	5.141×10^{21}	1.230	2.191×10^{21}	1.087	1.032×10^{21}
$\rho(B/A)$	4.149×10^{20}	1.693	1.283×10^{20}	1.562	4.345×10^{19}

Table 5. Convergence (L_1 norm) for the NSE convergence problem using the simplified-SDC algorithm.

field	$\epsilon_{64\to128}$	rate	$\epsilon_{128\to256}$	rate	$\epsilon_{256\to512}$
ρ	4.553×10^{19}	1.636	1.465×10^{19}	1.583	4.891×10^{18}
ρu	1.949×10^{28}	1.650	6.210×10^{27}	1.513	2.176×10^{27}
ρv	1.949×10^{28}	1.650	6.210×10^{27}	1.513	2.176×10^{27}
ρE	5.691×10^{37}	1.622	1.848×10^{37}	1.561	6.266×10^{36}
ρe	5.689×10^{37}	1.622	1.848×10^{37}	1.561	6.265×10^{36}
T	5.778×10^{20}	1.740	1.729×10^{20}	1.702	5.316×10^{19}
$\rho X(^1\mathrm{H})$	4.553×10^{-11}	1.636	1.465×10^{-11}	1.583	4.891×10^{-12}
$\rho X(^4\mathrm{He})$	2.541×10^{17}	1.807	7.262×10^{16}	1.634	2.339×10^{16}
$\rho X(^{12}\mathrm{C})$	1.140×10^{12}	1.838	3.189×10^{11}	1.670	1.002×10^{11}
$\rho X(^{16}\mathrm{O})$	5.413×10^{12}	1.819	1.535×10^{12}	1.658	4.864×10^{11}
$\rho X(^{48}\mathrm{Cr})$	7.020×10^{18}	1.793	2.026×10^{18}	1.614	6.621×10^{17}
$\rho X(^{52}\mathrm{Fe})$	3.631×10^{19}	1.579	1.215×10^{19}	1.597	4.018×10^{18}
$\rho X(^{54}\mathrm{Fe})$	4.670×10^{20}	1.807	1.335×10^{20}	1.869	3.654×10^{19}
$\rho X(^{56}\mathrm{Ni})$	4.858×10^{20}	1.808	1.387×10^{20}	1.860	3.820×10^{19}
ρY_e	2.372×10^{19}	1.649	7.564×10^{18}	1.587	2.518×10^{18}
$\rho\bar{A}$	4.009×10^{21}	1.693	1.240×10^{21}	1.690	3.843×10^{20}
$\rho(B/A)$	4.222×10^{20}	1.666	1.330×10^{20}	1.605	4.374×10^{19}

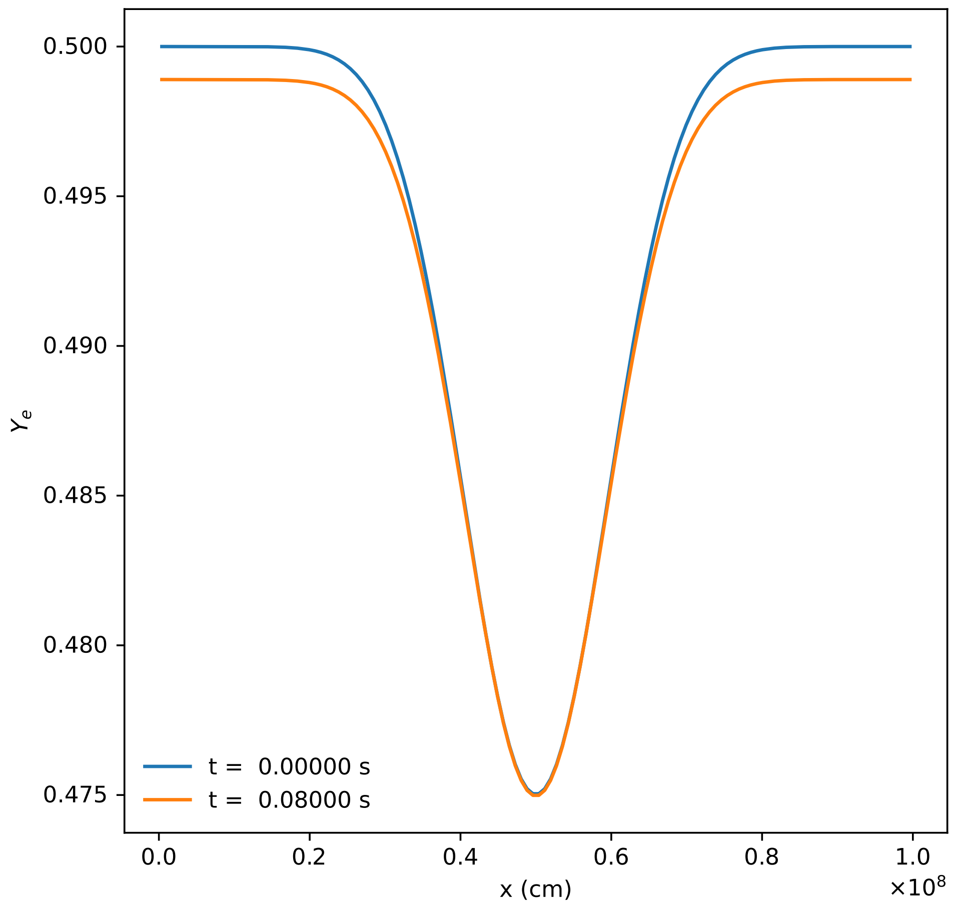

Figure 1. Profile of Y_e through the middle of the domain (slicing along x) for the 128^2 Strang NSE test simulation.

4.5. *Massive Star*

Do this in 1-d Show the amount of work needed for Strang vs. SDC

Play around with the threshold where NSE kicks in

5. SUMMARY

We presented a simplified spectral deferred corrections scheme for coupling hydrodynamics and reactions.

Future work is to extend this methodology to MHD.

ACKNOWLEDGMENTS

Castro is freely available at http://github.com/AMReX-Astro/Castro. All of the code and problem setups used here are available in the git repo. The work at Stony Brook was supported by DOE/Office of Nuclear Physics grant DE-FG02-87ER40317. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of Nuclear Physics, Scientific Discovery through Advanced Computing (SciDAC) program under Award Number DE-SC0017955. This research was 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.

Software: AMReX (Zhang et al. 2019), Castro (Almgren et al. 2010), GNU Compiler Collection (https://gcc.gnu.org/), Linux (https://www.kernel.org), matplotlib (Hunter 2007, http://matplotlib.org/) NumPy (Oliphant 2007; van der Walt et al. 2011), python (https://www.python.org/), SymPy (Meurer et al. 2017)

APPENDIX

A. JACOBIAN

To solve the reaction system implicitly, the ODE solver needs the Jacobian, $\partial \mathbf{R}/\partial \boldsymbol{\mathcal{U}}'$, where $\boldsymbol{\mathcal{U}}' = (\rho X_k, \rho e)^\intercal$ is the subset of the conserved variables we are integrating. We follow the method of Zingale et al. (2019) and factor this into two pieces,

$$\mathbf{J} = \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \boldsymbol{\mathcal{U}}'}. \tag{A1}$$

where the state $\mathbf{w}$ is chosen to match the set of variables used to evaluate the reaction rates. Writing this out for two species, X_α and X_β, we have

$$\boldsymbol{\mathcal{U}}' = \begin{pmatrix} \rho X_\alpha \\ \rho X_\beta \\ \rho e \end{pmatrix} \tag{A2}$$

For interfacing with the reaction network, we use

$$\mathbf{w} = \begin{pmatrix} X_\alpha \\ X_\beta \\ T \end{pmatrix} \tag{A3}$$

Note: even though we are using T here instead of e, we still do the overall ODE integration in terms of (ρe), consistent with the Strang method described in Zingale et al. (2021).

The Jacobian transformation $\partial \boldsymbol{\mathcal{U}}'/\partial \mathbf{w}$ is:

$$\frac{\partial \boldsymbol{\mathcal{U}}'}{\partial \mathbf{w}} = \begin{pmatrix} \rho & 0 & 0 \\ 0 & \rho & 0 \\ \rho e_{X_\alpha} & \rho e_{X_\beta} & \rho c_v \end{pmatrix} \tag{A4}$$

where we use the following notation for compactness:

$$e_{X_k} = \left.\frac{\partial e}{\partial X_k}\right|_{\rho,T,X_{j,j\neq k}} \tag{A5}$$

and the specific heat at constant volume is

$$c_v = \left.\frac{\partial e}{\partial T}\right|_{\rho,X_k} \tag{A6}$$

We get the inverse (computed via SymPy) as:

$$\frac{\partial \mathbf{w}}{\partial \boldsymbol{\mathcal{U}}'} = \begin{pmatrix} \frac{1}{\rho} & 0 & 0 \\ 0 & \frac{1}{\rho} & 0 \\ -\frac{e_{X_\alpha}}{\rho c_v} & -\frac{e_{X_\beta}}{\rho c_v} & \frac{1}{\rho c_v} \end{pmatrix} \tag{A7}$$

The reaction vector is

$$\mathbf{R}(\boldsymbol{\mathcal{U}}') = \begin{pmatrix} \rho\dot{\omega}_\alpha \\ \rho\dot{\omega}_\beta \\ \rho\dot{S} \end{pmatrix} \tag{A8}$$

We take all the quantities to be functions of ρ, e, and X_k, but since ρ doesn't change from the reactions (the reactive source of the continuity equation is zero), it is held constant in these derivatives. The Jacobian is computed as $\partial\mathbf{R}/\partial\mathbf{w}$:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{w}} = \begin{pmatrix} \rho\frac{\partial\dot{\omega}_\alpha}{\partial X_\alpha} & \rho\frac{\partial\dot{\omega}_\alpha}{\partial X_\beta} & \rho\frac{\partial\dot{\omega}_\alpha}{\partial T} \\ \rho\frac{\partial\dot{\omega}_\beta}{\partial X_\alpha} & \rho\frac{\partial\dot{\omega}_\beta}{\partial X_\beta} & \rho\frac{\partial\dot{\omega}_\beta}{\partial T} \\ \rho\frac{\partial\dot{S}}{\partial X_\alpha} & \rho\frac{\partial\dot{S}}{\partial X_\beta} & \rho\frac{\partial\dot{S}}{\partial T} \end{pmatrix} \tag{A9}$$

The final Jacobian is found by multiplying these two:

$$\frac{\partial \mathbf{R}}{\partial \boldsymbol{\mathcal{U}}'} = \begin{pmatrix} \frac{\partial\dot{\omega}_\alpha}{\partial X_\alpha} - \frac{e_{X_\alpha}}{c_v}\frac{\partial\dot{\omega}_\alpha}{\partial T} & \frac{\partial\dot{\omega}_\alpha}{\partial X_\beta} - \frac{e_{X_\beta}}{c_v}\frac{\partial\dot{\omega}_\alpha}{\partial T} & \frac{1}{c_v}\frac{\partial\dot{\omega}_\alpha}{\partial T} \\ \frac{\partial\dot{\omega}_\beta}{\partial X_\alpha} - \frac{e_{X_\alpha}}{c_v}\frac{\partial\dot{\omega}_\beta}{\partial T} & \frac{\partial\dot{\omega}_\beta}{\partial X_\beta} - \frac{e_{X_\beta}}{c_v}\frac{\partial\dot{\omega}_\beta}{\partial T} & \frac{1}{c_v}\frac{\partial\dot{\omega}_\beta}{\partial T} \\ \frac{\partial\dot{S}}{\partial X_\alpha} - \frac{e_{X_\alpha}}{c_v}\frac{\partial\dot{S}}{\partial T} & \frac{\partial\dot{S}}{\partial X_\beta} - \frac{e_{X_\beta}}{c_v}\frac{\partial\dot{S}}{\partial T} & \frac{1}{c_v}\frac{\partial\dot{S}}{\partial T} \end{pmatrix} \tag{A10}$$

We note that the form of these entries is the same as one would arrive at if you start with the rates expressed as $\omega_k(\rho, T(\rho, X_j, e), X_j)$ and note that constant e implies that

$$\left.\frac{\partial T}{\partial X_k}\right|_{\rho,e} = -\frac{e_{X_k}}{c_v} \tag{A11}$$

While VODE can compute the entire Jacobian, $\mathbf{J}$ numerically via differencing, we found that does not give reliable results. Instead, we compute compute the derivatives with respect to X_k and T one-sided differencing, following the algorithm in Radhakrishnan & Hindmarsh (1993) to minimize numerical noise. We then use the equation of state to compute e_{X_k} and c_v and construct the entries of the final Jacobian.

REFERENCES

Almgren, A., Sazo, M. B., Bell, J., et al. 2020, Journal of Open Source Software, 5, 2513, doi: 10.21105/joss.02513

Almgren, A. S., Bell, J. B., Nonaka, A., & Zingale, M. 2008, ApJ, 684, 449

Almgren, A. S., Beckner, V. E., Bell, J. B., et al. 2010, ApJ, 715, 1221, doi: 10.1088/0004-637X/715/2/1221

Brown, P. N., Byrne, G. D., & Hindmarsh, A. C. 1989, SIAM J. Sci. Stat. Comput., 10, 1038

Bryan, G. L., Norman, M. L., Stone, J. M., Cen, R., & Ostriker, J. P. 1995, Computer Physics Communications, 89, 149, doi: 10.1016/0010-4655(94)00191-4

Colella, P. 1990, Journal of Computational Physics, 87, 171, doi: 10.1016/0021-9991(90)90233-Q

Colella, P., & Glaz, H. M. 1985, J Comput Phys, 59, 264, doi: 10.1016/0021-9991(85)90146-9

Colella, P., & Woodward, P. R. 1984, Journal of Computational Physics, 54, 174, doi: 10.1016/0021-9991(84)90143-8

Fryxell, B., Olson, K., Ricker, P., et al. 2000, ApJS, 131, 273

Hunter, J. D. 2007, Computing in Science and Engg., 9, 90, doi: 10.1109/MCSE.2007.55

Katz, M. P., Zingale, M., Calder, A. C., et al. 2016, ApJ, 819, 94, doi: 10.3847/0004-637X/819/2/94

Katz, M. P., Almgren, A., Sazo, M. B., et al. 2020, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '20 (IEEE Press). https://dl.acm.org/doi/abs/10.5555/3433701.3433822

Ma, H., Woosley, S. E., Malone, C. M., Almgren, A., & Bell, J. 2013, ApJ, 771, 58, doi: 10.1088/0004-637X/771/1/58

McCorquodale, P., & Colella, P. 2011, Communications in Applied Mathematics and Computational Science, 6, 1

Meurer, A., Smith, C. P., Paprocki, M., et al. 2017, PeerJ Computer Science, 3, e103, doi: 10.7717/peerj-cs.103

Miller, G. H., & Colella, P. 2002, Journal of Computational Physics, 183, 26, doi: 10.1006/jcph.2002.7158

Nonaka, A., Bell, J. B., Day, M. S., et al. 2012, Combustion Theory and Modelling, 16, 1053, doi: 10.1080/13647830.2012.701019

Oliphant, T. E. 2007, Computing in Science and Engg., 9, 10, doi: 10.1109/MCSE.2007.58

Radhakrishnan, K., & Hindmarsh, A. C. 1993, Lawrence Livermore National Laboratory Report UCRL-ID-113855, 124

Speth, R., Green, W., MacNamara, S., & Strang, G. 2013, SIAM Journal on Numerical Analysis, 51, 3084, doi: 10.1137/120878641

van der Walt, S., Colbert, S. C., & Varoquaux, G. 2011, Computing in Science & Engineering, 13, 22, doi: 10.1109/MCSE.2011.37

Weaver, T. A., Zimmerman, G. B., & Woosley, S. E. 1978, ApJ, 225, 1021, doi: 10.1086/156569

Zhang, W., Almgren, A., Beckner, V., et al. 2019, Journal of Open Source Software, 4, 1370, doi: 10.21105/joss.01370

Zingale, M., Katz, M. P., Bell, J. B., et al. 2019, arXiv e-prints, arXiv:1908.03661. https://arxiv.org/abs/1908.03661

Zingale, M., Katz, M. P., Willcox, D. E., & Harpole, A. 2021, Research Notes of the AAS, 5, 71, doi: 10.3847/2515-5172/abf3cb

Zingale, M., Timmes, F. X., Fryxell, B., et al. 2001, ApJS, 133, 195, doi: 10.1086/319182

Zingale, M., Eiden, K., Cavecchi, Y., et al. 2019, Journal of Physics: Conference Series, 1225, 012005, doi: 10.1088/1742-6596/1225/1/012005