`DualFLD`: A FLD-based X-ray and UV Radiation Solver for ENZO

Daniel R. Reynolds

September 26, 2013

1 Introduction

This document describes a new, highly scalable, field-based dual X-ray and UV radiation solver for Enzo, `DualFLD`. In this solver, we transport up to two radiation fields, hereafter named E_{Xr} and E_{UV}. Both of these are assumed to be scalar-valued fields defined throughout the spatial domain, and are evolved using a flux-limited diffusion (FLD) approximation for the propagation, absorption and dilution of radiation in a cosmologically-expanding spatial domain. Each radiation field is assumed to have either a monochromatic SED at a specified input frequency, or is treated as an integrated radiation energy density with an assumed radiation spectrum. Separate SED choices may be made for both E_{Xr} and E_{UV}. Interaction between these radiation fields is assumed to occur only through interaction with the baryonic density and total energy fields defined in the volume, i.e. there is no direct X-ray to UV radiation coupling. The interactions with the baryonic densities and total energy fields occurs through four mechanisms:

(a) Absorption of radiation due to the monochromatic or frequency-integrated opacities valid over the frequency spectrum where each field is valid.

(b) Photo-ionization of chemistry fields due to the radiation present within each spatial cell.

(c) Photo-heating of the total energy field due to radiation present within each spatial cell.

(d) Secondary ionizations of chemistry fields due to X-ray radiation at frequencies exceeding 100 eV.

While the `DualFLD` solver handles the propagation of both radiation fields, it does not handle steps (b)-(d) directly, and instead computes photo-ionization and photo-heating rates that are passed to Enzo's built-in chemical ionization and gas heating solvers.

The defining characteristics of this solver in comparison with the `gFLDSplit` solver are three-fold. First, similarly to `gFLDSplit`, the `DualFLD` solver propagates radiation in a separate operator-split step from the chemistry and heating, though in this solver the radiation solve itself is split into separate UV and X-ray solves (since they are assumed to be non-interacting), and the photo-heating and photo-ionization rates are computed afterwards from the combined radiation contributions. Second, the `DualFLD` solver itself *cannot* perform ionization or heating directly, instead relying on Enzo's existing solvers for those physics. Third, the `DualFLD` module *only* allows a standard chemistry-dependent model (i.e. it does not allow the *local thermodynamic equilibrium* model that is supported by `gFLDSplit`).

This guide is designed to highlight the solvers and equations available in this module. Additional details on the model derivation, numerical methods and verification tests will be provided in subsequent publications. In order to best aid in usage of the `DualFLD` solver, we first provide details on how to use the solver within an Enzo simulation, and follow the description of those parameters with the supporting equations and algorithm description.

2 DualFLD usage

In order to use the DualFLD radiation solver module, and to allow optimal control over the solver methods used, there are a number of parameters that may be supplied to Enzo. We group these into two categories, those associated with the general startup of the module via the Enzo infrastructure, and those that may be supplied to the DualFLD module itself. However, prior to embarking on a description of these parameters, there are a few requirements for any problem that wishes to use the DualFLD module.

The Enzo source code is located in the subdirectory enzo/ of the code repository. Within this directory, the Makefile must be edited to specify how the code should be run, and where to look for machine-specific compilation instructions. The main parameter names within this Makefile are self-explanatory, but the following four parameters are used to determine the machine-specific file to use in compilation:

- ENZO_SYSTEM – this should hold a description of the machine itself (e.g. "jaguar")
- ENZO_COMPILER – this should hold a description of the compilers to use (e.g. "pgi")
- ENZO_MODE – this should hold either "mpi" or "hybrid", corresponding to the parallelization strategy (pure MPI or hybrid MPI/OpenMP, respectively)
- ENZO_SPEC – I'm not sure what this should be used for; in all cases I've seen it holds the word "test"

With these definitions, the Enzo Makefile will look for machine/compiler/parallelism specific definitions within a file

```
Make.${ENZO_SYSTEM}_${ENZO_COMPILER}.${ENZO_MODE}_${ENZO_SPEC}
```

located in the enzo/Macros/ subdirectory of the overall code repository. For example, with the definitions ENZO_SYSTEM=reynolds, ENZO_COMPILER=intel, ENZO_MODE=mpi, ENZO_SPEC=test, the Makefile will read the relevant compiler/library definitions from the file Make.reynolds_intel.mpi_test.

Within this machine-specific file, definitions must be included for the parameters:

- FC – MPI-enabled Fortran compiler
- DEFINES – Fortran pre-processor definitions
- FC_OPT – Fortran compiler optimization flags
- CC – MPI-enabled C compiler
- CC_DEF – C pre-processor definitions
- CC_OPT – C compiler optimization flags
- CCxx – MPI-enabled C++ compiler
- CCxx_DEF – C++ pre-processor definitions
- CCxx_OPT – C++ compiler optimization flags
- LDR – Program for linking the final executable
- LDR_OPT – Pre-processor definitions for the linker
- LIB – Library locations and files to link with (**must** include HYPRE, HDF5 and SZIP)

- `INCLUDE` – Include directories containing relevant header files (**must** include HYPRE and HDF5 header file locations)
- `OPTIONS` – unknown (currently empty)
- `OPT` – combined compilation options for generating `.o` files; currently contains `-c ${OPTIONS} ${INCLUDE}`
- `DEBUG` – unknown (currently empty)

The HYPRE library linked in must be version 2.8.0b or newer. If a user must compile HYPRE themselves to obtain this version, they should make note of the HYPRE configuration option `--with-no-global-partition`, which must be used for solver scalability when using over $\sim$1000 processors, but which results in slower executables on smaller-scale problems.

Final note: the SVN repository contains a file `enzo/mpi_save.h`. On most systems, this file is not needed to build Enzo, and may be ignored. However, on some systems Enzo may not compile with the native `mpi.h` file, in which case this file should be renamed to `enzo/mpi.h`. It is suggested that you first attempt to build Enzo with the file retaining the name `enzo/mpi_save.h`. If compilation fails due to MPI-related error messages, you may then rename the file the `enzo/mpi.h` and try compilation again.

2.1 Startup parameters

In a user's main problem parameter file, the following parameters must be set (their default values are in brackets):

- `RadiationHydrodynamics` [0] – this specifies to enable one of the FLD solver modules within Enzo's operator-split time-stepping approach. Allowable values include:
  - 0 – disable all FLD solvers
  - 1 – enable FLD solvers, use along-side other Enzo physics modules
  - 2 – enable FLD solvers and disable all other Enzo physics modules
- `ImplicitProblem` [0] – this specifies the type of FLD solver to use. The `DualFLD` solver corresponds to `ImplicitProblem = 4`.
- `ProblemType` [0] – as usual, this is problem-dependent. However, for FLD-based solvers in this version of Enzo, the value of ProblemType is typically within the 200's.
- `RadiationFieldType` [0] – this can be any value *except* 10 or 11, since those use pre-existing background radiation approximations.
- `RadiativeCooling` [0] – a nonzero value enables Enzo's built-in chemistry/heating solvers.
- `Multispecies` [0] – when `RadiativeCooling` is set to 1, a nonzero value for `Multispecies` will turn on Enzo's chemical ionization solvers. The particular value is used to enable various chemistry models, as described in the main Enzo documentation,
- `RadHydroParamfile` [NULL] – this should contain the filename (with path relative to this parameter file) that contains all module-specific solver parameters (discussed below). While the `DualFLD` module parameters may be supplied in the main parameter file, that filename must still be specified here (though it is not recommended, since the `ReadParameterFile.C` routine will complain about all of the 'unknown' parameters that are read elsewhere).
- `CoolDataParameterFile` [NULL] – this should contain the filename (with path relative to this parameter file) that contains all CoolData-specific solver parameters. Typical non-default values include

– `HydrogenFractionByMass` [0.76] – this gives the fraction of total mass comprised of Hydrogen; it is useful for performing Hydrogen-only simulations.

– `RateDataCaseBRecombination` [0] – a nonzero value denotes that Enzo should use case-B recombination rates instead of the default case-A rates.

In addition, if a user wishes to set up a new `ProblemType` that uses the `DualFLD` module, they must allocate standard Enzo baryon fields having the `FieldType` set to `RadiationFreq0` and `RadiationFreq1`. It is these baryon fields that will be evolved by the `DualFLD` module, with `RadiationFreq0` corresponding to the X-ray radiation field and `RadiationFreq1` corresponding to the UV radiation field. If radiative cooling and ionization are to be performed, additional baryon fields must be allocated with `FieldType` set to `kphHI` and `PhotoGamma`, as well as `kphHeI` and `kphHeII` if Helium is included, and `kdissH2I` if `Multispecies`¿1. All of these fields may be initialized to zero.

Furthermore, the `DualFLD` module currently computes its own emissivity fields $\eta_{Xray}(\mathbf{x})$ and $\eta_{UV}(\mathbf{x})$, in the routine `DualFLD_RadiationSource.src90`. A user may edit this file to add in an emissivity field corresponding to their own `ProblemType`. Alternatively, a user may separately fill in the `BaryonFields` `Emissivity0` and `Emissivity1` in other Enzo routine(s). Then, when Enzo is compiled using the pre-processor directive `EMISSIVITY` and run with the global parameter `StarMakerEmissivity` $\neq$ `0`, the `DualFLD` module will look to these `BaryonFields` for emissivity values instead of computing its own.

2.2 Module parameters

Once a user has enabled the `DualFLD` module, they have complete control over a variety of internal module parameters. The parameters are given here, with their default values specified in brackets, and references to the appropriate equations elsewhere in this document.

- `DualFLDXrayOnly` [0] – this parameter disables propagation of the UV radiation field in the solver, and eliminates all E_{UV}-related contributions to photo-heating and photo-ionization.
- `DualFLDXrayStatic` [0] – this parameter disables propagation of the X-ray radiation field in the solver, effectively freezing the E_{Xr} field to its initial configuration. This does not currently deplete the radiation density due to cosmological expansion, although such an enhancement would not be difficult to add to `DualFLD` in the future.
- `DualFLDUVStatic` [0] – this parameter disables propagation of the UV radiation field in the solver, effectively freezing the E_{UV} field to its initial configuration. This does not currently deplete the radiation density due to cosmological expansion, although such an enhancement would not be difficult to add to `DualFLD` in the future.
- `DualFLDXraySpectrum` [-1] – this parameter chooses the type of assumed X-ray radiation energy spectrum from equation (2). Allowed values include

  -1. monochromatic spectrum.

  0. power law spectrum,
  $$\chi(\nu) = \left(\frac{\nu}{\nu_{HI}}\right)^{-1.5}$$

  1. $T = 10^5$ K blackbody spectrum,
  $$\chi(\nu) = \frac{8\pi h \left(\frac{\nu}{c}\right)^3}{\exp\left(\frac{h\nu}{k_b 10^5}\right) - 1}.$$

- `DualFLDUVSpectrum` [-1] – this has the same options as above, but pertain to the UV radiation field (3).

- `DualFLDXrayFrequency` [500] – if the X-ray spectrum is monochromatic, this input specifies the frequency of the radiation field (in eV).
- `DualFLDUVFrequency` [13.6] – if the X-ray spectrum is monochromatic, this input specifies the frequency of the radiation field (in eV).
- `DualFLDChemistry` [3] – this parameter controls how many chemical species the solver should interact with (i.e. for opacity calculations and photo-ionization and photo-heating rates). Allowable values are
  0. no chemistry
  1. Hydrogen chemistry
  3. Hydrogen+Helium chemistry
- `DualFLDHFraction` [0.76] – this parameter controls the fraction of baryonic matter comprised of Hydrogen that the solver module should assume. Allowable values are $0 \leq$ `RadHydroHFraction` ≤ 1.
- `DualFLDMaxDt` [10^{20}] – this parameter sets the value of $\Delta t_{\max}$ from section 5.2; it must be greater than 0. This value is provided in *scaled* time units, i.e. $\Delta t_{\text{physical}} \leq \Delta t_{\max} *$ TimeUnits, where TimeUnits is Enzo's internal time scaling factor for the simulation.
- `DualFLDMinDt` [0] – this parameter sets the value of $\Delta t_{\min}$ from section 5.2; it must be non-negative. This value must also be given in scaled time units.
- `DualFLDInitDt` [10^{20}] – this parameter sets the initial time step size for the `DualFLD` module. We note that since the module will take the smaller of Δt_{FLD} and Δt_{Enzo}, the default value is never actually used. This value must also be given in scaled time units.
- `DualFLDDtNorm` [2] – this parameter sets the value of p from equation (18).
  - A value of 0 implies to use the max norm,
  - A value > 0 implies to use the corresponding p-norm,
  - Values < 0 are not allowed (reset to the default).
- `DualFLDDtGrowth` [1.1] – this gives the maximum time step size growth factor per solver step (i.e. allows an increase of 10% per step).
- `DualFLDDtXrayFac` [10^{20}] – this gives the value of τ_{tol} for the variables E_{Xr} from equation (20). They must be positive; the default essentially specifies no restrictions on Δt_{FLD}.
- `DualFLDDtUVFac` [10^{20}] – this gives the value of τ_{tol} for the variables E_{UV} from equation (20).
- `DualFLDXrayScaling` [1] – this gives the scaling factor s_{Xr} from (23); supplied values must be positive.
- `DualFLDUVScaling` [1] – this gives the scaling factor s_{UV} from (23); supplied values must be positive.
- `DualFLDTheta` [1] – this parameter specifies the value of θ in equation (15); requires $0 \leq \theta \leq 1$.
- `DualFLDXrBoundaryX0Faces`, `DualFLDXrBoundaryX1Faces`, `DualFLDXrBoundaryX2Faces` [0 0] – these specify the boundary-condition types from section 5.4 to use on the lower and upper boundaries in each direction for the X-ray radiation field. Allowable values are
  0. periodic (must match on both faces in a given direction)
  1. Dirichlet
  2. Neumann
- `DualFLDUVBoundaryX0Faces`, `DualFLDUVBoundaryX1Faces`, `DualFLDUVBoundaryX2Faces` [0 0] – these specify the boundary-condition types from section 5.4 to use on the lower and upper boundaries in each direction for the UV radiation field. Allowable values are

0. periodic (must match on both faces in a given direction)
1. Dirichlet
2. Neumann

- `DualFLDSolToleranceXray` [10^{-8}] – this parameter specifies the linear tolerance δ from section 5.1 for the X-ray solve. Allowable values must be between 10^{-15} and 1.
- `DualFLDSolToleranceUV` [10^{-8}] – this parameter specifies the linear tolerance δ from section 5.1 for the UV solve. Allowable values must be between 10^{-15} and 1.
- `DualFLDMaxMGItersXray` [50] – this positive parameter specifies the maximum number of multigrid iterations to perform for each radiation field in the MG-CG solver for the X-ray field from section 5.1.
- `DualFLDMaxMGItersUV` [50] – this positive parameter specifies the maximum number of multigrid iterations to perform for each radiation field in the MG-CG solver for the UV field from section 5.1.
- `DualFLDMaxPCGItersXray` [50] – this positive parameter specifies the maximum number of preconditioned conjugate gradient iterations to perform for each radiation field in the MG-CG solver for the X-ray field from section 5.1.
- `DualFLDMaxPCGItersUV` [50] – this positive parameter specifies the maximum number of preconditioned conjugate gradient iterations to perform for each radiation field in the MG-CG solver for the UV field from section 5.1.
- `DualFLDMGRelaxTypeXray` [1] – this parameter specifies the relaxation method used by the multigrid solver for the X-ray field:

  0. Jacobi
  1. Weighted Jacobi
  2. Red/Black Gauss-Seidel (symmetric)
  3. Red/Black Gauss-Seidel (nonsymmetric)

  For more information, see the HYPRE user manual.
- `DualFLDMGRelaxTypeUV` [1] – this parameter specifies the relaxation method used by the multigrid solver for the UV field. The values match those above.
- `DualFLDMGPreRelaxXray` [1] – this positive parameter specifies the number of pre-relaxation sweeps the multigrid solver should use in the MG-CG solver for the X-ray field from section 5.1.
- `DualFLDMGPreRelaxUV` [1] – this positive parameter specifies the number of pre-relaxation sweeps the multigrid solver should use in the MG-CG solver for the UV field from section 5.1.
- `DualFLDMGPostRelaxXray` [1] – this positive parameter specifies the number of post-relaxation sweeps the multigrid solver should use in the MG-CG solver for the X-ray field from section 5.1.
- `DualFLDMGPostRelaxUV` [1] – this positive parameter specifies the number of post-relaxation sweeps the multigrid solver should use in the MG-CG solver for the UV field from section 5.1.

3 Flux-limited diffusion radiation model

We begin with the equation for flux-limited diffusion radiative transfer in a cosmological medium [4],

$$\partial_t E_i + \frac{1}{a}\nabla \cdot (E_i \mathbf{v}_b) = \nabla \cdot (D_i \nabla E_i) - \frac{\dot{a}}{a} E_i - c\kappa_i E_i + \eta_i, \tag{1}$$

where here the comoving radiation energy density field E_i, emissivity η_i and opacity κ_i are functions of space and time, and where $i \in \{\mathrm{Xr}, \mathrm{UV}\}$. In this equation, the frequency-dependence of the respective radiation energy density field has been integrated away, under the premise of an assumed radiation energy spectrum,

$$\begin{aligned} & E_\nu(\nu, \mathbf{x}, t) = \tilde{E}_{Xr}(\mathbf{x}, t)\chi_{Xr}(\nu) + \tilde{E}_{UV}(\mathbf{x}, t)\chi_{UV}(\nu), \\ \Rightarrow & \\ & E_{Xr}(\mathbf{x}, t) = \int_0^\infty E_\nu(\nu, \mathbf{x}, t)\,\mathrm{d}\nu = \tilde{E}_{Xr}(\mathbf{x}, t)\int_0^\infty \chi_{Xr}(\nu)\,\mathrm{d}\nu, \qquad (2) \\ & E_{UV}(\mathbf{x}, t) = \int_0^\infty E_\nu(\nu, \mathbf{x}, t)\,\mathrm{d}\nu = \tilde{E}(\mathbf{x}, t)\int_0^\infty \chi_{UV}(\nu)\,\mathrm{d}\nu, \qquad (3) \end{aligned}$$

where $\tilde{E}_{Xr}$ and $\tilde{E}_{UV}$ are intermediate quantities (for analysis) that are never computed, and where we have assumed that the two spectra $\chi_{Xr}(\nu)$ and $\chi_{UV}(\nu)$ do not overlap (i.e. $\chi_{Xr}(\nu)$ disappears in the interval $[0, \nu_1)$ and $\chi_{UV}(\nu)$ disappears in the interval $[\nu_1, \infty)$). We note that if either assumed spectrum is the Dirac delta function, $\chi_i(\nu) = \delta_{\nu_i}(\nu)$, then E_i is a monochromatic radiation energy density at the ionization threshold $h\nu_i$, and the $-\frac{\dot{a}}{a}E$ term in equation (1), obtained through integration by parts of the redshift term $\frac{\dot{a}}{a}\partial_\nu E_\nu$, is omitted from (1). Similarly, the emissivity functions $\eta_i(\mathbf{x}, t)$ relate to the true emissivity $\eta_\nu(\nu, \mathbf{x}, t)$ by the formulas

$$\eta_{Xr}(\mathbf{x}, t) = \int_0^\infty \eta_\nu(\nu, \mathbf{x}, t)\,\mathrm{d}\nu \eta_{UV}(\mathbf{x}, t) = \int_0^\infty \eta_\nu(\nu, \mathbf{x}, t)\,\mathrm{d}\nu.$$

Within equation (1), the function D_i is the *flux limiter* that depends on face-centered values of E_i, ∇E_i and the opacity κ_i [2],

$$D_i = \min\left\{c\left(9\kappa_{i,f}^2 + R^2\right)^{-1/2}, D_{max}\right\}, \quad \text{and} \quad R = \max\left\{\frac{|\partial_x E_i|}{E_{i,f}}, R_{min}\right\}. \qquad (4)$$

Here the spatial derivative within R is computed using non-dimensional units at the computational face adjoining two neighboring finite-volume cells, $D_{max} = 0.006\,c\,L_{unit}$ and $R_{min} = 10^{-20}/L_{unit}$ with L_{unit} the length non-dimensionalization factor for the simulation, and the face-centered radiation energy density and opacity are computed using the arithmetic and harmonic means, respectively,

$$E_{i,f} = \frac{E_{i,1} + E_{i,2}}{2}, \qquad \kappa_{i,f} = \frac{2\kappa_{i,1}\kappa_{i,2}}{\kappa_{i,1} + \kappa_{i,2}},$$

where here $E_{i,1}$ and $E_{i,2}$ are the two values of E_i in the cells adjacent to the face. Among the many available limiter formulations we have tested [1, 2, 4], this version performs best at producing causal radiation propgation speeds in the low-opacity limit typical of reionization simulations.

4 Model couplings

In general, radiation calculations in Enzo are used in simulations where chemical ionization states are important. For these situations, we couple the radiation equation (1) with equations for both the conservation of gas energy and primordial chemistry ionization/recombination,

$$\partial_t e + \frac{1}{a}\mathbf{v}_b \cdot \nabla e = -\frac{2\dot{a}}{a}e - \frac{1}{a\rho_b}\nabla \cdot (p\mathbf{v}_b) - \frac{1}{a}\mathbf{v}_b \cdot \nabla\phi + G - \Lambda + \dot{e}_{SF}, \qquad (5)$$

$$\partial_t \mathtt{n}_j + \frac{1}{a}\nabla \cdot (\mathtt{n}_j \mathbf{v}_b) = \alpha_{j,k}\mathtt{n}_e\mathtt{n}_k - \mathtt{n}_j\Gamma_j^{ph}, \qquad j \in \{\mathrm{HI}, \mathrm{HII}, \mathrm{HeI}, \mathrm{HeII}, \mathrm{HeIII}\}. \qquad (6)$$

Here, $\mathtt{n}_j$ is the comoving number density for each chemical species, $\mathtt{n}_k$ corresponds to chemical species that interact with species $\mathtt{n}_j$, and $\mathtt{n}_e$ is the electron number density. In these equations, all terms are evolved by

Enzo's built-in chemistry and gas energy solvers, though some of the relevant rates result from radiation-dependent couplings. Specifically, the gas can be photo-heated by the radiation through the term

$$G = \frac{c\,E_{UV}\,\sum_j \mathtt{n}_j \int_{\nu_j}^{\infty} \sigma_j\,\chi_{UV}\left(1-\frac{\nu_j}{\nu}\right)\,d\nu}{\rho_b\,\int_0^{\infty}\chi_{UV}d\nu} + \frac{Y_\Gamma\,c\,E_{Xr}\,\sum_j \mathtt{n}_j \int_{\nu_j}^{\infty} \sigma_j\,\chi_{Xr}\left(1-\frac{\nu_j}{\nu}\right)\,d\nu}{\rho_b\,\int_0^{\infty}\chi_{Xr}d\nu}, \tag{7}$$

for $j \in \{\text{HI,HeI,HeII}\}$, where the X-ray secondary photo-heating coefficient Y_Γ depends on the electron fraction ξ in a cell via the formula

$$Y_\Gamma = 0.9971\left[1-\left(1-\xi^{0.2663}\right)^{1.3163}\right]. \tag{8}$$

Within the Enzo code base, G is stored in the baryon field `PhotoGamma`, for communication between `DualFLD` and Enzo's heating/cooling solvers.

Additionally, the photo-ionization rates Γ_j^{ph} within equation (6) depend on the X-ray and UV radiation fields via the formulas

$$\Gamma_j^{ph} = \frac{Y_j\,c\,E_{Xr}\,\int_{\nu_j}^{\infty}\frac{\sigma_j(\nu)\chi_{Xr}(\nu)}{\nu}\,\mathrm{d}\nu}{h\,\int_0^{\infty}\chi_{Xr}(\nu)\,\mathrm{d}\nu} + \frac{c\,E_{UV}\,\int_{\nu_j}^{\infty}\frac{\sigma_j(\nu)\chi_{UV}(\nu)}{\nu}\,\mathrm{d}\nu}{h\,\int_0^{\infty}\chi_{UV}(\nu)\,\mathrm{d}\nu}. \tag{9}$$

In this formula, we employ the X-ray photo-ionization coefficients

$$Y_{HI} = 0.3908\left(1-\xi^{0.4092}\right)^{1.7592}, \tag{10}$$

$$Y_{HeI} = 0.0554\left(1-\xi^{0.4614}\right)^{1.666}, \tag{11}$$

$$Y_{HeII} = 0. \tag{12}$$

Within the Enzo code base, the rates Γ_{HI}^{ph}, Γ_{HeI}^{ph} and Γ_{HeII}^{ph} are held in the baryon fields `kphHI`, `kphHeI` and `kphHeII` for communication between `DualFLD` and Enzo's chemistry solvers.

Lastly, the frequency-integrated opacities depent on the chemical state at each spatial location,

$$\kappa_{Xr} = \frac{\sum_j \mathtt{n}_j \int_{\nu_j}^{\infty}\chi_{Xr}\,\sigma_j\,d\nu}{\int_0^{\infty}\chi_{Xr}\,d\nu}, \quad j \in \{\text{HI,HeI,HeII}\} \tag{13}$$

$$\kappa_{UV} = \frac{\sum_j \mathtt{n}_j \int_{\nu_j}^{\infty}\chi_{UV}\,\sigma_j\,d\nu}{\int_0^{\infty}\chi_{UV}\,d\nu}, \quad j \in \{\text{HI,HeI,HeII}\}, \tag{14}$$

where these integrals with the assumed radiation spectra $\chi_{Xr}(\nu)$ and $\chi_{UV}(\nu)$ handle the change from the original frequency-dependent radiation equation to the integrated grey radiation equations.

Within the `DualFLD` module, the baryon field `kdissH2I` is always set to 0.

5 Numerical solution approach

We evolve these models in an operator-split fashion, wherein we solve the radiation equations (1) separately from the gas energy correction and chemistry equations (5) and (6), which are evolved together. These solves are coupled to Enzo's existing operator-split solver framework in the following manner:

(i) Project the dark matter particles onto the finite-volume mesh to generate a dark-matter density field;

(ii) Solve for the gravitational potential and compute the gravitational acceleration field;

(iii) Evolve the hydrodynamics equations using an up-to-second-order accurate explicit method, and have the velocity $\mathbf{v}_b$ advect both the X-ray and UV radiation fields, E_{Xr} and E_{UV};

(iv) Evolve the coupled gas energy correction and chemistry evolution equations;

(v) Advect the dark matter particles with the Particle-Mesh method;

(vi) Evolve the radiation fields implicitly in time using an up-to-second-order accurate method.

The implicit solution approach for step (vi) is similar to the one from [3]; here we describe only enough to point out the available user parameters, and more fully describe some additional options available in the solver.

In solving the steps (vi) we use a method of lines approach for the space-time discretization of (1), in that we first discretize the equations in space using a second-order-accurate, uniform-grid, finite volume discretization, and then evolve the resulting system of ODEs in time.

5.1 Radiation subsystem

Assuming that all spatial derivatives are treated using standard second-order centered difference approximations on our finite-volume grid, we need only discuss the time-discretization of our radiation equation (1). Our approach follows a standard two-level θ-method,

$$\begin{aligned} E_i^n - E_i^{n-1} - \theta\Delta t\left(\nabla\cdot\left(D_i^{n-1}\nabla E_i^n\right) - \frac{\dot{a}}{a}E_i^n - c\kappa_i^n E_i^n + \eta_i^n\right) \\ - (1-\theta)\Delta t\left(\nabla\cdot\left(D_i^{n-1}\nabla E_i^{n-1}\right) - \frac{\dot{a}}{a}E_i^{n-1} - c\kappa_i^{n-1}E_i^{n-1} + \eta_i^{n-1}\right) = 0, \end{aligned} \tag{15}$$

where the parameter $0 \le \theta \le 1$ defines the time-discretization, and where we have assumed that the advective portions of (1) have already been taken care of through Enzo's hydrodynamics solvers. Recommended values of θ are 1 (backwards Euler) and $\frac{1}{2}$ (trapezoidal, a.k.a. Crank-Nicolson).

Whichever θ value we use (as long as it is nonzero), the equation (15) is linearly-implicit in the time-evolved radiation energy density E_i^n. We write this in predictor-corrector form (for ease of boundary condition implementation), which we will write as

$$Js = b, \qquad E_i^n = E_i^{n-1} + s. \tag{16}$$

We approximately solve this linear equation for the update s, to a tolerance δ,

$$\|Js - b\|_2 \le \delta, \tag{17}$$

using using a multigrid-preconditioned conjugate gradient iteration.

Both the X-ray and UV fields are solved using the same time-discretization parameter θ, and the same multigrid solver parameters `DualFLDSolTolerance`, `DualFLDMaxMGIters`, `DualFLDMGRelaxType`, `DualFLDMGPreRelax` and `DualFLDMGPostRelax`; however, it would be trivial to extend the current `DualFLD` solver to allow specification of different values for each radiation field.

5.2 Time-step selection

Time steps are chosen adaptively in an attempt to control error in the calculated solution. To this end, we first define an heuristic measure of the time accuracy error in a radiation field E_i as

$$err = \left(\frac{1}{N}\sum_{j=1}^{N}\left(\frac{E_{i,j}^n - E_{i,j}^{n-1}}{\omega_j}\right)^p\right)^{1/p}, \tag{18}$$

where the weighting vector ω is given by

$$\omega_j = \sqrt{E_{i,j}^n E_{i,j}^{n-1}} + 10^{-3}, \quad j = 1, \ldots, N. \tag{19}$$

i.e. we scale the radiation change by the geometric mean of the old and new states, adding on a floor value of 10^{-3} in case any of the states are too close to zero. This approach works well when the internal solution variables are unit-normalized, or at least close to unit-normalized, since the difference between the old and new solutions, divided by this weighting factor ω, should give a reasonable estimate of the number of significant digits that are correct in the solution.

With these error estimates (18) for both E_{Xr} and E_{UV}, we set the new time step size for each subsystem based on the previous time step size and a user-input tolerance $\tau_{\rm tol}$ as

$$\Delta t^n = \frac{\tau_{\rm tol} \Delta t^{n-1}}{err}. \tag{20}$$

Since E_{Xr} and E_{UV} are evolved separately, we allow either solver to subcycle at a faster rate if necessary to allow convergence of the underlying linear solver. However, in general we enforce that both fields utilize the same step size,

$$\Delta t^n = \min\{\Delta t_{Xr}^n, \Delta t_{UV}^n, \Delta t_{Enzo}^n\}, \tag{21}$$

where $\Delta t_{\rm Enzo}$ is the time step size that Enzo's other routines (e.g. hydrodynamics) would normally take. We further note that the `DualFLD` solver module will force Enzo to similarly take this more conservative time step size, due to the tight physical coupling between radiation transport and chemical ionization.

Additionally, a user may override these adaptive time step controls with the input parameters $\Delta t_{\rm max}$ and $\Delta t_{\rm min}$. However, even with such controls in place the overall time step will still be selected to adhere to the bound required by Enzo's other physical modules, i.e.

$$\Delta t^n = \min\{\Delta t_{\rm min}^n, \Delta t_{Enzo}^n\}. \tag{22}$$

5.3 Variable rescaling

In case Enzo's standard unit non-dimensionalization using `DensityUnits`, `LengthUnits` and `TimeUnits` is insufficient to render the resulting solver values E_{Xr} and E_{UV} to have nearly unit magnitude, the user may input additional variable scaling factors to be used inside the `DualFLD` module. The basic variable non-dimensionalization of these fields is to create a non-dimensional radiation field value by dividing the physical value (in ergs/cm^3) by the factor `DensityUnits * LengthUnits`^{2} `* TimeUnits`$^{-2}$. As would be expected, the values of E_{Xr} and E_{UV} may differ by orders of magnitude, so it is natural that they should be non-dimensionalized differently.

To this end, if we denote these user-input values as s_{Xr}, and s_{UV}, then the `DualFLD` module defines the rescaled variables

$$\tilde{E}_{Xr} = E_{Xr}/s_{Xr}, \qquad \tilde{E}_{UV} = E_{UV}/s_{UV}, \tag{23}$$

and then uses the rescaled variables $\tilde{E}_{Xr}$ and $\tilde{E}_{UV}$ in its internal routines instead of Enzo's "non-dimensionalized" internal variables E_{Xr} and E_{UV}. If the user does not know appropriate values for these scaling factors *a-priori*, a generally-applicable rule of thumb is to first run their simulation for a small number of time steps and investigate Enzo's HDF5 output files to see the magnitude of the values stored internally by Enzo; if these are far from unit-magnitude, appropriate scaling factors s_{Xr} and s_{UV} should be supplied in the `DualFLD` parameter input file.

5.4 Boundary conditions

As the radiation equation (1) is parabolic, boundary conditions must be supplied on the radiation field E_i. The `DualFLD` module allows three types of boundary conditions to be placed on the radiation field:

0. Periodic,
1. Dirichlet, i.e. $E_i(x,t) = g(x),\ x \in \partial\Omega$, and
2. Neumann, i.e. $\nabla E_i(x,t) \cdot n = g(x),\ x \in \partial\Omega$.

In most cases, the boundary condition types (and values of g) are problem-dependent. When adding new problem types, these conditions should be set near the bottom of the file `DualFLD_Initialize.C`, otherwise these will default to either (a) periodic, or (b) will use $g = 0$, depending on the user input boundary condition type.

6 Concluding remarks

We wish to remark that the module is not large (one header file, 14 C++ files, 2 F90 files), and all files begin with the `DualFLD` prefix. While we have strived to ensure that the module is bug-free, there is still work to be done in enabling additional physics, including more advanced time-stepping interactions with the rest of Enzo (especially when ionization sources "turn on" abruptly), and adaptive or static mesh refinement.

Feedback/suggestions to are welcome.

References

[1] J. C. Hayes and M. L. Norman, *Beyond Flux-limited Diffusion: Parallel Algorithms for Multidimensional Radiation Hydrodynamics*, Ap. J. Supp., 147 (2003), pp. 197–220.

[2] J. E. Morel, *Diffusion-limit asymptotics of the transport equation, the $p_{1/3}$ euqations, and two flux-limited diffusion theories*, J. Quant. Spectrosc. Radiat. Transfer, 65 (2000), pp. 769–778.

[3] M. L. Norman, D. R. Reynolds, G. C. So, and R. P. Harkness, *Direct numerical simulation of reionization in large cosmological volumes I: numerical methods and tests*, Ap. J., (submitted).

[4] D. R. Reynolds, J. C. Hayes, P. Paschos, and M. L. Norman, *Self-consistent solution of cosmological radiation-hydrodynamics and chemical ionization*, J. Comput. Phys., 228 (2009), pp. 6833–6854.