Computer Physics Communications 207 (2016) 415–425

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

FESTR: Finite-Element Spectral Transfer of Radiation spectroscopic modeling and analysis code☆

Peter Hakel

Computational Physics Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

ARTICLE INFO

Article history:
Received 29 January 2016
Received in revised form
19 May 2016
Accepted 26 May 2016
Available online 16 June 2016

Keywords:
Radiation transport
Raytracing

ABSTRACT

We report on the development of a new spectral postprocessor of hydrodynamic simulations of hot, dense plasmas. Based on given time histories of one-, two-, and three-dimensional spatial distributions of materials, and their local temperature and density conditions, spectroscopically-resolved signals are computed. The effects of radiation emission and absorption by the plasma on the emergent spectra are simultaneously taken into account. This program can also be used independently of hydrodynamic calculations to analyze available experimental data with the goal of inferring plasma conditions.

Program summary

Program title: FESTR

Catalogue identifier: AFAR_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AFAR_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: BSD 3-Clause license

No. of lines in distributed program, including test data, etc.: 1237516

No. of bytes in distributed program, including test data, etc.: 17542918

Distribution format: tar.gz

Programming language: C++.

Computer: HPC, PC.

Operating system: Linux, MacOS.

RAM: Problem dependent (based on size of input)

Classification: 1.3, 2.2, 20.2.

Nature of problem:
Calculation of spectral signals by postprocessing hydrodynamics simulations. Analysis of experimental spectroscopic data to infer plasma temperature and density conditions, and its chemical composition. Simultaneous treatment of spatial non-uniformity (on 3D unstructured meshes) along with spectroscopic-quality radiation transport.

Solution method:
Rays are cast across a 3D unstructured mesh that characterizes local temperature, density, and chemical composition of the material. Analytic solution to the 1D (along the ray) steady-state, local radiation transport equation is repeatedly used to gradually build a synthetic spectrum for each ray.

Restrictions:
Steady-state approximation of the radiation transport equation is used. Scattering as a radiation source is not included. Given plasma conditions are considered fixed; potential feedback of computed radiation back into plasma temperature and equation-of-state is neglected. Doppler shifts are not modeled at this

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

E-mail address: hakel@lanl.gov.

http://dx.doi.org/10.1016/j.cpc.2016.05.027

time. The quality of the computed results critically depends on the accuracy of external atomic databases used as input by FESTR. `Polygon` objects are assumed to be convex at present.

Running time:
Problem dependent (based on size of input)—the suite of 1190 enclosed unit tests runs in less than a minute on most tested platforms.

1. Introduction

For many decades, spectroscopic modeling and analysis has been a valuable tool applicable across a wide range of fields including astrophysics [1–3], plasma physics [4–15], and inertial confinement fusion (ICF) research [16–24]. A collection of photons recorded by a diagnostic instrument carries information about the properties of the material that emitted them, or through which this electromagnetic radiation has passed. More specifically, material conditions such as temperature, density, and chemical composition are imprinted on the emergent spectra. In hot, dense plasmas, for instance, the intensity of spectral lines characteristic of a given chemical element is a signature of that element's presence and abundance; intensity ratios of suitable pairs of spectral lines can be used to infer temperature; spectral line shapes and widths are sensitive to plasma density.

In general, spectroscopic modeling capabilities rest on a chain of theoretical and computational tools starting with fundamental atomic structure calculations (wave functions of electronic bound states in atoms) and probabilities of various atomic processes (e.g., decay rates associated with spectral line emission, cross sections for collisional processes such as excitation and ionization, etc.). In the next step, this type of data is fed into models that calculate optical properties of the material (spectrally resolved emissivity, absorption, and scattering) for given temperature and density conditions; in local thermal equilibrium (LTE) such a model can be based on thermodynamic arguments—in non-local thermal equilibrium (NLTE) conditions these models assume the form of a system of collisional-radiative rate equations [25]. The atomic physics and opacity team at Los Alamos National Laboratory (LANL) has developed and maintained such a suite of computer codes for many years [26]; the CATS (semi-relativistic [27,28]) and RATS (fully relativistic [29]) codes for fundamental atomic structure and radiative decay rates, the ACE code for collisional excitation cross sections [30], the GIPPER code for modeling the various ionization processes [31], and the ATOMIC code to perform both LTE and NLTE calculations ultimately yielding the desired optical properties of hot plasmas [26,32–34]. Over the years, this suite has been successfully used in a wide range of modeling and diagnostic applications (see [25,35], and references therein).

In this LANL suite, the ATOMIC code has been the final step in many modeling efforts, as it calculates the emissivity, absorption, and scattering data for a plasma of a given electron temperature T_e, radiation temperature T_r, and electron number density n_e (or, alternatively, mass density). As long as the plasma is sufficiently close to being uniform (i.e., it has no significant spatial non-uniformity), the results of a 0D code such as ATOMIC can be directly compared with experimentally recorded spectra—with the additional assumption that the emergent radiation is dominated either by self-emission (i.e., negligible absorption and scattering), or by opacity (in which case the self-emission is weak in relation to an external backlighter). In many cases, however, these assumptions are not valid—first, there are spatial non-uniformities in the material through which different temperatures and densities all make contributions to the emergent spectral signal, and, second, neither the emissivity nor the opacity dominates the formation of the spectrum, as both make important contributions. While these issues can be addressed by some existing software products (see, for example, SPECT3D [36]), we have decided to extend the LANL suite of codes with an additional modeling tool that expands the range of validity of our "in-house" spectral modeling capabilities even closer to comparisons with experimental data by removing the assumptions of plasma uniformity and the dominance of either emissivity or the opacity in spectral formation. The resulting new code named FESTR (Finite-Element Spectral Transfer of Radiation) is the subject of this paper. Spatial non-uniformity of the domain is handled with raytracing across a 3D unstructured mesh, in conjunction with piecewise analytic solution to the radiation-transport equation to model the interplay between the emissivity and the opacity of each zone (a locally uniform finite element) containing the material. It is a 3D generalization of previous 1D efforts, in which the same radiation-transport technique was used [10,23,24,37–40]. FESTR was written in C++; in the source code, as well as in this paper, the names of its classes are marked with upper-case initial letters.

This paper is outlined as follows. Since the main purpose of the FESTR code is to calculate the spectra of electromagnetic radiation collected at the location of a diagnostic instrument (see Fig. 1), in Section 2.1 we begin by describing the way the various diagnostics (i.e., detectors) are modeled in this code. In Section 2.2 we continue by detailing the geometry of tracing of rays that impinge on the diagnostics. Then, in Section 2.3, we show how to obtain mixed optical properties for each uniform spatial zone based on local plasma conditions, and how we build the synthetic spectrum by following the already calculated ray paths across the mesh. In Sections 2.4 and 2.5 we then discuss the two alternative modes of running FESTR: the postprocessing of hydrodynamic simulations, and the extraction of plasma conditions by the analysis of available experimental data. In Section 2.6 we summarize some important relationships between the building blocks of the code. Finally, we briefly describe some of the techniques used in the development and testing of FESTR and we provide a simple example that illustrates the main functionalities of the code.

2. Code description

2.1. Diagnostics

Class `Diagnostics` is a collection of `Detector` objects representing tally surfaces on which spectra are recorded. Each `Detector` is rectangular in shape and is subdivided into rectangular regions (called "patches" in the code) whose size is dictated by the spatial resolution of the actual instrument. This design is mainly motivated by the goal of modeling the multi-monochromatic imager (MMI) diagnostic instruments [20,41–50], which provide spatial, as well as spectral resolution. At every time instant, each patch collects spectra from potentially many `Rays` arriving from various directions (called "`Ray` bundles" in the code).

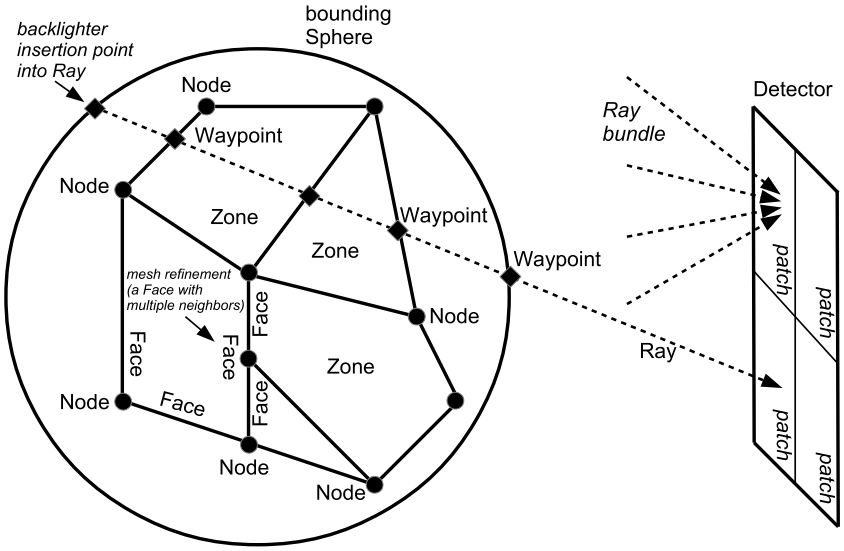

Fig. 1. Schematic illustration of the three-dimensional geometric data structures of FESTR.

In cases when the plasma size is small relative to its distance to the `Detector`, such parallax effects are negligible and the parallel `Ray` approximation (with each bundle containing just one `Ray`, parallel to all others, which then defines its patch) can also be used. Such space- and direction-resolved spectra may be further aggregated into space- and time-integrated spectra. Conversely, by integrating over a photon-energy range while retaining the spatial resolution of the data, narrow and broadband synthetic images can also be extracted. A special path in the code is used to speed up the calculation for cases with spherical symmetry. We note that this "patch-based" approach is not limited to the MMI as it also could be applied to other diagnostics such as image plates, CCD, and film.

2.2. Raytracing

The bounding `Sphere` separates the plasma confined within its volume from all the `Detectors` of the `Diagnostics` located outside. Plasma geometry is described by a 3D unstructured `Mesh` of `Zones`, which are uniform in temperature, density, and stoichiometry. Each `Zone` is defined by its bounding `Faces`. In the current implementation, a `Face` can be a convex `Polygon`, a `Sphere` (i.e., a spherical shell), and a `Cone` (a conical ribbon that appears in 2D geometry with RZ cylindrical symmetry). A `Surface` is a special, composite `Face` that consists of the other, more fundamental `Face` objects. In order to facilitate the modularity and extensibility of the FESTR geometry model, `Face` is made an abstract base class from which the actual `Face` types are derived and which provides the interface through which the raytracing is performed. Ultimately, each `Face` is defined by an ordered collection of `Nodes` that define points in 3D space. `Mesh` connectivity is stored by having each `Face` maintain the list of its neighbors, which are `Faces` from other `Zones` with which this `Face` is in surface contact.

Class `Ray` defines objects that carry the specific intensity [37] (units W cm^{-2} sr^{-1} eV^{-1}) of the spectrum and that also contain information about their current location and velocity (i.e., direction). In order for `Detector` patches to collect a spectrum at its location from a certain direction, each `Ray` is cast from its patch "backwards" into the `Mesh` to calculate its path, which is subsequently stored as a stack of `Waypoints`. These `Waypoints` (intersections of `Rays` with `Faces`) are calculated polymorphically by using the current `Ray` location and direction, as well as `Mesh` connectivity, via virtual functions declared in class `Face` (also, see Appendix B). With this arrangement, all the various `Face` types can coexist within the same calculation; also, new `Face` types can be added to FESTR in the future without having to modify the raytracing algorithm. The raytracing commences when a `Ray` cast by a `Detector` patch first hits the bounding `Sphere`, and it terminates when this `Ray`, having crossed the entire `Mesh`, hits the bounding `Sphere` again on the other side (see Appendix B for additional details).

2.3. Element mixing and radiation transport

After the raytracing procedure is completed, the `Ray` direction is reversed and its spectral content is initialized with a backlighter. At present, two backlighter types have been implemented: (1) a constant (flat) value (which could be zero), and, (2) a thermal spectrum (i.e., a Planckian) at a specified radiation temperature. Then, the stack of `Waypoints` is unwound as the `Ray` traverses the `Mesh` towards its destination, the original `Detector` patch. The distance between subsequent `Waypoints` defines the chord length L of the path the `Ray` spends within each `Zone` that it has encountered. `Zones` are, by definition, uniform and thus this length can be used to advance the specific intensity carried by the `Ray` according to the analytic solution of the steady-state radiation transport equation [37] in 1D (i.e., along a `Ray` parameterized by spatial coordinate s)

$$\frac{dI_\upsilon}{ds} = \varepsilon_\upsilon - \kappa_\upsilon I_\upsilon, \tag{1}$$

for a uniform medium of length L

$$I_\upsilon = I_\upsilon^0 e^{-\kappa_\upsilon L} + \frac{\varepsilon_\upsilon}{\kappa_\upsilon}\left(1 - e^{-\kappa_\upsilon L}\right) \tag{2}$$

where I_υ^0 and I_υ are, respectively, the spectra on entry into and on exit out of the `Zone`. The assumption of steady state eliminates the time-derivative term from the radiation transport equation by regarding the speed of light as infinite. This simplification is justified in ICF applications where rays travel practically instantaneously across the entire object.

The spectrally resolved emissivity ε_υ (units W cm^{-3} sr^{-1} eV^{-1}) and extinction coefficient κ_υ (cm^{-1}) for each `Zone` are computed using fundamental atomic structure and cross section data appropriate for the temperature and density conditions experienced

by the present chemical elements.[1] For reasons of modularity, portability, and computational expediency, FESTR was written as a Database driven code with respect to atomic data. Thus, any suitable source of atomic data can be used to drive FESTR; for our applications, we have used the ATOMIC code (fed by data from CATS/RATS, ACE, and GIPPER) from the LANL suite to compute the spectral part of the Database consisting of emissivity per atom $\varepsilon^A_{i\nu}$ (units W sr^{-1} eV^{-1}), and absorption and scattering per atom (units cm^2) values for the relevant chemical elements i on problem-appropriate grids indexed by electron temperature T_e (units eV, see Appendix A), radiation temperature T_r (units eV), and electron number density n_e (units cm^{-3}). The photon energy grid $h\nu$ (units eV) is dictated by the available Diagnostics. Average ionization values $\bar{z}$ (or $\langle z\rangle$, the number of free electrons per ion, unitless) for each element are stored in the equation-of-state (EOS) part of the Database in order to carry out the element mixing procedure described in the next paragraph.

While the temperature of a Zone is assumed to be known (see next two sections), the electron number density is *a priori* unknown, since it arises from the ionization processes as a value shared by all the elements that are present in the Zone. Given the Zone's stoichiometry, expressed via the atomic number densities n_i (units cm^{-3}) of individual elements, then electron number density n_e is given by the charge neutrality constraint

$$n_e = \sum_i n_i \bar{z}_i \left(T_e, T_r, n_e\right) . \tag{3}$$

We solve this implicit equation for n_e by performing a search across the EOS part of the Database for the closest available match (thus handling the possibility of the "real" density falling outside of the tabulated range). Once this n_e is known, we can construct the mixed emissivity ε_ν by querying the spectral part of the Database corresponding to this closest match (i.e., without any interpolation) and computing

$$\varepsilon_\nu = \sum_i n_i \varepsilon^A_{i\nu} \left(T_e, T_r, n_e\right) . \tag{4}$$

The mixed extinction coefficient κ_ν is calculated using an analogous formula. With this information now available, the Ray spectrum can be advanced across the Zone by means of Eq. (2) (see Appendix B for additional details). This procedure is similar to the approach described by Ref. [51] and also to the iterative technique implemented in the multi-element path of the ATOMIC code [52].

Finally, on arrival at the Detector, the spectrum can be aggregated into various integrated quantities (see the Diagnostics section, 2.1), and also convolved with the instrumental resolution function, which is hereby modeled as a normalized Gaussian of a given full width at half maximum (FWHM).

2.4. Hydrodynamic postprocessing

FESTR can be used to calculate synthetic spectral outputs by postprocessing any hydrodynamic simulation whose output can be cast into the FESTR format (see the src/Sample subdirectory contained in the downloadable TGZ archive file). This file format (currently ASCII) is a direct representation of the data structures used in the code, which, in turn, were designed to implement the chosen finite-element technique for integrating the radiation transport equation. The outermost loop in the code runs over the individual time intervals covering the time history from the hydrodynamic simulation. These time steps are completely independent of one another due to the assumption of steady-state in the radiation transport. The geometry of the problem (i.e., the Mesh) can be different for each time step. Class Hydro, with its temperature and density data for each Zone, along with fundamental material characteristics in classes Table and Database, is the conduit through which the code accesses this information.

2.5. Analysis of experimental data

For the purposes of analysis the FESTR code can be operated "in reverse" relative to the hydrodynamic postprocessing mode. In this analysis mode, the code searches given grids of temperatures and densities on a fixed Mesh for the conditions that result in spectral signals that best match available experimental data. Thus, the quantities T_e, T_r, n_i (formerly known from Hydro within each Zone) are now parameters to be found via search and optimization with their search ranges to be specified by the user in the Hydro input files. The data to be matched are defined as Objectives, which together form a Goal. At present, FESTR performs an exhaustive search across a given search grid by repurposing the time-step loop from the postprocessing mode, in which the former time index is redefined to represent individual search cases. The cost of this computationally very expensive approach limits the present search capabilities to rather small problems; our future plans include the implementation of more sophisticated methods such as the Pareto genetic algorithm for multi-objective optimization [53–55]. At present, some savings (i.e., computational time scaling linearly instead of exponentially with the number of Zones) can be achieved in cases with spherical symmetry, in which the plasma conditions can be retrieved by "peeling the onion", i.e., fitting experimental space-resolved spectra one spherical layer at a time starting on the outside and working toward the center (while using the results already found for all the previous, outer layers [39]). In this manner, the original multi-objective search and optimization problem is converted to a sequence of (more manageable) single-objective searches and fits. The main motivation behind this development is the planned ability to perform 3D reconstructions from multiple, simultaneously recorded MMI data while allowing for the detection of ion species separation [56–59].

2.6. FESTR class relationships

In this subsection we provide the following list, highlighting some of the important relationships between FESTR objects:

- A Mesh consists of Zones.
- A Zone is defined by its Faces.
- A Face is one of the following types: Polygon, Sphere, Cone, Surface. (This list is extensible.)
- A Face is defined by the integer labels of its Nodes.
- A Grid connects Nodes' integer labels with their actual locations (Vector3d objects).
- A Face maintains a list of its neighbors (other Faces) in order to facilitate Ray handovers between Zones during raytracing. (The bounding Sphere is a special Face that has no neighbors.)
- A Diagnostics is a collection of Detectors.
- A Goal is a collection of Objectives.
- A Ray carries with it its current location and velocity (both Vector3d) and the specific intensity of its spectrum (ArrDbl—an "array of doubles").
- A Ray's trajectory through the Mesh is represented by a stack of Waypoints.

[1] The removal of energy *from* a line of sight by scattering can be included in the extinction coefficient without difficulty here. However, the addition of scattering to the effective emissivity (i.e., intensity enhancement by energy scattered *into* a given line of sight) makes the radiation transport equation non-local; since this effect is not expected to be important in ICF applications, it is not included in the present version of FESTR.

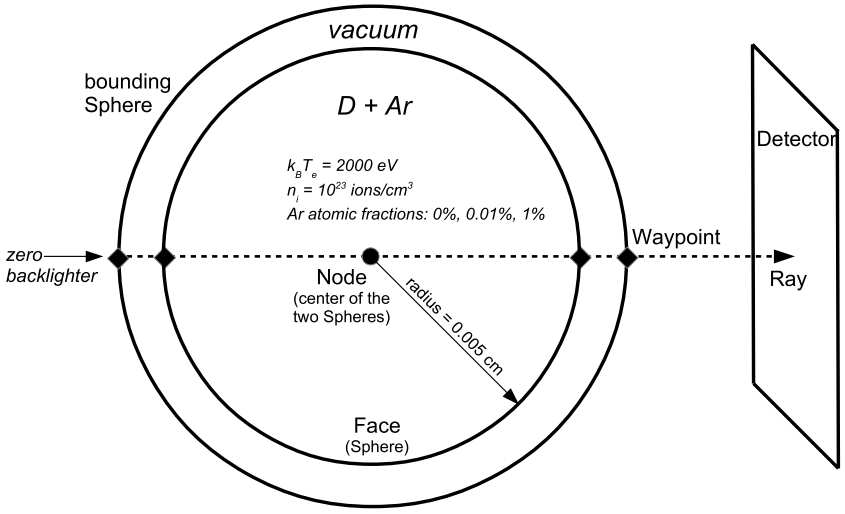

Fig. 2. Geometry of the example described in Section 4.

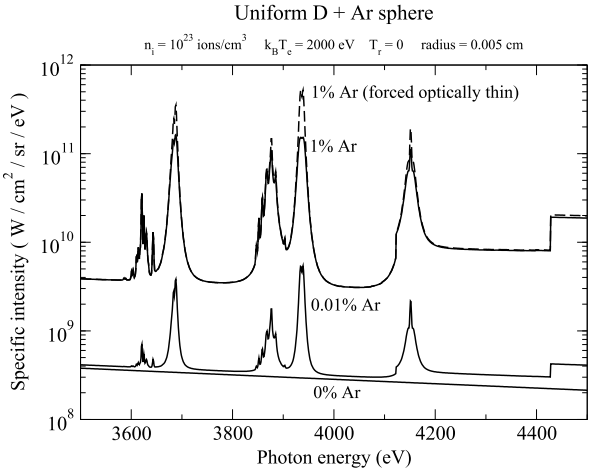

Fig. 3. Synthetic spectra calculated with FESTR using the geometry from Fig. 2; the underlying atomic and optical data were calculated (up to and including the gamma lines, see Table 1) via the fully relativistic path of the Los Alamos suite of codes [26,29].

2.7. Class `Progress`

The user is able to monitor the progress of a calculation via `Progress` objects placed throughout the FESTR source code. These objects are organized in a nested hierarchy of tasks and subtasks; for example, each time step contains a loop over `Detectors`, each of which is subdivided into patches and then even further into individual `Rays`. Certain input files (see Appendix C) contain frequency-of-print parameters that control how the completion of each stage of a calculation is reported to the user. For instance, if the frequency is set to 2, then the completion of every even-numbered task[2] within the scope of the associated `Progress` object is reported to the user; the completion of all odd-numbered tasks (and all their subtasks) is not reported in this case. The `Progress` object's place in this hierarchy is reflected in the level of indentation used in its progress reports. The code will also announce the creation of a new `Progress` object (by printing the completion of task 0) and the completion of its last task. If the total number of tasks is known in advance, this number is also reported and followed by a percentage of completion rounded to an integer. The raytracing task is an exception, as the number of `Waypoints` is *a priori* unknown; there the total number is replaced by "?" and the completion percentage is omitted. A frequency parameter set to 0 turns off these updates for that `Progress` object, and all its subtasks.

[2] `Progress` indexing of actual tasks starts at 1; task 0 is the creation of a new `Progress` object.

3. Code development, testing, and documentation

FESTR was written using test-driven development [60]; the code is broken up into a large number of rather short routines with clearly defined contracts binding the routines' preconditions with their postconditions. There are no global variables; data are accessed either by virtue of belonging to the same class, or by passing them as arguments. This modular design should also aid in the planned future parallelization of the code. Routines are accompanied by unit tests. While there are a number of sophisticated third-party unit-test frameworks available, we decided to develop a simple, native `Test` class to help with future portability of the code. These tests are defined by means of additional source code that is compiled and then linked with the actual FESTR code to generate a special, unit-test executable. No additional software needs to be installed to run these tests; the already assumed available C++ compiler is sufficient for this purpose.

The unit tests also serve as a form of code documentation, as they provide detailed examples of expected outputs (postconditions) for given inputs (preconditions). In addition, PDFs of scanned notes are enclosed with the distribution to illustrate certain concepts and to show derivations of equations. Finally, we also include a set of webpages generated by Doxygen [61] that provide an alternative insight into the source code; this documentation can be accessed by opening the file `doc/html/index.html` in a web browser.

4. Example

Fig. 2 shows the geometry of a simple example that illustrates the essential functionalities of FESTR. Here we investigate the spectral signal produced by a uniform ball of plasma of radius 0.005 cm, electron temperature 2000 eV (radiation temperature was set to zero), and total ion density of 10^{23} ions/cm^3. With most of the ions being deuterium, we study the spectral formation as various trace amounts of argon are introduced into the plasma. The results are shown in Fig. 3.

When no Ar is present, the spectrum consists of pure continuum from the ionized deuterium. At the chosen temperature of 2000 eV, Ar is not quite fully ionized (ionization potentials are 4121 eV for He-like Ar and 4426 eV for H-like Ar [62], see corresponding edges in Table 1) and therefore the spectrum acquires K-shell line features characteristic of the He-like and H-like Ar ions (see Table 1). At Ar fraction of 0.01% the lines are already quite prominent, but the continuum between lines still comes mostly from D. This regime therefore illustrates the complexities of spectral formation when multiple chemical elements are present in the plasma. At Ar fraction of 1% the spectrum is entirely dominated by Ar emission. By that point opacity (self-absorption) effects have begun to make their mark on the spectrum. The dashed line in Fig. 3 is a FESTR calculation for 1% Ar fraction, in which the extinction coefficient was artificially zeroed out (i.e., the optically thin approximation was invoked). Thus the observed reduction of the actual intensity around line centers, combined with having no discernible change in the continuum between the lines, is a testament to the importance of including opacity effects in spectroscopic modeling. In fact, in cases with a non-zero backlighter (whether external, or self-generated via spatial non-uniformity), some spectral lines can remain visible in emission while others may already flip into absorption (see Fig. 7 in Ref. [23] for results calculated by a 1D predecessor to the FESTR code).

5. Summary

We have developed a new spectral modeling code named FESTR that can be used for postprocessing of hydrodynamic simulations and for analysis of experimental data. The implemented model includes the effects of spatial non-uniformity, mixing of elements in hot plasmas, and radiation transport. FESTR has been released under the 3-clause BSD license and is available for download at CPC and GitHub (https://github.com/LANLhakel/FESTR).

Acknowledgments

Los Alamos National Laboratory is operated by Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under Contract No. DE-AC52-06NA25396. We thank C.J. Fontes, H.M. Johns, and D.P. Kilcrease for valuable suggestions during the preparation of this manuscript.

Appendix A. Unit conversions

Quantity	FESTR unit	SI unit equivalent
Specific intensity	$\mathrm{W\,cm^{-2}\,sr^{-1}\,eV^{-1}}$	$4.136\times10^{-11}\ \mathrm{W\,m^{-2}\,sr^{-1}\,Hz^{-1}}$
Emissivity	$\mathrm{W\,cm^{-3}\,sr^{-1}\,eV^{-1}}$	$4.136\times10^{-9}\ \mathrm{W\,m^{-3}\,sr^{-1}\,Hz^{-1}}$
Absorption, scattering, extinction coefficient	$\mathrm{cm^{-1}}$	$10^{2}\ \mathrm{m^{-1}}$
Particle number density	$\mathrm{cm^{-3}}$	$10^{6}\ \mathrm{m^{-3}}$
Emissivity per atom	$\mathrm{W\,sr^{-1}\,eV^{-1}}$	$4.136\times10^{-15}\ \mathrm{W\,sr^{-1}\,Hz^{-1}}$
Absorption, scattering, extinction coefficient per atom	$\mathrm{cm^{2}}$	$10^{-4}\ \mathrm{m^{2}}$
Temperature	eV (with implied multiplication by the Boltzmann constant k_B)	11,605 K
Energy	eV	1.602×10^{-19} J

Table 1
Main spectral emission features of argon ions in Fig. 3; energies are from Ref. [62].

Argon spectral feature	Transition	Energy (eV)
Helium-beta (He-β)	$1s\,3p \rightarrow 1s^2$	3684
Helium-gamma (He-γ)	$1s\,4p \rightarrow 1s^2$	3875
Helium edge	$1s\,\varepsilon l \rightarrow 1s^2$	4121
Lyman-beta (Ly-β)	$3p \rightarrow 1s$	3936
Lyman-gamma (Ly-γ)	$4p \rightarrow 1s$	4150
Lyman edge	$\varepsilon l \rightarrow 1s$	4426

Appendix B. Details of raytracing and radiation transport

`Face` is an abstract base class from which actual `Zone` boundaries are derived as objects of type `Sphere`, `Cone`, `Polygon`, and `Surface`. The physical location and shapes of these objects are based on information stored in `Grid`, `Node`, and `Vector3d` objects. Class `Face` declares a pure virtual function `intercept`, which is implemented in all classes derived from `Face` to calculate the next `Waypoint` as the intersection of a straight line (a `Ray`) with the appropriate specific geometric shape (`Sphere`, `Cone`, `Polygon`, or `Surface`) in 3D space. Once the path of a `Ray` through the `Mesh` has been calculated during the raytracing phase, its spectrum (specific intensity I_ν) is initialized according to the backlighter settings in the originating `Detector` object for this `Ray`.

During the subsequent radiation transport phase (see Fig. 4) a `Ray` enters a `Zone` carrying the spectrum I_ν^0 which had been built up along its previous track across the `Mesh`. The physical length L that the `Ray` will travel in the current `Zone` is calculated as the distance between the two `Waypoints` belonging to the current `Zone`. Monochromatic emissivity and extinction coefficients are calculated from the `Hydro`-provided temperature and density conditions for this `Zone` (see Eqs. (3) and (4)) using the contents of the `Database`, with material names handled by the `Table` object (see Appendix C). The spectrum carried by the `Ray` is then updated according to Eq. (2) and it is this spectrum I_ν with which the `Ray` will exit the current `Zone` and enter the next one. At that point I_ν becomes the new I_ν^0 for the next `Zone` and this process repeats until the `Ray` crosses the entire `Mesh`. Finally, the acquired spectrum is processed by the `Detector` that had originally created this `Ray` and cast it across the `Mesh`.

Appendix C. Description of FESTR input setup

The names of many input and output files contain integer labels for items such as the time-step index and spatial-patch coordinates in a `Detector`. The number of digits used (and hence, the amount of left-padding with zeros) is determined automatically to match the minimum number of digits necessary to accommodate the largest number for each quantity in a given calculation.

The simplest way to compile FESTR is to execute the command

```
make
```

in the `src` directory, which will use the g++ compiler to create the `festr` executable file in the same directory. The code takes one additional command-line argument to run:

```
./festr <options_file>
```

where `<options_file>` represents the name of the file containing options that define the calculation (a sample `<options_file> = options.txt` file already exists in this directory). In this Appendix we describe the contents of this file and the formats of the associated input and output files for FESTR. The option names are followed by a colon and must be provided in this given order in the file `<options_file>`.

$$\frac{dI_\nu}{ds} = \varepsilon_\nu - \kappa_\nu I_\nu \Longrightarrow I_\nu = I_\nu^0 e^{-\kappa_\nu L} + \frac{\varepsilon_\nu}{\kappa_\nu}\left(1 - e^{-\kappa_\nu L}\right)$$

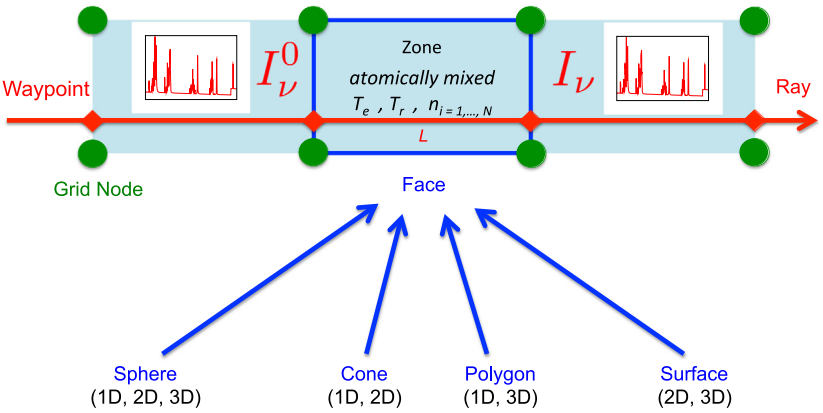

Fig. 4. Schematic of raytracing and radiation transport implementation in FESTR (color online).

`Top_path:`

This option defines the directory relative to which other directories with data are specified below.

`Goal:`

Specifying `none` launches FESTR in the hydro post-processing mode; otherwise this option specifies the directory containing the experimental spectra to be matched by running FESTR in the analysis mode. For instance, the contents of the directory `src/Test/Goal1` define a four-objective search-and-optimization problem used by the relevant unit tests. The `Objective` names in the 1st column of file `list.txt` correspond to the eponymous files (with `.txt` file name extension) containing the data to be matched. The 2nd column defines the overall weight of each `Objective` within the multi-objective `Goal`. The 3rd column is set to `true`, if the `Objective` file also contains the abscissas column. The 4th column is `true`, if the `Objective` file also contains the individual weights column (these weights default to 1, if this flag is set to false). Setting the 5th column to `true` allows FESTR to automatically match units between the calculated and experimental data by optimally rescaling the calculated spectrum before the fitness of the match is evaluated. The names and properties of individual `Objectives` must match `Detector` objects, see item `Diagnostics` below.

Two additional codes, that support the preparation of `Objective` files, are included with this distribution. The first auxiliary code, `src/Scale/scalexy.cpp`, can be used to calibrate both the abscissas and ordinates of the new `Objective` file according to the contents of the input file named `scale.txt`. This step is often necessary in order to reconcile the spectral line positions (abscissas) between the `Detector` and the `Database` before least-squares fitting, as well as to flat-field the experimental data due to the intensity response (ordinates) of the recording instrument. The second auxiliary code, `src/ObjPrep/objprep.cpp`, attaches the weights (either 0 or 1) to the new `Objective` file in order to define the photon energy bands (given in the input file named `ranges.txt`), that will be active in the least-squares fitting routine.

`Output:`

This option defines the directory into which all output data will be placed by the code.

`Material_table_path:`

This option defines the directory containing the list of materials used by FESTR (see option `Material_table_file_name`).

`Material_table_file_name:`

The contents of this file define FESTR's `Table` object, which facilitates the connection between the material names used by `Hydro` (1st column) and File handle (4th column) used by `Database` (for details on both, see their respective options below).

`Hydro:`

This option defines the directory containing the materials' geometric distributions in space and their physical conditions such as temperatures and densities. The integers in file names enumerate the time step index—individual time instants (in seconds) are given in file `times.txt` (the last time instant is only used to derive the duration of the last time interval). File `materials.txt` lists those materials from the 1st column of file `Material_table_path/Material_table_file_name` that are present in this calculation. File `bounding_sphere.txt` defines the x, y, z coordinates of the center of the bounding `Sphere`, followed by its radius (all in cm). The bounding `Zone` is labeled 0, its outer `Face` is the bounding `Sphere` labeled (0, 0) and its inner `Face` (in contact with the rest of the `Mesh`) is a `Surface` labeled (0, 1) (see the definition of `FaceID` in Figs. 5 and 6).

In Figs. 5 and 6 we provide detailed explanations and a graphical representation of the `Hydro` input files for time index 0 (label visible in all file names following an underscore) stored in the directory `src/Test/Hydro1`. File `grid_0.txt` contains the coordinates (cm) and velocities (cm/s, currently unused) of eight nodes (labels 0 through 7) as shown in Fig. 5. In Fig. 6 we highlight selected parts of file `mesh_0.txt` which defines all the necessary geometric objects shown in Fig. 5. (Note that, at present, all `Polygons` are assumed to be convex; the generalization of methods such as `Polygon::contains` to accommodate concave `Polygons` is left as future work.)

For examples of `Faces` of type `Cone` see directory `src/Test/Hydro4`. This class implements rotation surfaces specified in a plane in 2D and then spun around the z-axis in RZ cylindrical symmetry. In this case, files `grid_*.txt` only utilize the first two coordinates ($x \rightarrow R$, $y \rightarrow Z$) and in files `mesh_*.txt` only two `Nodes` are used to define a `Cone`.

We point out that the existence of class `Surface` is a consequence of defining the first integer in `FaceID` as type `size_t` (to accommodate a potentially large number of `Zones` in the calculation) and the second integer as `short int` to save memory, since in general each `Zone` only has a handful of `Faces` and thus a large index range is not necessary there. In 2D

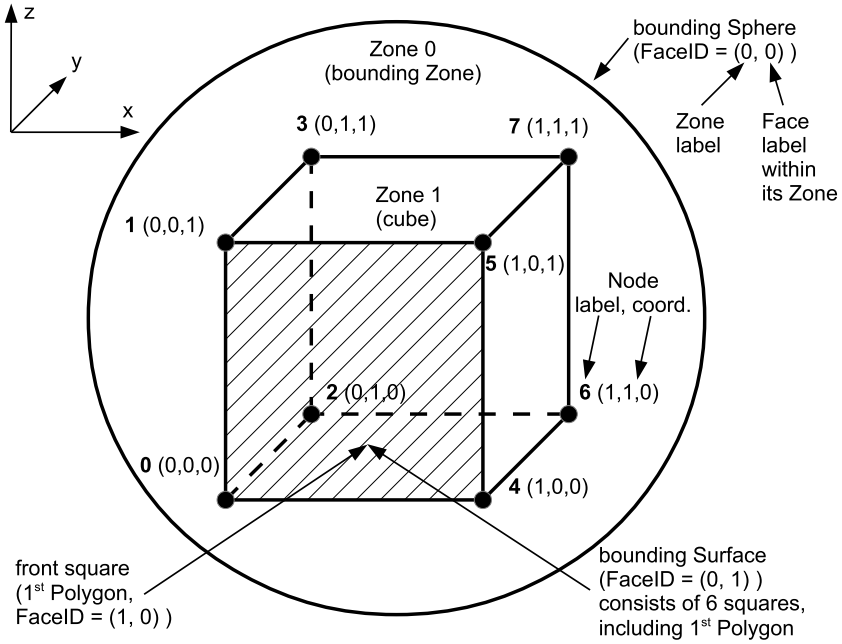

Fig. 5. Geometry at time index 0 for `Hydro` from `src/Test/Hydro1`.

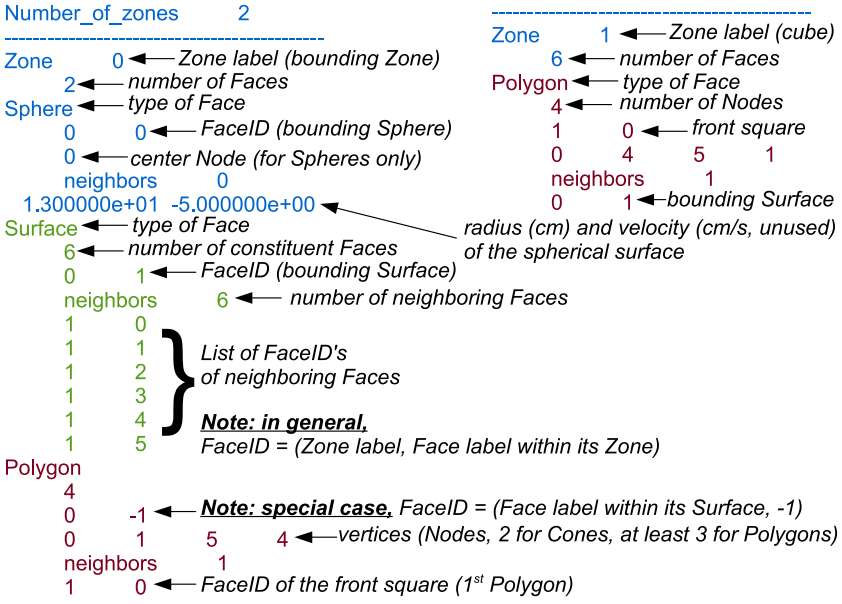

Fig. 6. Annotated parts of file `src/Test/Hydro1/mesh_0.txt` (file contents use regular font in color online), annotations are in black italics with arrows.

and 3D geometries, however, this may not be the case for the bounding `Zone` as its inner surface can consist of a large number of `Faces` forming the outer surface of the rest of the `Mesh`. In order to treat the bounding `Zone` on the same footing with all the other `Zones` for raytracing and radiation-transport purposes, class `Surface` was created as a composite `Face` that consists of other, fundamental objects derived from `Face`. That way, the bounding `Zone` formally can have only two `Faces`, the outer (bounding) `Sphere` (0, 0) and the inner `Surface` (0, 1). Within this `Surface` then, the constituent `Faces` are labeled using the *first* index in their `FaceID` (which is of type `size_t` and hence can handle large values), and the second index is set to −1 as a flag indicating that this `Face` is part of a `Surface`.

In the post-processing mode, files `time_*.txt` store the material conditions for all the `Zones` in the `Mesh`: `te` is the electron temperature (eV), `tr` is the radiation temperature (eV), `np` is the total particle number density (cm^{-3}), `nmat` is the number of materials in this `Zone`, followed by the material names (as specified in file `materials.txt` and in the 1st column of file `Material_table_path/Material_table_file_name`) accompanied by their *number* fractions, which must add up to 1. The bounding `Zone` is usually specified as vacuum, i.e. with `nmat = 0`.

In the analysis mode, there is formally only one time interval and FESTR searches for the temperature and density values on a fixed `Mesh` that best satisfy the `Objectives` of the given `Goal`. File `time_0.txt` contains the definitions of electron and radiation temperature search grids and also the particle-density search grids, which are specified in the `material densities` section (in contrast to the post-processing mode, the `np` entry is not present in the analysis mode). For example, for `Zone 1` in file `src/Test/Analysis1/time_0.txt` we have

```
te 6100.0 6400.0 4 lin eV
```

which will cause the code to consider four electron temperatures on a linear grid between 6100 and 6400 eV (i.e., 6100, 6200, 6300, 6400). The same file and Zone also specify

```
d 1.0e19 2.0e19 5 lin
```

to define the search grid for atom densities of deuterium. For densities, FESTR will always include the value of *zero* in the search grid in addition to the user specification; therefore, here the code will consider *six* densities for deuterium (0, 1.00e19, 1.25e19, 1.50e19, 1.75e19, 2.00e19).[3] In order to manage these search grids in the code, class Cell was created as a counterpart of the original class Zone. Then, each candidate case is formally treated as a separate "time step" in which Zones are filled with materials retrieved from the search grids in their corresponding Cell objects, spectra are calculated as in the post-processing mode, and their fitness is evaluated against the Objectives of the given Goal. In the end, files best_case.txt (formatted as time_0.txt) and best_case.dat (formatted in columns for plotting) are written out along with the best matching spectral outputs.

TOPS_command:

This option is always set to none in the publicly released version of FESTR, as the invocation of the TOPS code [63] is only available within the computing environment of the Los Alamos National Laboratory.

Database:

This option defines the directory housing the atomic databases interrogated by FESTR. The associated data are divided into three subdirectories: grids, eos, and spectra.

Subdirectory grids contains files describing the grids on which the atomic data are defined; these files are te_grid.txt (electron temperature grid), tr_grid.txt (radiation temperature grid), ne_man_grid.txt (electron number density–mantissa grid), ne_exp_grid.txt (electron number density–exponent grid), and hv_grid.txt (photon energy grid). The electron number density grid used by the Database is formed in the code as the direct product of its corresponding mantissa and exponent grids. All of these files contain a Number of bits record, which is currently unused, but is already included here in the anticipation of a future implementation of genetic-algorithm-based techniques in the analysis mode (see Section 2.5).

Subdirectory eos contains ion-stage fractional populations and average ionization degrees (zbar is the label in the contents of the file, zb is used in the file name) for all electron temperature, radiation temperature, and electron number density values specified on the grids described above.

Subdirectory spectra houses data for emission (file name containing em), absorption (ab), and scattering (sc), all in units per atom, on the same (T_e, T_r, n_e) grid as the EOS data, and monochromatically resolved on the photon energy grid defined in the grids/hv_grid.txt file. All file names begin with the File handle string, as specified in the 4th column of the file Material_table_path/Material_table_file_name and are placed in subdirectories named after the same File handle string. The three grid parameters (T_e, T_r, n_e) are encoded in each file name; the electron number density is as shown in its two grid files, and the temperatures are integers with five digits (left-padded with zeros) in the current version of the code. For example, data for hydrogen at $T_e = 6400$ eV, $T_r = 0$ eV, $n_e = 10^{16}$ cm^{-3} are stored in the files located in the directory <Database>:

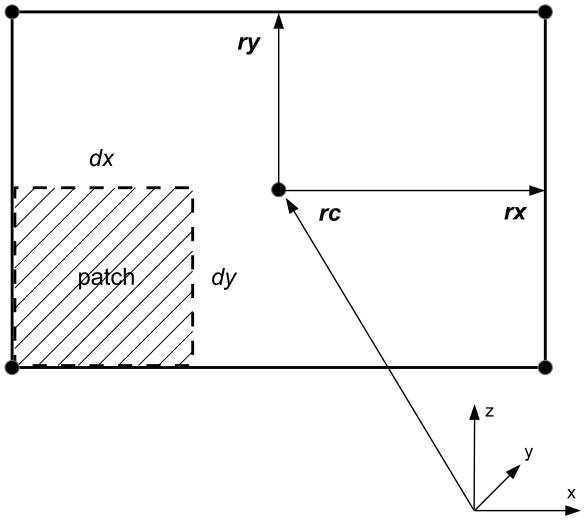

Fig. 7. Schematic of a rectangular Detector object.

eos/z01/z01_te06400ev_tr00000ev_ne5.0e16pcc_zb.txt (ion-stage fractional populations, average ionization degree),

spectra/z01/z01_te06400ev_tr00000ev_ne5.0e16pcc_em.txt (monochromatic emission per atom),

spectra/z01/z01_te06400ev_tr00000ev_ne5.0e16pcc_ab.txt (monochromatic absorption per atom),

spectra/z01/z01_te06400ev_tr00000ev_ne5.0e16pcc_sc.txt (monochromatic scattering per atom).

The photon energy grid is not included in these files in order to save space. These same files could also be used for deuterium and tritium since FESTR is designed to work with particle number densities (as opposed to mass densities) and isotope effects (such as differences in thermal Doppler broadening of spectral lines) are negligible in our regimes of interest. Therefore, in general, all isotopes of a given element share the same File handle and hence access the same database files.

Diagnostics:

This option specifies the directory holding the definitions of all Detectors, which ultimately determine how individual Rays are sent across the Mesh. File list.txt starts with the frequency parameter for its associated Progress object (see Section 2.7) monitoring either the time steps (in postprocessing mode) or search cases (in analysis mode), followed by the number of Detectors in this Diagnostics. Each Detector is given a name, angular discretizations for its Ray bundles (setting ntheta = nphi = 0 invokes the parallel Ray approximation), and the frequency of Progress prints for individual Rays.

The same directory also contains a file for each Detector named <dname>.txt where <dname> is the Detector name given in the file list.txt. The <dname> entry is also the first record in this Detector-specific file. The path record defines a subdirectory within the Output directory (see the beginning of this Appendix) into which all output files associated with this Detector will be placed. Setting path to "/" will result in output files being placed directly into the Output directory. All these output directories must already exist when the code is launched; at present FESTR does not create its own output directories at runtime. Next, frequencies of prints for tracking progress on individual patches and Rays are defined. The symmetry option can be none (retaining rectangular patches) or spherical (overlaying annular patches across the Detector).

In Fig. 7 we display a schematic of a rectangular Detector (symmetry = none). The center of the Detector is defined as rc

[3] The reason for introducing this feature is that we wanted FESTR always to be able to consider zero density (i.e., the absence of a material) as a candidate for the best solution, including those cases for which densities are specified on a logarithmic grid by replacing lin with log.

and its orientation and size are specified via `rx` and `ry`. The patch sizes in each direction are given as `dx` and `dy`. All dimensions are in centimeters; `ry` and `dy` are ignored if `symmetry = spherical`. The photon energy range of interest can be limited with the `range` option. In order to simulate the finite spectral resolution of a diagnostic instrument, the calculated spectra can be convolved with a Gaussian profile whose width is specified by the `fwhm` parameter; a value of `fwhm` that is negative or zero will bypass this step. Finally, each `Ray` from this `Detector` is initialized with a `backlighter`; this can be a constant value in W cm^{-2} sr^{-1} eV^{-1} following the word `flat`, or a thermal spectrum of a specified radiation temperature given in eV following the word `blackbody`. The associated `Progress` object is set up so that the completion of *all* `Detectors` is reported, provided the completion of the encompassing time step (in postprocessing mode) or search case (in analysis mode) is also being announced to the user.

Appendix D. Description of FESTR output

The output files are located either directly in the `Output` directory or in its subdirectories, according to individual `Detector` settings described in Appendix C. Each output file name starts with the `Detector` name followed by a hyphen, after which a string describing the type of data is appended, and the file name ends with the extension `.txt`. For instance, for a `Detector` named `MMI`, FESTR generated the following output files during one of its runs:

`MMI-hv_grid.txt`: photon-energy grid (eV) on which spectra are calculated for every `Ray` associated with this `Detector`.

`MMI-backlighter.txt`: backlighter values (W cm^{-2} sr^{-1} eV^{-1}) used to initialize every `Ray` associated with this `Detector`.

`MMI-yp_ix071_iy0_it0_ip0_time3.txt`: spectrum (W cm^{-2} sr^{-1} eV^{-1}) for the `Ray` issued from patch with the *x*-coordinate index `ix = 71` and *y*-coordinate index `iy = 0`, cast into the direction within its `Ray` bundle defined by the polar-angle (theta) index `it = 0` and the azimuthal angle (phi) index `ip = 0`, covering the `Hydro` time interval number 3. If the parallel `Ray` approximation is invoked, then the names of all of these files always contain `it0_ip0`. If `symmetry` defined in the `Detector` is `spherical`, all file names always contain `iy0`. The `symmetry = spherical` is only available if the parallel `Ray` approximation is also invoked.

`MMI-yp_ix071_iy0_time3.txt`: spectrum for the patch with the *x*-coordinate index `ix = 71` and *y*-coordinate index `iy = 0`, covering the `Hydro` time interval number 3. If the parallel `Ray` approximation is invoked (i.e., there is only one `Ray` for each patch), the contents of this file are identical (including the unit W cm^{-2} sr^{-1} eV^{-1}) to the corresponding output file whose name contains `it0_ip0`. If, on the other hand, there are many `Rays` arriving at this patch from their various directions (enumerated by their theta and phi angles), then this file contains the spectrum obtained by integrating these spectra over this `Ray` bundle's solid angle, and multiplied by the area of the patch, yielding a spectrum in units W eV^{-1}.

`MMI-ys_time3.txt`: space-integrated spectrum covering the `Hydro` time interval number 3. If the parallel `Ray` approximation is invoked, the space integration is carried out across the source (the `Mesh`) resulting in the units W sr^{-1} eV^{-1}; otherwise the space integration is performed by adding spectra from all patches across this `Detector`, in which case the spectrum is given in units W eV^{-1}.

`MMI-yt_ix071_iy0.txt`: time-integrated spectrum for the patch with the *x*-coordinate index `ix = 71` and *y*-coordinate index `iy = 0`, covering the entire `Hydro` time history. If the parallel `Ray` approximation is invoked, the units are J cm^{-2} sr^{-1} eV^{-1}; otherwise the spectrum is given in units J eV^{-1} which includes the multiplication by the area of the patch.

`MMI-yst.txt`: space and time-integrated spectrum covering the entire `Hydro` time history. If the parallel `Ray` approximation is invoked, the space integration is carried out across the source (the `Mesh`) resulting in the units J sr^{-1} eV^{-1}; otherwise the space integration is performed by adding spectra from all patches across this `Detector`, in which case the spectrum is given in units J eV^{-1}.

In order to save disk space, in these output spectral files we do not repeat the photon-energy grid shown in the file `MMI-hv_grid.txt`; consequently, these files are not ready for direct plotting. Instead, an auxiliary code named `plotxy` can be created by executing the `src/Plot/c_plotxy` script and then moved to the `Output` directory. Then, for instance, executing the command

```
./plotxy MMI-yst
```

will merge the photon-energy grid from the file `MMI-hv_grid.txt` as the abscissas and the data from the file `MMI-yst.txt` as ordinates into a two-column file named `MMI-yst.dat` for plotting. This same `plotxy` code can also be used in a similar fashion to help visualize the contents of the `Database` spectral files. For example, (referring to Appendix C) after the user has copied the `plotxy` executable and the `<Database>/grids/hv_grid.txt` into the directory `<Database>/spectra/z01`, then executing

```
./plotxy z01_te06400ev_tr00000ev_ne5.0e16pcc_em
```

will create the file named `z01_te06400ev_tr00000ev_ne5.0e16pcc_em.dat` that can be plotted.

The directory `src/Histories` contains another auxiliary code called `time_hist` that can be used to further process FESTR output. The compilation is achieved by executing the `c_time_hist` script. For the present example, invoking the resulting executable in the `Output` directory with the following arguments:

```
./time_hist MMI-ys 3600 3700
```

will generate the file named `MMI-ys_3600_3700.dat`, which contains time history of the space-integrated signal integrated across the photon-energy band from 3600 to 3700 eV.

Finally, synthetic images can be created via the Python script `img.py` located in the directory `img`; we refer the user to the top of this file for the description of its usage. The data displayed by this Python script need not have any special symmetry. However, in the present version, only spherically symmetric data from FESTR can be distilled into the proper format for `img.py`; this task is accomplished via the program `sph_img.cpp` in the same directory (see this source file for an example of its usage).

References

[1] A.N. Cox, J.N. Stewart, D.D. Eilers, Astrophys. J. Suppl. 11 (1965) 1–21.
[2] M.J. Seaton, J. Phys. B: At. Mol. Phys. 20 (1987) 6363–6378.
[3] I. Hubeny, D.G. Hummer, T. Lanz, Astron. Astrophys. 282 (1994) 151–167.
[4] V.L. Jacobs, M. Blaha, Phys. Rev. A 21 (1980) 525–546.
[5] D. Duston, J. Davis, Phys. Rev. A 23 (1981) 2602–2621.
[6] J.C. Kieffer, J.P. Matte, M. Chaker, Y. Beaudoin, C.Y. Chien, S. Coe, G. Mourou, J. Dubau, M.K. Inal, Phys. Rev. E 48 (1993) 4648–4658.
[7] H. Nishimura, T. Kiso, H. Shiraga, T. Endo, K. Fujita, A. Sunahara, H. Takabe, Y. Kato, S. Nakai, Phys. Plasmas 2 (1995) 2063–2074.
[8] R.C. Mancini, A.S. Shlyaptseva, P. Audebert, J.P. Geindre, S. Bastiani, J.C. Gauthier, G. Grillon, A. Mysyrowicz, A. Antonetti, Phys. Rev. E 54 (1996) 4147–4154.
[9] M.E. Sherrill, R.C. Mancini, J.E. Bailey, A. Filuk, B. Clark, P. Lake, J. Abdallah Jr., Rev. Sci. Instrum. 72 (2001) 957–960.
[10] U. Andiel, K. Eidmann, P. Hakel, R.C. Mancini, G.C. Junkel-Vives, J. Abdallah, K. Witte, Europhys. Lett. 60 (2002) 861–867.
[11] S.B. Hansen, D.J. Ampleford, M.E. Cuneo, N. Ouart, B. Jones, C.A. Jennings, A. Dasgupta, C.A. Coverdale, G.A. Rochau, G. Dunham, J.L. Giuliani, J.P. Apruzese, Phys. Plasmas 21 (2014) 031202-1-5.
[12] I.M. Hall, T. Durmaz, R.C. Mancini, J.E. Bailey, G.A. Rochau, I.E. Golovkin, J.J. MacFarlane, Phys. Plasmas 21 (2014) 031203-1-7.

[13] A.S. Safronova, V.L. Kantsyrev, A.A. Esaulov, A.S. Chuvatin, M.E. Weller, V.V. Shlyaptseva, I. Shrestha, S.F. Keim, A. Stafford, C.A. Coverdale, J.P. Apruzese, N.D. Ouart, J.L. Giuliani, Phys. Plasmas 21 (2014) 031205-1-7.
[14] T. Nagayama, J.E. Bailey, G. Loisel, S.B. Hansen, G.A. Rochau, R.C. Mancini, J.J. MacFarlane, I. Golovkin, Phys. Plasmas 21 (2014) 056502-1-15.
[15] V.V. Ivanov, R.C. Mancini, D. Papp, P. Hakel, T. Durmaz, R. Florido, Phys. Plasmas 21 (2014) 082704-1-8.
[16] D.A. Haynes Jr., D.T. Garber, C.F. Hooper Jr., R.C. Mancini, Y.T. Lee, D.K. Bradley, J. Delettrez, R. Epstein, P.A. Jaanimagi, Phys. Rev. E 53 (1996) 1042–1050.
[17] N.C. Woolsey, A. Asfaw, B. Hammel, C. Keane, C.A. Back, A. Calisti, C. Mossé, R. Stamm, B. Talin, J.S. Wark, R.W. Lee, L. Klein, Phys. Rev. E 53 (1996) 6396–6402.
[18] C.J. Fontes, J. Abdallah Jr., R.E.H. Clark, D.P. Kilcrease, J. Quant. Spectrosc. Radiat. Transfer 65 (2000) 223–230.
[19] I.E. Golovkin, R.C. Mancini, J. Quant. Spectrosc. Radiat. Transfer 65 (2000) 273–286.
[20] L.A. Welser, R.C. Mancini, J.A. Koch, N. Izumi, H. Dalhed, H. Scott, T.W. Barbee Jr., R.W. Lee, I.E. Golovkin, F. Marshall, J. Delettrez, L. Klein, J. Quant. Spectrosc. Radiat. Transfer 81 (2003) 487–497.
[21] J.J. MacFarlane, I.E. Golovkin, R.C. Mancini, L.A. Welser, J.E. Bailey, J.A. Koch, T.A. Mehlhorn, G.A. Rochau, P. Wang, P. Woodruff, Phys. Rev. E 72 (2005) 066403-1-14.
[22] H.M. Johns, R.C. Mancini, P. Hakel, T. Nagayama, V.A. Smalyuk, S.P. Regan, J. Delettrez, Phys. Plasmas 21 (2014) 082711-1-12.
[23] J.A. Baumgaertel, P.A. Bradley, S.C. Hsu, J.A. Cobble, P. Hakel, I.L. Tregillis, N.S. Krasheninnikova, T.J. Murphy, M.J. Schmitt, R.C. Shah, K.D. Obrey, S. Batha, H. Johns, T. Joshi, D. Mayes, R.C. Mancini, T. Nagayama, Phys. Plasmas 21 (2014) 052706-1-9.
[24] P. Hakel, G.A. Kyrala, P.A. Bradley, N.S. Krasheninnikova, T.J. Murphy, M.J. Schmitt, I.L. Tregillis, R.J. Kanzleiter, S.H. Batha, C.J. Fontes, M.E. Sherrill, D.P. Kilcrease, S.P. Regan, Phys. Plasmas 21 (2014) 063306-1-11.
[25] C.J. Fontes, J. Colgan, J. Abdallah Jr., in: Yu. Ralchenko (Ed.), Modern Methods in Collisional-Radiative Modeling of Plasmas, Springer International Publishing, Switzerland, 2016, pp. 17–50.
[26] C.J. Fontes, H.L. Zhang, J. Abdallah Jr., R.E.H. Clark, D.P. Kilcrease, J. Colgan, R.T. Cunningham, P. Hakel, N.H. Magee, M.E. Sherrill, J. Phys. B: At. Mol. Opt. Phys. 48 (2015) 144014-1-17.
[27] R.D. Cowan, The Theory of Atomic Structure and Spectra, University of California Press, Berkeley, California, 1981.
[28] J. Abdallah Jr., R.E.H. Clark, R.D. Cowan, CATS: Cowan Atomic Structure Code, Los Alamos Manual No. LA-11436-M-I, Los Alamos, New Mexico, 1988.
[29] D.H. Sampson, H.L. Zhang, C.J. Fontes, Phys. Rep. 477 (2009) 111–214.
[30] R.E.H. Clark, J. Abdallah Jr., G. Csanak, J.B. Mann, R.D. Cowan, ACE: Another Collisional Excitation Code, Los Alamos Manual No. LA-11436-M-II, Los Alamos, New Mexico, 1988.
[31] R.E.H. Clark, J. Abdallah Jr., J.B. Mann, Astrophys. J. 381 (1991) 597–600.
[32] N.H. Magee, J. Abdallah, J. Colgan, P. Hakel, D.P. Kilcrease, S. Mazevet, M. Sherrill, C.J. Fontes, H.L. Zhang, in: J.S. Cohen, S. Mazevet, D.P. Kilcrease (Eds.), 14th Topical Conference on Atomic Processes in Plasmas, AIP Conference Proceedings, New York, 2004, pp. 168–179.
[33] P. Hakel, M.E. Sherrill, S. Mazevet, J. Abdallah Jr., J. Colgan, D.P. Kilcrease, N.H. Magee, C.J. Fontes, H.L. Zhang, J. Quant. Spectrosc. Radiat. Transfer 99 (2006) 265–271.
[34] J. Colgan, D.P. Kilcrease, N.H. Magee, M.E. Sherrill, J. Abdallah Jr., P. Hakel, C.J. Fontes, J.A. Guzik, K.A. Mussack, Astrophys. J. 817 (2016) 116-1-10.
[35] J. Colgan, D.P. Kilcrease, N.H. Magee Jr., J. Abdallah Jr., M.E. Sherrill, C.J. Fontes, P. Hakel, H.L. Zhang, High Energy Density Phys. 14 (2015) 33–37.
[36] J.J. MacFarlane, I.E. Golovkin, P. Wang, P.R. Woodruff, N.A. Pereyra, High Energy Density Phys. 3 (2007) 181–190.
[37] D. Mihalas, Stellar Atmospheres, Freeman and Co., San Francisco, California, 1970.
[38] P. Hakel, X-ray Line Spectral Signatures of Plasmas Driven by High-Intensity Ultra-Short Laser Pulses, University of Nevada, Reno dissertation, Reno, Nevada, 2001.
[39] L. Welser-Sherrill, D.A. Haynes, R.C. Mancini, J.H. Cooley, R. Tommasini, I.E. Golovkin, M.E. Sherrill, S.W. Haan, High Energy Density Phys. 5 (2009) 249–257.
[40] J. Colgan, E.J. Judge, H.M. Johns, D.P. Kilcrease, J.E. Barefield II, R. McInroy, P. Hakel, R.C. Wiens, S.M. Clegg, Spectrochim. Acta B 110 (2015) 20–30.
[41] B. Yaakobi, F.J. Marshall, D.K. Bradley, Appl. Opt. 37 (1998) 8074–8080.
[42] J.A. Koch, T.W. Barbee Jr., S. Dalhed, S. Haan, N. Izumi, R.W. Lee, L.A. Welser, R.C. Mancini, F.J. Marshall, D. Meyerhofer, T.C. Sangster, V.A. Smalyuk, J.M. Soures, L. Klein, I. Golovkin, in: J.S. Cohen, S. Mazevet, D.P. Kilcrease (Eds.), 14th Topical Conference on Atomic Processes in Plasmas, AIP Conference Proceedings, New York, 2004, pp. 53–60.
[43] N. Izumi, T.W. Barbee, J.A. Koch, R.C. Mancini, L.A. Welser, Rev. Sci. Instrum. 77 (2006) 083504-1-4.
[44] R. Tommasini, J.A. Koch, N. Izumi, L.A. Welser, R.C. Mancini, J. Delettrez, S. Regan, V. Smalyuk, Rev. Sci. Instrum. 77 (2006) 10E303-1-3.
[45] L. Welser-Sherrill, R.C. Mancini, J.A. Koch, N. Izumi, R. Tommasini, S.W. Haan, D.A. Haynes, I.E. Golovkin, J.J. MacFarlane, J.A. Delettrez, F.J. Marshall, S.P. Regan, V.A. Smalyuk, G. Kyrala, Phys. Rev. E 76 (2007) 056403-1-9.
[46] T. Nagayama, R.C. Mancini, R. Florido, R. Tommasini, J.A. Koch, J.A. Delettrez, S.P. Regan, V.A. Smalyuk, J. Appl. Phys. 109 (2011) 093303-1-10.
[47] T. Nagayama, R.C. Mancini, R. Florido, D. Mayes, R. Tommasini, J.A. Koch, J.A. Delettrez, S.P. Regan, V.A. Smalyuk, Phys. Plasmas 19 (2012) 082705-1-15.
[48] R.C. Mancini, H.M. Johns, T. Joshi, D. Mayes, T. Nagayama, S.C. Hsu, J.A. Baumgaertel, J. Cobble, N.S. Krasheninnikova, P.A. Bradley, P. Hakel, T.J. Murphy, M.J. Schmitt, R.C. Shah, I.L. Tregillis, F.J. Wysocki, Phys. Plasmas 21 (2014) 122704-1-7.
[49] T. Nagayama, R.C. Mancini, R. Florido, D. Mayes, R. Tommasini, J.A. Koch, J.A. Delettrez, S.P. Regan, V.A. Smalyuk, Phys. Plasmas 21 (2014) 050702-1-5.
[50] T. Nagayama, R.C. Mancini, D. Mayes, R. Tommasini, R. Florido, Rev. Sci. Instrum. 86 (2015) 113505-1-11.
[51] C.J. Fontes, M.E. Sherrill, J. Abdallah Jr., Modeling NLTE Atomic Kinetics of Mixed Materials, Los Alamos National Laboratory Memorandum, X-1:07-02(U), 2007.
[52] M.E. Sherrill, R.C. Mancini, J. Bailey, A. Filuk, B. Clark, P. Lake, J. Abdallah Jr., Phys. Rev. E 76 (2007) 056401-1-16.
[53] I. Golovkin, R. Mancini, S. Louis, Y. Ochi, K. Fujita, H. Nishimura, H. Shiraga, N. Miyanaga, H. Azechi, R. Butzbach, I. Uschmann, E. Förster, J. Delettrez, J. Koch, R.W. Lee, L. Klein, Phys. Rev. Lett. 88 (2002) 045002-1-4.
[54] L.A. Welser, R.C. Mancini, J.A. Koch, N. Izumi, S.J. Louis, I.E. Golovkin, T.W. Barbee Jr., S.W. Haan, J.A. Delettrez, F.J. Marshall, S.P. Regan, V.A. Smalyuk, D.A. Haynes Jr., R.W. Lee, J. Quant. Spectrosc. Radiat. Transfer 99 (2006) 649–657.
[55] T. Nagayama, R.C. Mancini, L.A. Welser, S. Louis, I.E. Golovkin, R. Tommasini, J.A. Koch, N. Izumi, J. Delettrez, F.J. Marshall, S.P. Regan, V. Smalyuk, D. Haynes, G. Kyrala, Rev. Sci. Instrum. 77 (2006) 10F525-1-3.
[56] G. Kagan, X.-Z. Tang, Phys. Plasmas 19 (2012) 082709-1-8.
[57] G. Kagan, X.-Z. Tang, Phys. Rev. Lett. 109 (2012) 269501-1-1.
[58] G. Kagan, X.-Z. Tang, Phys. Lett. A 378 (2014) 1531–1535.
[59] K. Molvig, A.N. Simakov, E.L. Vold, Phys. Plasmas 21 (2014) 092709-1-19.
[60] J. Langr, Modern C++ Programming with Test-Driven Development, Pragmatic Bookshelf, Dallas, Texas, Raleigh, North Carolina, 2013.
[61] Doxygen software documentation tool, www.doxygen.org.
[62] Yu. Ralchenko, A. Kramida, J. Reader, and the NIST ASD Team, NIST Atomic Spectra Database version 4.1, 2011, available at http://physics.nist.gov/asd (National Institute of Standards and Technology, Gaithersburg, MD).
[63] http://aphysics2.lanl.gov/cgi-bin/opacrun/tops.pl.