DRAFT VERSION DECEMBER 17, 2020
Typeset using LATEX **modern** style in AASTeX62

So you've decided to detonate helium

ABSTRACT

Helium detonations can be challenging to model because of the difficulty in accurately coupling hydrodynamics and reactions. This is particularly acute in the ash of the detonation where nuclear statistical equilibrium is established. We look a several different methods of coupling explicit hydrodynamics and stiff reaction sources, applied to detonations which reach NSE conditions. 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

Reactive flow can be challenging to model when the timescale for changes in the nuclear abundances due to reactions is comparable to the hydrodynamical timescales. Traditional methods of coupling hydrodynamics and reactions used in astrophysics use operator splitting—each physical process acts on the output of the previous process. This makes it easy to add physics in a modular way to a simulation code, but competition between physical processes can cause the coupling to breakdown. In general, the error in the splitting controlled by the timestep.

A particular difficult phase of evolution to model is the nuclear statistic equilibrium that sets in with helium burning for temperatures in excess of 4×10^9 K. Physically, the forward and reverse rates 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. This large change 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. example? GCD?

In this paper, we look at some different approaches to coupling hydrodynamics and reactions, using a helium detonation that attains NSE as our testbed. We use the Castro hydrodynamics code for all of our numerical experiments, and all of the code to reproduce the results in this paper are in the Castro github repository.

2. NUMERICAL METHODOLOGY

We solve the Euler equations for compressible, reacting flow. We'll focus on one-dimensional flow, and the system appears as:

$$\frac{\partial \boldsymbol{\mathcal{U}}_t}{\partial t} + \frac{\partial \mathbf{F}^{(x)}(\boldsymbol{\mathcal{U}})}{\partial x} = \mathbf{S}(\boldsymbol{\mathcal{U}}) \tag{1}$$

where $\boldsymbol{\mathcal{U}} = (\rho, (\rho X_k), (\rho u), (\rho E))^\intercal$ are the conserved fluid quantities: mass density, ρ, velocity, u, specific total energy, E, and nuclear species mass fractions, X_k. The specific total energy relates to the specific internal energy as $E = e + u^2/2$. The corresponding fluxes are

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

Here the pressure, p, enters, and is found via the equation of state,

$$p = p(\rho, X_k, e) \tag{3}$$

Finally, $\mathbf{S}(\boldsymbol{\mathcal{U}})$ are the source terms, which we decompose into hydrodynamical sources, $\mathbf{H}$ (like gravity), and reactive sources, $\mathbf{R}$,

$$\mathbf{S}(\boldsymbol{\mathcal{U}}) = \mathbf{H}(\boldsymbol{\mathcal{U}}) + \mathbf{R}(\boldsymbol{\mathcal{U}}) \tag{4}$$

In this paper, we will not consider any hydrodynamical sources, but we will carry the term $\mathbf{H}$ around for completeness. The reactive sources take the form:

$$\mathbf{R}(\boldsymbol{\mathcal{U}}) = \begin{pmatrix} 0 \\ \rho \dot{\omega}_k \\ 0 \\ \rho \dot{S} \end{pmatrix} \tag{5}$$

where $\dot{\omega}_k$ is the creation rate for species k and $\dot{S}$ is the energy generation rate per unit mass.

Sometimes it is preferable to work with the primitive variables, $\mathbf{q} = (\rho, X_k, u, p, (\rho e))^\intercal$. Here, the system appears as:

$$\mathbf{q}_t + \mathbf{A}^{(x)}(\mathbf{q})\mathbf{q}_x = \mathbf{S}_\mathbf{q} \tag{6}$$

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

$$\mathbf{A}^{(x)}(\mathbf{q}) = \begin{pmatrix} u & 0 & \rho & 0 & 0 \\ 0 & u & 0 & 0 & 0 \\ 0 & 0 & u & 1/\rho & 0 \\ 0 & 0 & \Gamma_1 p & u & 0 \\ 0 & 0 & \rho h & 0 & u \end{pmatrix} \tag{7}$$

where Γ_1 is an adiabatic index, $\Gamma_1 = d\log p/d\log\rho|_s$. 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, $\mathbf{S_q}$, can again be decomposed into hydrodynamic sources (now in terms of the primitive variables) and reaction terms,

$$\mathbf{S_q} = \mathbf{S_q}^{\rm hydro} + \mathbf{R}(q) \tag{8}$$

with

$$\mathbf{R}(q) = \begin{pmatrix} 0 \\ \dot{\omega}_k \\ 0 \\ \Gamma_1 p \sigma \dot{S} \\ \rho \dot{S} \end{pmatrix} \tag{9}$$

where

$$\sigma \equiv \frac{\partial p/\partial T|_\rho}{\rho c_p \partial p/\partial \rho|_T} \tag{10}$$

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).

We use the Castro hydrodynamics code Almgren et al. (2010), together with the corner transport upwind piecewise parabolic method hydrodynamics (Miller & Colella 2002). This is a finite-volume method that uses characteristic tracing to predict a time-centered flux through the interfaces of the grid zones. For pure hydrodynamics, these time-centered fluxes result in second-order accurate time integration. With reactions, we want to couple hydrodynamics and the reaction sources to second order. Nuclear reaction sources are stiff, and need to be integrated using implicit methods for stabilty. Operator splitting is traditionally employed here, and our goal in this paper is to look at several different approaches for splitting.

We describe several time-integration techniques for coupling hydrodynamics and reactions in the next sections.

2.1. *Strang Splitting*

2.2. *Timestep Limiters and Retry Mechanism*

In addition to the standard CFL timestep limiter for explicit hydrodynamics, Castro can also enforcing timestep limiters based on the energy generation or abundance changes over a timestep.

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 failing 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. At the moment, this retry mechanism only works with the CTU hydrodynamics + Strang splitting. We enable it here, since, as will be shown shortly, this combination of methods has the most difficulty in integrating the detonation.

2.3. *Balanced Splitting*

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

2.4. *Spectral Deferred Corrections*

The basic idea of spectral deferred corrections is break the timestep into a number of temporal nodes and use low order difference methods in a correction equation that, upon iteration, achieves a high-order solution coupling the different physics together.

2.5. *Simplified-SDC Method*

A second-order accurate time integration method using a simple deferred correction strategy was shown in (Nonaka et al. 2012). A version of this was also implemented in the Maestro low-Mach number hydrodynamics code some time ago. A compressible version, for use in Castro is slightly different due to the need to do some operations on the conserved variable state and some on the primitive variable state. We describe the application of this to the equations of compressible hydrodynamics below:

- *Initialization*
  - Set the first iterate of the primitive-variable reactive source terms from the last step:
$$\mathcal{I}_{\mathbf{q}}^{n+1/2,(0)} = \mathcal{I}_{\mathbf{q}}^{n-1/2,(s_\mathrm{max})} \tag{11}$$
- *Iterate*

  Iterate from $k = 1, \ldots, s_\mathrm{max}$. For second-order accuracy, $s_\mathrm{max} = 2$ is sufficient.
  - *Create the Advective Update Term*
    - convert $\boldsymbol{\mathcal{U}} \rightarrow \mathbf{q}$. This is an algebraic transformation, but will require the EOS.
    - predict $\mathbf{q}$ to the interfaces at $n + 1/2$ using the CTU PPM method. The source terms, $\mathbf{S_q}$, used in the prediction are:
$$\mathbf{S_q} = \mathbf{S_q^{\mathrm{hydro}}}^{n+1/2} + \mathcal{I}_{\mathbf{q}}^{n+1/2,(k-1)} \tag{12}$$

      Here we use the iteratively lagged integrals of the primitive variable terms accounting only for reactions, $\mathcal{I}_{\mathbf{q}}^{n+1/2,(k-1)}$, as the reactive source. Any hydrodynamic source terms are time-centered using the previous iteration:
$$\mathbf{S_q^{\mathrm{hydro}}}^{n+1/2} = \frac{1}{2}\left(\mathbf{S_q^{\mathrm{hydro}}}^{n} + \mathbf{S_q^{\mathrm{hydro}}}^{n+1,(k-1)}\right) \tag{13}$$

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

$$\left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]_{i,j,k}^{n+1/2,(k)} = -\frac{\mathbf{F}^{(x)}(\boldsymbol{\mathcal{U}}_{i+1/2}^{n+1/2}) - \mathbf{F}^{(x)}(\boldsymbol{\mathcal{U}}_{i-1/2}^{n+1/2})}{\Delta x} \tag{14}$$

Now the conservative hydrodynamics source terms are computed by first updating to the new state with advection only[1], as:

$$\boldsymbol{\mathcal{U}}^{\star\star} = \boldsymbol{\mathcal{U}}^n + \Delta t\left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]^{n+1/2,(k)} \tag{15}$$

The final advective update term is then:

$$\left[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})\right]_{i,j,k}^{n+1/2,(k)} = \left[\tilde{\boldsymbol{\mathcal{A}}}(\boldsymbol{\mathcal{U}})\right]_{i,j,k}^{n+1/2,(k)} + \mathbf{H}^{n+1/2} \tag{16}$$

with

$$\mathbf{H}^{n+1/2} = \begin{pmatrix} 0 \\ 0 \\ \mathbf{S}_{\rho\mathbf{U}}^{n+1/2} \cdot \mathbf{e}_x \\ S_{\rho E}^{n+1/2} \end{pmatrix} \tag{17}$$

if $\mathbf{H}$ depended on the outcome of the burn, we might need to do things differently here

- *Update the System Using a Method of Lines Integration*
  - Define the hydro source terms—these will be time-centered, but piecewise-constant in time throughout the integration.
  - With the spatial discretization done, our system appears as a system of ODEs:

$$\left.\frac{d\boldsymbol{\mathcal{U}}}{dt}\right|_{i,j,k} = \left[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})\right]_{i,j,k}^{n+1/2,(k)} + \mathbf{R}(\boldsymbol{\mathcal{U}}) \tag{18}$$

what is the right form of acceleration here? should we do piecewise linear in time?

As we are integrating this system we need to get the temperature, T, for the rate evaluations. We construct this by subtracting the kinetic energy from the total energy to get the specific internal energy, e, and then calling the equation of state.

Note that our advection terms are piecewise constant approximations, so when we integrate our system, their contributions will be linear in time. Since (ρX_k) sees a linear advective term, ρ does as well. $(\rho\mathbf{U})$ and (ρE) will also have linear contributions in time from advection. We can see that the internal energy will also have a linear contribution by recognizing

need to think about this more

[1] 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

that the kinetic energy term, $K = (\rho|\mathbf{U}|^2)/\rho$ is linear in time. Since the reactions don't affect ρ and $(\rho\mathbf{U})$, we can algebraically update these using this piecewise linear behavior:

$$\rho(t) = \rho^n + [\boldsymbol{\mathcal{A}}(\rho)]^{n+1/2,(k)}\, t \tag{19}$$

$$(\rho\mathbf{U})(t) = (\rho\mathbf{U})^n + [\boldsymbol{\mathcal{A}}((\rho\mathbf{U}))]^{n+1/2,(k)}\, t \tag{20}$$

At the end of the integration, our new state is $\boldsymbol{\mathcal{U}}^{n+1,(k)}$.

– *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:

fix signs on A

$$\boldsymbol{\mathcal{I}}_{\boldsymbol{\mathcal{U}}}^{(k)} = \frac{\boldsymbol{\mathcal{U}}^{n+1,(k)} - \boldsymbol{\mathcal{U}}^n}{\Delta t} - [\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})]^{n+1/2,(k)} \tag{21}$$

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

$$\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)} = \frac{\mathbf{q}^{n+1,(k)} - \mathbf{q}^n}{\Delta t} - [\boldsymbol{\mathcal{A}}(\mathbf{q})]^{n+1/2,(k)} \tag{22}$$

but we need the advective update for $\mathbf{q}$, which we have not constructed. 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:

$$\boldsymbol{\mathcal{U}}^\star = \boldsymbol{\mathcal{U}}^n + \Delta t[\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{U}})]^{n+1/2,(k)} \tag{23}$$

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:

$$[\boldsymbol{\mathcal{A}}(\mathbf{q})]^{n+1/2,(k)} = \frac{\mathbf{q}^\star - \mathbf{q}^n}{\Delta t} \tag{24}$$

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 Eq. 22. Putting all of this together, we see:

$$\boldsymbol{\mathcal{I}}_{\mathbf{q}}^{(k)} = \frac{\mathbf{q}^{n+1,(k)} - \mathbf{q}^\star}{\Delta t} \tag{25}$$

2.6. *Old Primitive Stuff*

and our primitive system as:

$$\mathbf{q}_t = \boldsymbol{\mathcal{A}}(\mathbf{q}) + \mathbf{R}(\mathbf{q}) \tag{26}$$

with

$$\boldsymbol{\mathcal{A}}(\mathbf{q}) = -\mathbf{A}^{(x)}(\mathbf{q})\mathbf{q}_x - \mathbf{A}^{(y)}(\mathbf{q})\mathbf{q}_y - \mathbf{A}^{(z)}(\mathbf{q})\mathbf{q}_z + \mathbf{S}_{\mathbf{q}}^{\text{hydro}} \tag{27}$$

3. SIMULATIONS

3.1. *Reaction Convergence*

3.2. *Flames*

ECSNe flames

aprox13 flames

simplified-SDC takes 3x more timesteps than Strang

look at number of right hand side calls

3.3. *Detonations*

Our focus here is modeling helium detonations. Previously, we showed Katz & Zingale (2019) that accurate modeling the ignition of a detonations requires a resolution such that $\tau_s < \tau_e$, where $\tau_s = \Delta x / c_s$ is the sound crossing time of a zone and $\tau_e = e/\dot{e}$ is the nuclear energy injection timescale. For the detonations explored here, we will work in that regime.

Helium detonations can be important in different models of Type Ia supernova. In the double detonation model for Type Ia supernovae, a helium detonation is used to trigger a carbon detonation in the underlying white dwarf. White dwarf collisions or mergers can have helium detonations if there is a thin layer of helium on the surface of the white dwarfs. We will model two densities, $2 \times 10^6\ \mathrm{g\ cm^{-3}}$, e.g., as used in **?** for double detonation models, and $10^7\ \mathrm{g\ cm^{-3}}$ to see the effects at higher densities.

Effect of number of SDC iterations

Effect of timestep

Effect of resolution

Difference densities?

Something about expense? We can just compare CPU time.

3.4. *Massive Star*

we need to look at the number of RHS calls spatially

also need to look at tolerances in the burner

analytic Jacobian run is much faster

what does Strang run do

3.5. *X-ray bursts*

crashed with too many subcycles. Need to rerun with new Jacobian

3.6. *Hydro and Gravity*

Merging stars

it doesn't run more than a few steps with simplified SDC

has a small density error and too many subcycles after 128 steps

4. SUMMARY

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

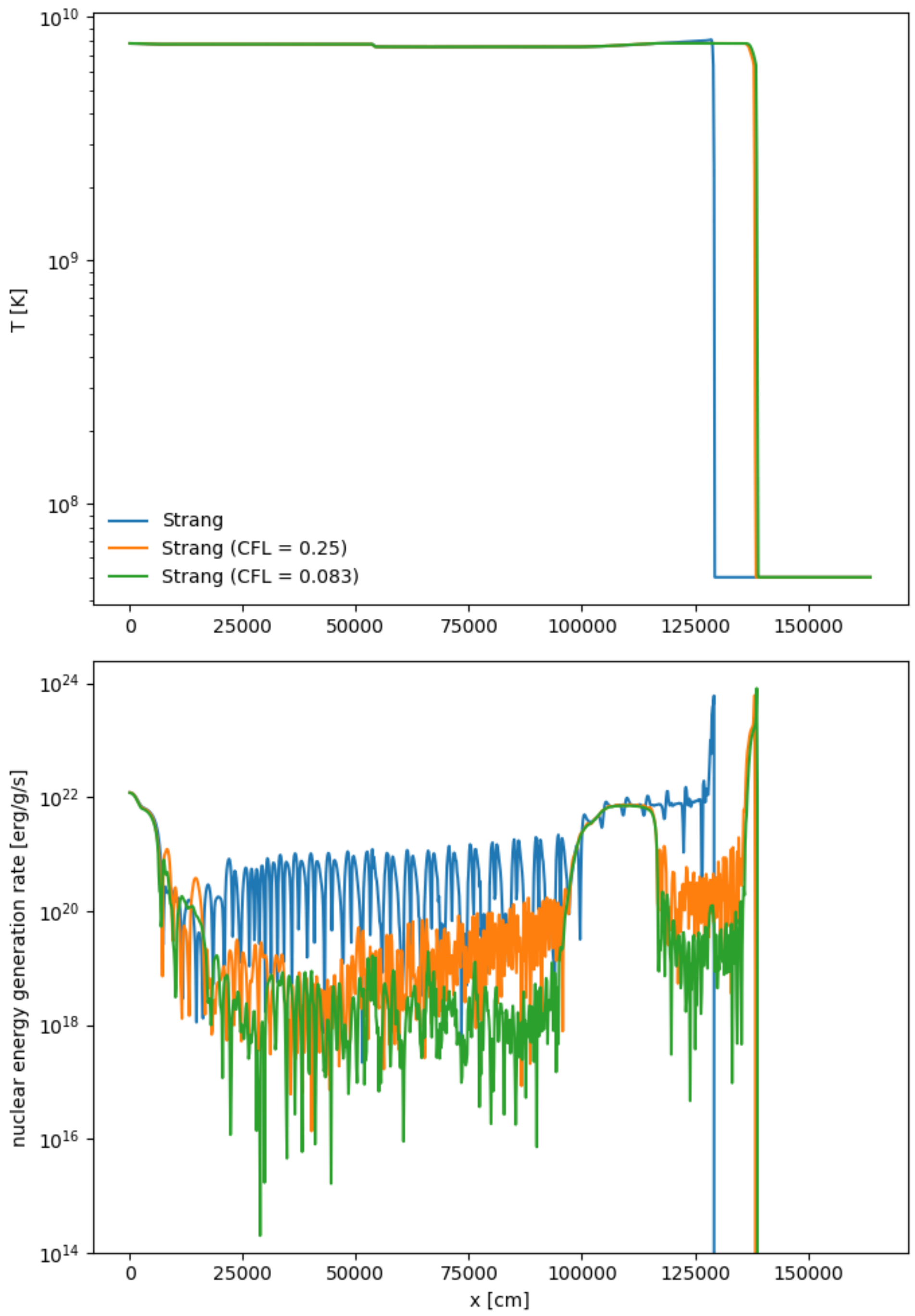

Figure 1. Effect of timestep size on the detonation using Strang splitting.

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.

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,

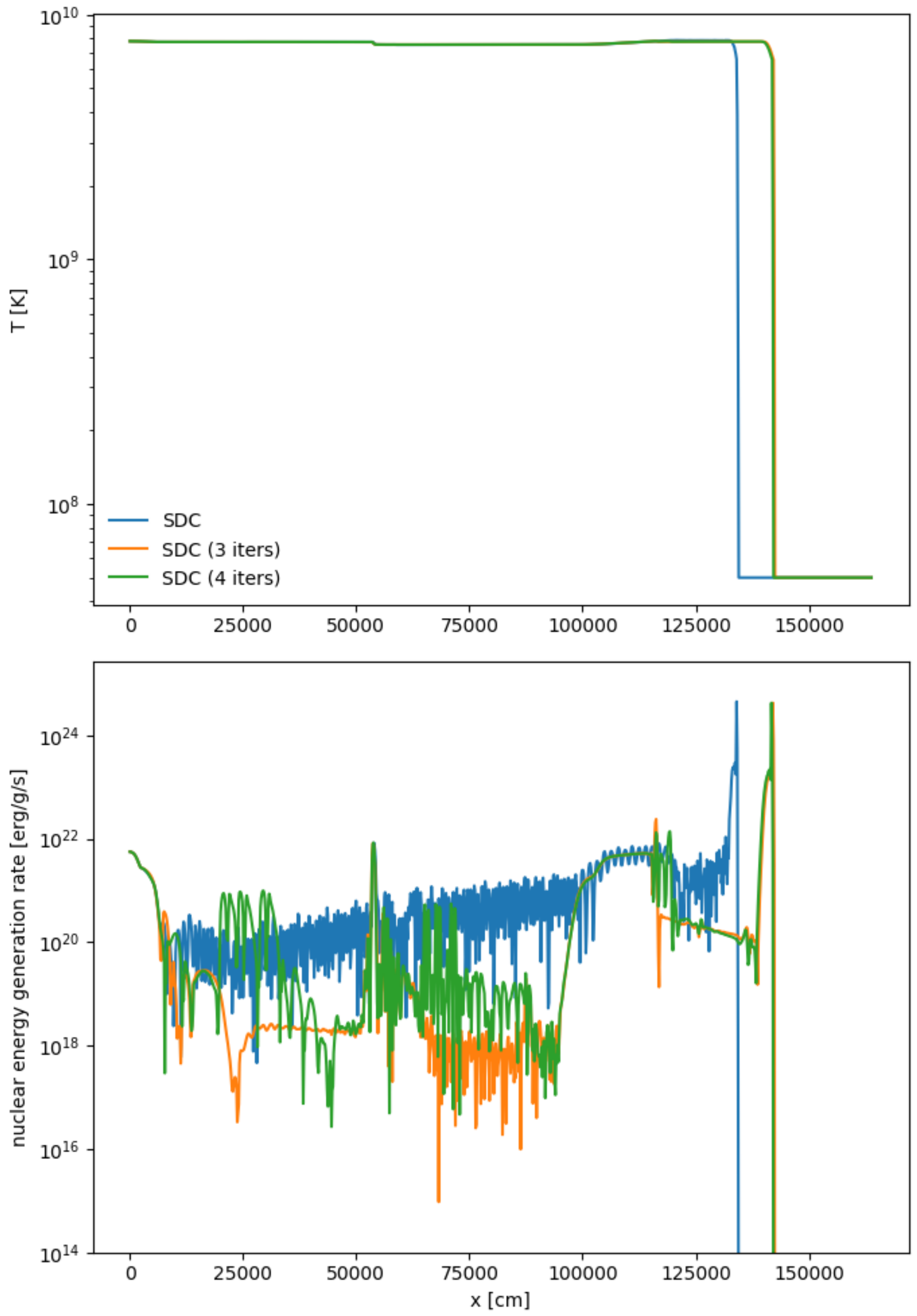

Figure 2. Effect of number of SDC iterations detonation.

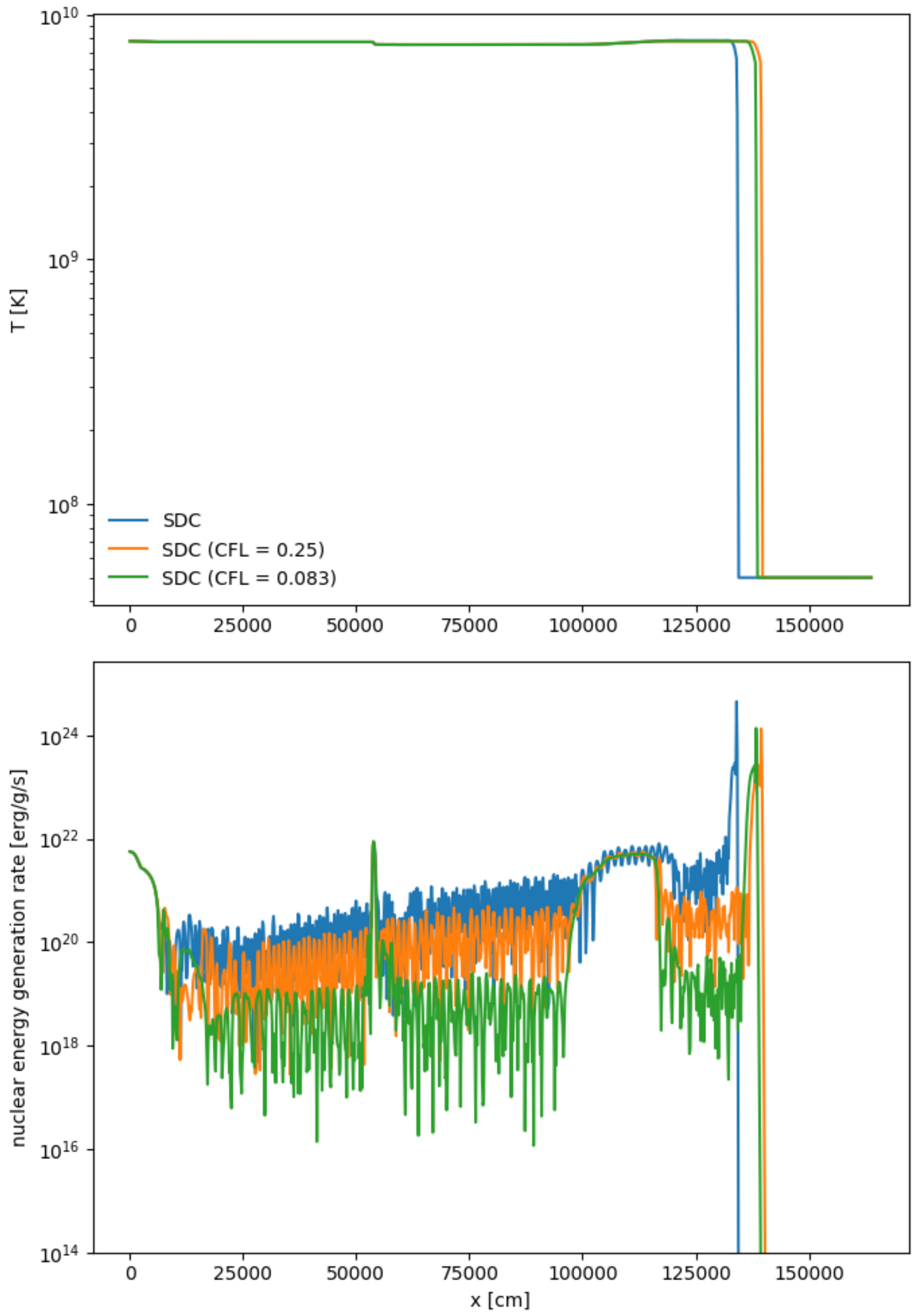

Figure 3. Effect of timestep size on the detonation using the SDC method.

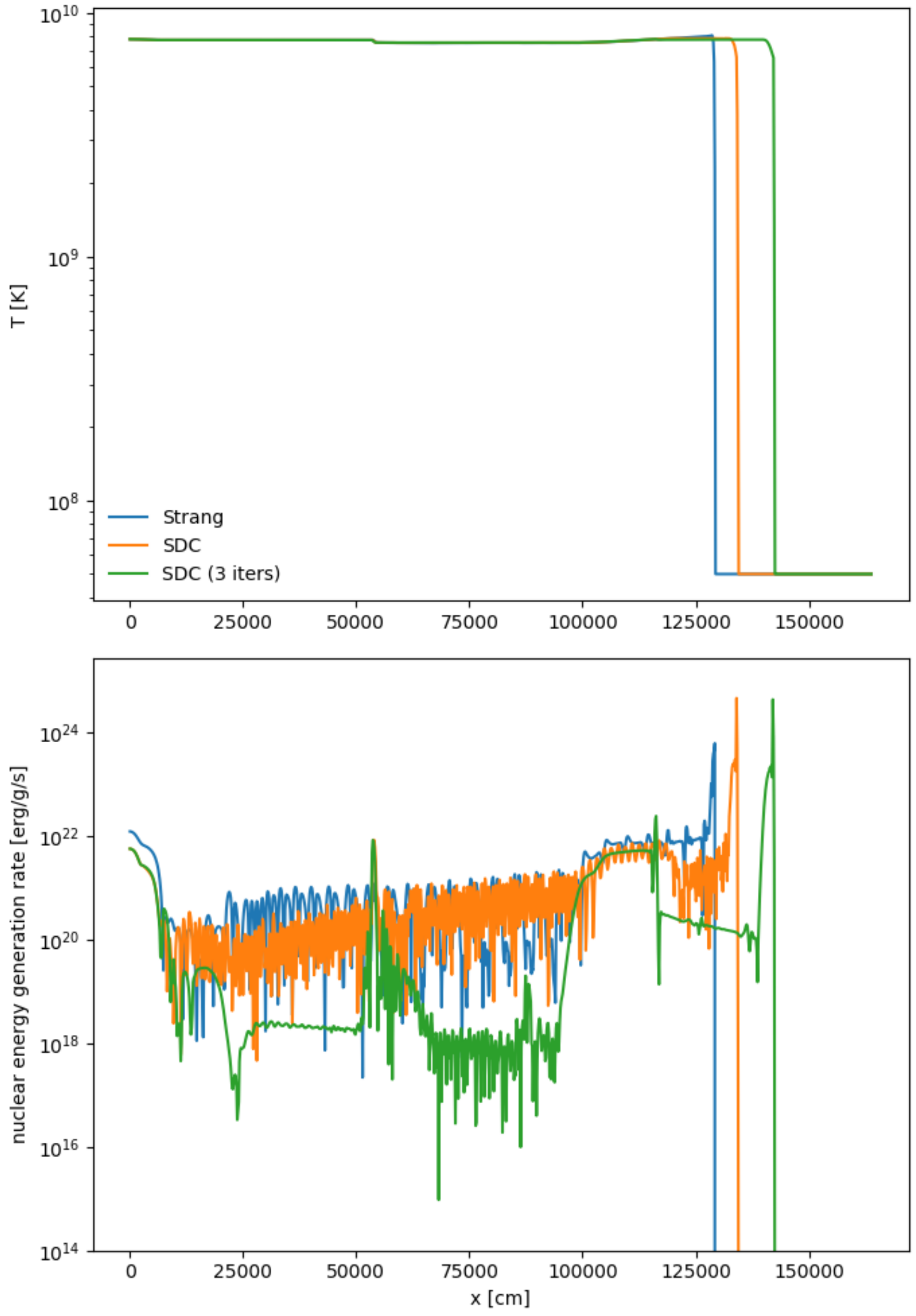

Figure 4. Comparison of Strang and SDC on the detonation problem.

http://matplotlib.org/) NumPy (Oliphant 2007; van der Walt et al. 2011), python (https://www.python.org/)

APPENDIX

A. JACOBIAN

To solve the reaction system implicitly, the ODE solver needs the Jacobian, $\partial\mathbf{R}/\partial\boldsymbol{\mathcal{U}}$. We follow the method of Zingale et al. (2019) and factor this into two pieces,

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

.

Even though it has no reactive sources, we include ρ in our conservative state for the purposes of computing the Jacobian (we denote the conserved state used for the Jacobian $\boldsymbol{\mathcal{U}}'$. Writing this out for two species, X_α and X_β, we have

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

We take the intermediate state to be $\mathbf{w} = (\rho, X_\alpha, X_\beta, K, T)^\intercal$, where K is the kinetic energy:

$$K = \frac{1}{2}|\mathbf{U}|^2 \tag{A3}$$

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

$$\frac{\partial\boldsymbol{\mathcal{U}}}{\partial\mathbf{w}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ X_\alpha & \rho & 0 & 0 & 0 \\ X_\beta & 0 & \rho & 0 & 0 \\ \rho e_\rho + e + K & \rho e_{X_\alpha} & \rho e_{X_\beta} & \rho & \rho e_T \\ \rho e_\rho + e & \rho e_{X_\alpha} & \rho e_{X_\beta} & 0 & \rho e_T \end{pmatrix} \tag{A4}$$

where we use the following notation for compactness:

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

and write the kinetic energy as $K = |\mathbf{U}|^2/2$. We get the inverse (computed via SymPy, see the included Jupyter notebook) is:

$$\frac{\partial \mathbf{w}}{\partial \boldsymbol{\mathcal{U}}'} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{X_\alpha}{\rho} & \frac{1}{\rho} & 0 & 0 & 0 \\ -\frac{X_\beta}{\rho} & 0 & \frac{1}{\rho} & 0 & 0 \\ -\frac{K}{\rho} & 0 & 0 & \frac{1}{\rho} & -\frac{1}{\rho} \\ \frac{\sum_k X_k e_{X_k} - \rho e_\rho - e}{\rho e_T} & -\frac{e_{X_\alpha}}{\rho e_T} & -\frac{e_{X_\beta}}{\rho e_T} & 0 & \frac{1}{\rho e_T} \end{pmatrix} \tag{A6}$$

The reaction vector is

$$\mathbf{R}(\boldsymbol{\mathcal{U}}') = \begin{pmatrix} 0 \\ \rho\dot{\omega}_\alpha \\ \rho\dot{\omega}_\beta \\ \rho\dot{S} \\ \rho\dot{S} \end{pmatrix} \tag{A7}$$

and the Jacobian is computed as $\partial\mathbf{R}/\partial\mathbf{w}$:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{w}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \dot{\omega}_\alpha + \rho\frac{\partial\dot{\omega}_\alpha}{\partial\rho} & \rho\frac{\partial\dot{\omega}_\alpha}{\partial X_\alpha} & \rho\frac{\partial\dot{\omega}_\alpha}{\partial X_\beta} & 0 & \rho\frac{\partial\dot{\omega}_\alpha}{\partial T} \\ \dot{\omega}_\beta + \rho\frac{\partial\dot{\omega}_\beta}{\partial\rho} & \rho\frac{\partial\dot{\omega}_\beta}{\partial X_\alpha} & \rho\frac{\partial\dot{\omega}_\beta}{\partial X_\beta} & 0 & \rho\frac{\partial\dot{\omega}_\beta}{\partial T} \\ \dot{S} + \rho\frac{\partial\dot{S}}{\partial\rho} & \rho\frac{\partial\dot{S}}{\partial X_\alpha} & \rho\frac{\partial\dot{S}}{\partial X_\beta} & 0 & \rho\frac{\partial\dot{S}}{\partial T} \\ \dot{S} + \rho\frac{\partial\dot{S}}{\partial\rho} & \rho\frac{\partial\dot{S}}{\partial X_\alpha} & \rho\frac{\partial\dot{S}}{\partial X_\beta} & 0 & \rho\frac{\partial\dot{S}}{\partial T} \end{pmatrix} \tag{A8}$$

REFERENCES

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

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

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

Katz, M. P., & Zingale, M. 2019, ApJ, 874, 169, doi: 10.3847/1538-4357/ab0c00

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

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

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