MULTI-IFE Technical Manual[1]

Rafael Ramis[2]
E. T. S. I. Aeronáutica y del Espacio, Universidad Politécnica de Madrid[3]

July 4, 2015

[1]file: [13_MULTI-fs]/doc/2015-MULTI-IFE.tex
[2]email: rafael.ramis@upm.es
[3]mail: E. T. S. I. Aeronáutica y del Espacio, P. Cardenal Cisneros 3, 28040 Madrid, Spain

Contents

1 Introduction **5**
- 1.1 MULTI-IFE - A One-Dimensional Computer Code for Inertial Fusion Energy (IFE) Target Simulations *R. Ramis and J. Meyer-ter-Vehn* . . . 5
- 1.2 Code overview . . . 5
- 1.3 Physical units . . . 6

2 Data structures **7**
- 2.1 `typesimstate` . . . 7
- 2.2 `typesimthermo` . . . 7
- 2.3 `typesimparam` . . . 8
- 2.4 `typesimlaserwkb` . . . 8
- 2.5 `typesimlasermaxwell` . . . 8
- 2.6 `typesimlaser3D` . . . 8
- 2.7 `typesimpulse` . . . 8
- 2.8 `typematerial` . . . 8
- 2.9 `eostable` . . . 9
- 2.10 `grouptable` . . . 9
- 2.11 `multitable` . . . 9
- 2.12 `typeenergy` . . . 9
- 2.13 `typesimlayers` . . . 9

3 Program structure **11**
- 3.1 Code organization . . . 11
- 3.2 Initialization block (`init`) . . . 11
- 3.3 Ending block (`finish`) . . . 13
- 3.4 Integration loop (`integrate`) . . . 14
- 3.5 Subroutine `ascii_output` . . . 14
- 3.6 Subroutine `write_output_record` . . . 14
- 3.7 Integration subroutine `step` . . . 16
- 3.8 Integration subroutine `schritt` . . . 16
- 3.9 Thermodynamic properties (`eosall`) . . . 18
- 3.10 Laser deposition (`laser`) . . . 18
- 3.11 Thermonuclear package (`fusion`) . . . 19
- 3.12 Electron heat transport and i-e energy exchange (`heat`) . . . 19
- 3.13 Ion heat diffusion (`heation`) . . . 19
- 3.14 Hydrodynamics (`hydro`) . . . 19
- 3.15 Radiative transport (`group`) . . . 19
- 3.16 Electron collision frequency (`collfreq`) . . . 19

4 Input file format **21**
4.1 namelist block `parameters` . . . 21
4.2 namelist block `material` . . . 23
4.3 namelist block `layer` . . . 23
4.4 namelist block `pulse_wkb` . . . 23
4.5 namelist block `pulse_maxwell` . . . 24
4.6 namelist block `pulse_3D` . . . 25
4.7 namelist block `pulse_shape` . . . 25
4.8 Example of input file . . . 25

5 Output file **29**
5.1 File structure . . . 29

6 Table file format **31**
6.1 Inverted equation of state tables . . . 31
6.2 Ionization tables . . . 32
6.3 Opacity/emissivity tables . . . 32

7 Installation **33**
7.1 Self standing mode . . . 33
7.2 Part of the MULTI environment . . . 34
7.2.1 r94 system . . . 34
7.2.2 CPC package . . . 34

8 Examples **37**
8.1 Au/Al foil submitted to a 300 ps pulse . . . 38
8.2 Al foil submitted to a 300 fs laser pulse . . . 39
8.3 Directly Driven IFE Target . . . 40
8.4 Indirectly Driven IFE Target . . . 41

Chapter 1

Introduction

1.1 MULTI-IFE - A One-Dimensional Computer Code for Inertial Fusion Energy (IFE) Target Simulations *R. Ramis and J. Meyer-ter-Vehn*

This paper (Ref. [1]) has been submitted to Computer Physics Communications as a Computer Programs in Physics Papers (CPiP) contribution. This sort of publication implies the incorporation of the code to the Comp. Phys. Comm. Program Library (Queen's University, Belfast). The present manual is included as part of the computer code and contains all technical and computational details that are out of the scope of a scientific paper. The **MULTI-IFE** code has been developed from the **MULTI-fs** code [2] (CPC catalog Id. **aekt_v1_0**), that in turn was developed from the **MULTI** code [3] (CPC catalog Id. **abbv_v1_0**). Although the physics, numerical algorithms, and computer implementation are drastically different, the present code inherits the spirit of these codes. In fact, most of the capabilities of **MULTI** and **MULTI-fs** are also available in **MULTI-IFE**.

1.2 Code overview

The physical background and algorithms are discussed in detail in the submitted paper. Briefly, the code **MULTI-IFE** includes:

1. One dimensional planar, cylindrical, or spherical geometry.
2. Multi-layer multi-material shells.
3. Laser deposition by either:
   (a) WKB aproximation with normal incidence and a fraction of energy coupled to the critical density.
   (b) Maxwell's equations solver (only for planar geometry).
   (c) WKB with 3D ray tracing (only for spherical geometry).
4. Implicit hydrodynamics with separate ion and electron temperatures.
5. Multigroup NON-LTE radiation transport.
6. Flux limited electron thermal diffusion.
7. Ion thermal diffusion.
8. Tabulated matter properties (EOS, opacities, emissivities, and ionization).

9. DT fusion package with non-local deposition (diffusion of α-particles).
10. Electron collisionality modelled by either:
    (a) Classical plasma model.
    (b) Electron-phonon interaction.
    (c) Drude-Sommerfeld model.

The **MULTI-IFE** code has been written with the idea that the users can modify and addapt the code to their specific applications. By this reason, the structure has been maintained as clean, logical, and compact as possible. The source code is completely contained in a unique FORTRAN 95/2003 file with about 4000 lines, that can be compiled by a single command. The source file contains numerous comments. A consistent naming of variables, derived types, and subroutines has been attempted. In addition to the source file, several files are included in the package:

1. Input and output files for the examples.
2. Tables with the material properties used in the examples.
3. Miscelaneous documentation, including this manual.
4. A graphic postprocessor (for Linux systems).

Details of installation are discussed in Chap. 7. The input file `fort.12` is composed by the "namelist" blocks described in Chap. 4. The format of files with material properties is explained in Chap. 6. The code generates a printout in "standard-output" intended as a diagnostic check (echo of all input data, description of read tables, summary of integration, and a final energy balance). The code produces also an ASCII file with important variables at specified times (`fort.11`) and a binary file with massive data (`fort.10`). This file, described in Chap. 5, is intended to be managed by graphic postprocessors.

1.3 Physical units

The CGS system of units (cm, g, s) is used for all variables, except temperature and thermal radiation frequency that are expressed in eV (1 eV = 10605 K = 2.4180×10^{14} s^{-1}). The definition of some magnitudes depend on the geometry. For example, "laser power" means laser intensity in planar geometry, power per unit of length in cylindrical geometry, and true power in spherical geometry.

Chapter 2

Data structures

The main variables used by **MULTI-IFE** are grouped in a few number of data structures. In FORTRAN language such structures are called "derived data types". This feature allows to manage as a sole unit a set of heterogeneous variables (including also other structures). Structures can be passed as subroutine arguments without needing to mention their individual components. The structure components are accessed by using the FORTRAN "component selector" operator (e.g., if `s` is an structure of type `typesimstate`, then `s%x` is an array with the coordinates of cell interfaces). The code is build around these structures as a set of subprograms that create, destroy, modify, or print structures. To understand the code, one must start looking to how these "derived data types" are defined. All of them are briefly described below. For more details, one should go to the source file.

2.1 `typesimstate`

This structures contain all the variables that define the current state of the simulation: the value of time (`time`), the coordinates (`x`) and velocities of interfaces (`v`), the mass density (`r`), the specific energies of electrons (`ee`) and ions (`ei`), the density of energy of α-particles[1] (`ea`), the molar fraction of tritium in thermonuclear fuel (`f`), and the coefficients (`wa`). The array of coeffcients `wa`, upgraded after each time step, contains the fractions of laser energy to be coupled to matter in each individual transport process. `x` and `v` are "interface defined" arrays of size $N+1$, `r`, `ee`, `ei`, `ea`, and `f` are "cell defined" arrays of size N, and `wa` is an array of size N_g+1, where N and N_g are the number of cells and radiation groups, respectively. Several structures of this type can exist simultaneously (e.g., one with the values at the begining of time step, other to store the final values, and a third one containing intermediate values). Because N and N_g are unknown at compilation time, all arrays are implemented as FORTRAN "pointers". Service suboutines allow to request memory for the data (`allocsimstate`), return the memory to the system (`freesimstate`), and copy data from one structure to other (`copysimstate`).

2.2 `typesimthermo`

The **MULTI-IFE** code uses as independent thermodynamic variables the mass density and the specific energies of electrons and ions. Pressures (`pe`, `pi`) and temperatures (`te`, `ti`) can be computed as function of these variables by interpolation in tables. The structures of type `typesimthermo` store these magnitudes in arrays of size N, as well as the ionization number (`zi`) and the thermodynamical partial derivatives (e.g., $(\partial P_e/\partial e_e)_\rho \equiv$`dpedee`). Service subroutines allow to request memory for the data (`allocsimthermo`), return the memory to the system (`freesimthermo`), and copy data from one structure to other (`copysimthermo`).

[1]Notice that `ee` and `ei` are energies per unit of mass while that `ea` is energy per unit of volume.

2.3 typesimparam

This structure contains all modeling parameters (e.g., `flf`, the flux limiter factor for electron heat flux), numerical parameters (e.g., `dtmax`, the maximum value for the integration time step), and constant quantities (e.g., `dm`, an array with the mass in each cell). Some of these quantities are taken directly from the input file, and others are computed by the program from input data (e.g., `n`, the total number of cells). The exact meaning of each input parameter is described in different sections across this manual. This structure contains other structures as components. Because only one "instance" is required, no specific memory allocation/deallocation subroutines are needed.

2.4 typesimlaserwkb

This structure, member of `typesimparam`, contains the parameters used by the traditional laser deposition package (e.g., laser wavelength) where WKB (Wentzel, Kramers, and Brillouin) propagation in planar, cylindrical, or spherical symmetries with normal incidence is assumed. Laser energy is absorbed by inverse bremsstrahlung and by a generic anomalous mechanism at critical surface. The component `enable=1` indicates that the package is active. If `enable=0`, the package is disabled and zero deposition is returned. The rest of parameters are listed in Chap. 4.

2.5 typesimlasermaxwell

Similar to `typesimlaserwkb` but for laser light treated by solving Maxwell's equations in planar geometry with oblique incidence and linear "p" or "s" polarization. Also here, the component `enable=1` indicates that the package is active. The rest of parameters are listed in Chap. 4.

2.6 typesimlaser3D

Similar to `typesimlaserwkb` but for laser light treated, in spherical geometry, as a discrete number of rays that are traced in 3D space taking into account refraction and inverse bremsstrahlung absorption. Also here, the component `enable=1` indicates that the package is active. The rest of parameters are listed in Chap. 4.

2.7 typesimpulse

This structure contains the time profile shape of the laser pulse (or thermal radiation pulse) as a table. For a given time, the incident power is obtained by interpolation. The details are given in Chap. 4.

2.8 typematerial

This type of structure contains all the properties of a given material: the material name, the atomic number (`zi`), the mass number (`ai`), and substructures with equation-of-states, ionization, opacities, and emissivities tables (see below). Also the name of the files containing the tables and the identifier codes (not more needed after loading the tables) are stored here. The `typesimparam` structure, defined above, contains **mat**, an array of `typematerial` structures, one for each defined material. The component **mid** of the `typesimparam` structure is an array of N numbers, indicating the material in each cell by its position in the array of `typematerial` structures.

2.9 `eostable`

Contains a single equation-of-state (electron or ion component). Two tables are required, with pressure and temperature of the component as functions of mass density and specific energy: $P(\rho_i, \Delta E_j)$ and $T(\rho_i, \Delta E_j)$. Where ρ_i indicates an array of density values, and ΔE_j an array of energies above the "cold energy". An one-dimensional table with the values of "cold energy" $E_0(\rho_i)$ is also part of this structure.

2.10 `grouptable`

This table contains a set of values $\chi(\rho_i, T_j)$ that depend on density and temperature. These are either:

- the decimal logarithm of Planck or Rosseland opacity (expressed in cm^2 g^{-1}).
- the decimal logarithm of normalized emissivity.
- the decimal logarithm of ionization (**In that case, exceptionally, the values of T_j are expressed in keV instead eV!**)

If the opacity or emissivity corresponds to a given frequency interval, the upper and lower values (`fmax` and `fmin`, expressed in eV) are included in the table. For full spectrum averaged opacities and for ionization tables, both values are zero.

2.11 `multitable`

Contains an array of `grouptable` elements with either Planck, Rosseland, or emissivity tables. In principle, three `multitable` structures are required for each material. Nevertheless, in the case of emissivities, if the array is empty, thermodynamic quasi-equilibrium is assumed.

2.12 `typeenergy`

This structure contain not essential data generated during the simulation, like:

- administrative data: number of time steps (successfull and total).
- instantaneous specific laser energy deposition $d(x)$ and radiation field $I(\nu, x)$.
- other instantaneous magnitudes: reflectivity and transmision, fusion power, radiative input/output powers, specific internal and kinetic energies, ...
- time integrated quantities, radiation input/output energies, number of neutrons, ...
- initial values of energies.

2.13 `typesimlayers`

This auxiliar structure is used temporaly to store the definition of multi-layered targets. After initialization is not used anymore.

Chapter 3

Program structure

3.1 Code organization

The source code resides in a single file (around 4356 lines). The bulk of the program is organized as a FORTRAN "module" called `multidata` containing three parts:

1. Physical constants (taken from [6]).

2. Definition of all data structures described in Chap. 2

3. Program units: subroutines and functions.

The main program (**multi_ife**), placed after that "module", contains only one executable sentence, a call to the subroutine "main", the true entry point of the code. The top level flow structure is schematized in Fig. 3.1. The subroutine `main` uses only 3 variables, that are structures of types `typesimparam`, `typesimstate`, `typesimenergy`, respectively. This subroutine calls three subroutines `init`, `integrate` and `finish`, that are the entry points of three blocks of subroutines. The first block initializes everything, the second one carries out all numerical job, and the last one writes a short summary. These blocks include numerous calls to a block containing output subroutines (`ascii_output`, `write_binary_record`, and the ones beginning with `print_`).

3.2 Initialization block (`init`)

The input file `fort.12` contains control input data as a set of FORTRAN "namelist" blocks (described in Chap. 4). Blocks with repeated name are used to define different layers and different materials. Because some compilers require a different name for each block, the code uses an indirect access to this file. Subroutine `namelistblock` copies an individual block from file `fort.12` to file `block`, from where it can be read. The following subroutines are called by `init`:

1. `read_parameters` - initializes most of scalar values in structure `typesimparam` from data in namelist block "`parameters`".

2. `read_laser_3d`, `read_laser_wkb`, and `read_laser_maxwell` - these subroutines read optional namelist blocks "`pulse_3d`", "`pulse_wkb`", and "`pulse_maxwell`" to initialize substructures with the laser definitions. Only one block can be present, if none is present, the laser deposition is set to zero.

3. `read_pulse` - this subroutine reads the optional namelist block "`pulse_shape`" with a generic table. It is only needed if option "`itype=4`" has been selected in one of the laser definition blocks or in the radiation boundary conditions.

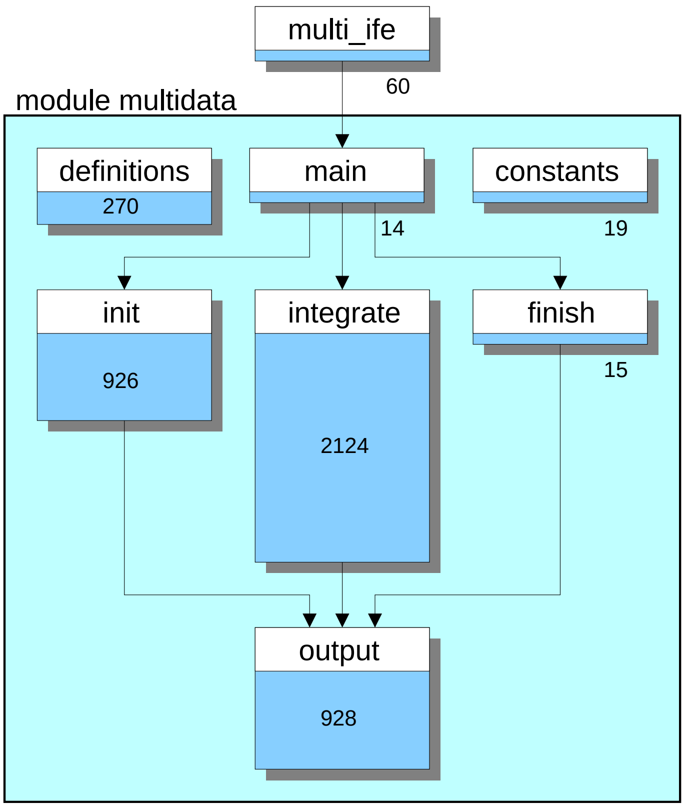


Figure 3.1: **MULTI-IFE** program structure. The numbers are the approximate number of source code lines (including comments) of each component.

4. `read_material_definitions` - this subroutine reads definitions of materials in namelist blocks “`material`”. Each material has a name, and some physical properties defined by tables contained in external files. Each table is defined by the file name (by default `fort.13`) and by an identifier (by default 0, that means: first table in the file). The internal format of these files is described in Chap. 6. Each material is stored as an element of array `mat` in the structure of type `typesimparameters`.

5. `read_material_data` - this subroutine loads the material tables from external files. An error occurs if a file is not accessible or it is bad formatted. This subroutine calls low level subroutines named `load_table_*`.

6. `read_layer_definitions` - this subroutine reads layer definitions from namelist blocks “`layer`” into a (temporal) structure of type `typesimlayers`.

7. `init_state` - using the layer description and the problem parameters, the state vector is initialized. First, the subroutine `frequencies` computes the number of radiation groups and their frequencies from the opacity tables of all materials. The structures (or elements) that depend on the number of cells or the number of radiation groups (`typesimstate`, and `typesimparam`) are now allocated. In second place, the values in the state vector are initialized. This requires to call subroutine `eosenergy` to get specific internal energies as function of initial temperatures and densities.

8. `update_energies` allocates a structure of type `typesimenergy` and initializes the balance of energy.

9. Once all variables have been initialized, a battery of output subroutines (named `print_*`) are called. The analysis of their output can be used to detect anomalies (e.g., after a modification of the code) or mistakes (e.g., missing or wrong input parameters).

3.3 Ending block (`finish`)

Once simulation finishes, the subroutine `print_energies` prints the relevant quantities available in structure `typesimenergy`, as indicated in table 3.1.

Name	Description
	Number of integration time steps
`istep`	Valid steps
`itry`	Total (some of them failed)
	Energy balance
`enie0`	Internal energy of electrons (initial)
`enii0`	Internal energy of ions (initial)
`enia0`	Energy in α-particles (initial)
`enki0`	Kinetic energy (initial)
`delas`	Absorbed laser energy
`derad`	Absorbed thermal radiation (can be negative)
`defus`	Energy of fusion coupled into the simulation (see note A)
`enie`	Internal energy of electrons (final)
`enii`	Internal energy of ions (final)
`enia`	Energy in α-particles (final)
`enki`	Kinetic energy (final)
`error`	Numerical error on energy balance (see note B)
	Detail of radiative coupling
`radoutl`	Radiative losses thought the left boundary
`radinl`	Incident radiation thought the left boundary
`radoutr`	Radiative losses thought the right boundary
`radinr`	Incident radiation thought the right boundary
	Other quantities
`laser`	Incident laser energy
`fusion`	Released fusion energy (see note A)
`neutrons`	Number of fusion neutrons (see note A)
`alphas`	Energy lost by escaped α-particles (see note A)

Table 3.1: Values displayed by subroutine `print_energies`.

Note A

In each fusion reaction ($D + T \to n + \alpha$), a neutron and an α-particle are generated. In the algorithm implemented in the code, the neutron escapes freely from the numerical grid. Part of the energy on α-particles, 1/5 of the fusion energy (`fusion`), is coupled into the simulation (`defus`) and the rest is lost by escaping α-particles (`alphas`).

Note B

The initial energy (`enie0`+`enii0`+`enrae0`+ `enki0`) plus the deposited energy (`delas`+`defus`+`derad`) should be equal to the final energy (`enie`+`enii`+`enrae`+ `enki`). The sum of energies (with appropriate sign) must be zero. The sum of positive and negative values, e^+ and e^-, are evaluated, and the relative error in energy is defined as

$$\texttt{error} = 2\frac{|e^+ - e^-|}{e^+ + e^-}$$

Because the numerical algorithms preserve energy, the inbalance is due only to arithmetic round off.

3.4 Integration loop (`integrate`)

The initial state defined by a structure of type `typesimstate` is advanced in time by subroutine `step` to generate a new state (see Fig. 3.2). Also the secondary data in `typesimenergy` and the next time step `dt` are upgraded by the subroutine `step`. The process is repeated until either the exit time or the maximum number of steps are reached. At specified time intervals or given number of time steps, a binary record (into file `fort.10`) or an ascii printout (into file `fort.11`) are generated by subroutines `write_output_record` and `ascii_output`, respectively.

3.5 Subroutine `ascii_output`

This subroutine prints the main variables at a given time. The structure of this subroutine has been maintained rather simple, so that users can customize the output to their needs. By defaut the output is written into file `fort.11`.

3.6 Subroutine `write_output_record`

This subroutine receives as arguments the set of simulation parameters, the state vector, and the auxiliar quantities. First, a set of additional quantities are computed by auxiliar subroutines (`compute_*`):

1. Number of each cell (`nc`) and interface (`ni`).
2. Mass coordinate at cell centers (`cmc`) and interfaces (`cmi`).
3. Coordinates of cell centers (`xc`) and mean frequency of radiation groups (`frec`).
4. Electron number-density (`dens`).
5. Total pressure (electrons, ions, and alpha particles) (`pt`).
6. Specific laser deposition (power per unit of mass) (`ad`).
7. Area associated with each interface (`a`).
8. Radiation fluences (fluxes times area at interfaces): total (`s`), for photons going from left to right (`sp`), and for photons going from right to left (`sm`).
9. Radiation temperature at interfaces (`tr`).
10. Radiation spectra: energy fluence in negative and positive directions at requested interfaces (`irad_left` and `irad_right`) per unit of frequency (`radl` and `radr`).
11. Average implosion velocity (`vimplo`) of thermonuclear fuel (specified by input parameter `nfuel`). Only negative velocities are considered.
12. Confinement parameter ($\int \rho dr$) for the whole thermonuclear fuel (`rhorDT`) and for the part of fuel where ion temperature is above 5 keV (`rhorHS`).

13. In-flight-aspect-ratio (`ifar`), defined as $\frac{1}{2}(r_e + r_i)/(r_e - r_i)$, where r_i and r_e the first radius to the left/right of density peak where the density is a factor e lower than the peak value.

14. Entropy parameter `alpha`, defined as the mass averaged value of the ratio between ion plus electron pressure and Fermi's pressure, averaged for fuel cells with a density larger that 10% of the maximum fuel density.

Once all these quantities have been computed, a "binary" record is written in file `fort.10`. That file is composed of two "sections": the first one with static data (for example, cell masses), and the second one with dynamic data (changing with time). Each "section" is composed of several "records" containing several "words". In the current implementation each "word" is an ASCII formatted line containing either a text string or a formatted number with edit descriptor `i6` or `e12.5`. The structure of one "section" is:

- First "record": an integer number indicating the number of items in the rest of "records" of this "section".
- Second "record": the names of the variables, each with 8 characters (including blanks). For arrays, the name is repeated as many time as elements in the array.
- Third "record": a zero for scalar variables and an integer number for array elements.
- Fourth "record": values of variables given in the same order as in the previous "records".
- Additional "records": in the case of dynamic quantities (i.e., in the second "section"), values of the variables at different times.

When the subroutine `write_output_record` is called by first time, the whole first "section" is written (static data), as well as the heading of the second section (three "records") and the first data "record" of the second "section". In successive calls, only a data "record" is written. To facilitate the creation of such a file, output is done through auxiliar subroutines. The subroutine `write_vector(mode,icount,nr,label,n,value)` is called for each array to be written. The argument `value` is an one-dimensional array of size `n`. The first argument (`mode`) can take four values:

0 - Does not write anything. The second argument (`icount`) is incremented by the number of elements to be written.

1 - Writes the name of the variable (`label`) as many times as elements to be written (see below).

2 - Writes succesive integer numbers. As many as elements to be written.

3 - Writes data. This processs is controlled by argument `nr`.

  - `nr`$= 0$: write all elements in `value`
  - `nr`> 0 : `value` is assumed to be a cell centered magnitude. Write the averages over `nr` cells. (`n`+`nr`-1)/`nr` values are written.
  - `nr`< 0 : `value` is assumed to be a interface centered magnitude. Write one value each `|nr|` interfaces. (`n`+`|nr|`-2)/`|nr|`+1 values are written.

The simplified version, subroutine `write_item` is used for scalars.

3.7 Integration subroutine `step`

The subroutine `schritt` is called to produce a new state vector after a requested time interval `dt` (See Fig. 3.2-b). The subroutine returns also a scalar value `var` that represents the maximum variation of selected quantities (e.g. density, temperature,. . .) relative to specified variations (e.g., the parameter `dtrvar` specifies a nominal relative variation of density). Ideally, the value of `var` should be close to unity. If `var` exceeds the value specified by parameter `dtbreak` ($\simeq 2$), or if the subroutine `schritt` aborts due to negative density, the integration steps is considered invalid and it is tried again with halved `dt`. The process is repeated until success (`var<dtbreak`) occurs or the minimum value for the time step (parameter `dtmin`) is reached. Afterwards, the subroutine

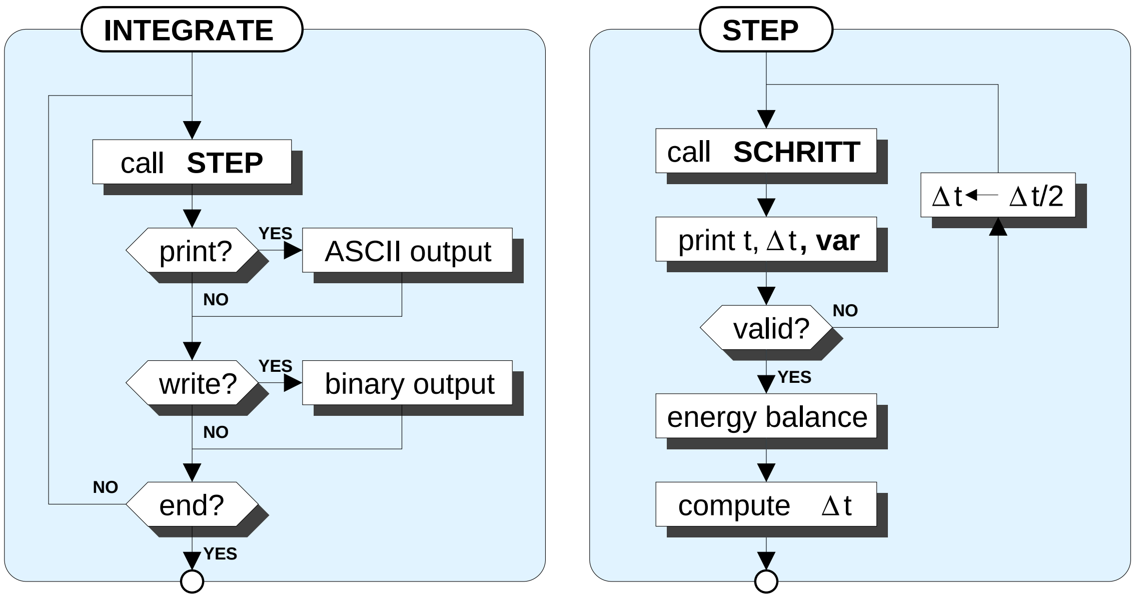


Figure 3.2: Integration subroutines `integrate` and `step`.

`update_energy` is called to integrate powers, fluences, and spacial distributions into total energies. Finally, based on the current value of `dt` and `var`, a new value of `dt` is evaluated as

$$\mathtt{dt} \leftarrow \frac{\mathtt{dt}}{\mathtt{var}}$$

with the idea to have `var`$\simeq$1 in the next time step. Provisions are taken to avoid that the relative increment of time step exceeds the value specified by parameter `dtfactor` and to force that time step is between `dtmin` and `dtmax`.

3.8 Integration subroutine `schritt`

This subroutine performs (or tries to perform) a time step. It uses a fractional step method (also known as "time-splitting"). Physical processes are advanced separately, one by one. For example, heat transport is solved in a static configuration, afterwards the hydrodynamic motion is advanced without heat diffusion, and so on. The subroutine `schritt` calls subroutines that change the state vector by only one physical process. Ideally, the structure would be something like:

- Deposit laser energy. Only the internal energy of electrons (`ee`) is modified.
- Advance thermonuclear reactions. Only the fraction of tritium (`f`) and the density of α-particles (`ea`) are modified.

- Diffuse α-particles and energy interchange. Only the internal energies of electrons, ions, and α-particles (`ee`, `ei`, and `ea`) are changed.
- Electron heat diffusion. Only `ee` changes.
- Ion heat diffusion. Only `ei` changes.
- Ion-electron relaxation. Both `ee` and `ei` change.
- Hydrodynamics. All variables modified, except the fraction of DT.
- Radiation transport by photons in first frequency group. Only `ee` changes.
- Radiation transport by photons in second frequency group. Only `ee` changes.

  ...
- Radiation transport by photons in $\texttt{ng}^{th}$ frequency group. Only `ee` changes.

The actual structure, shown in Fig. 3.3, is a bit more complicated. The number of radiation groups `ng` can be large (increases with spectral resolution). With the above scheme, most of the CPU time would be devoted to radiation transport. When a radiation group is solved, the variation of `ee` is small is comparison with variations due to electron heat transport or other processes. Integration error is thus dominated by the non-radiative processes. To balance this situation, all processes, except radiation, are split in `msplit` parts (1 $\leq$`msplit`$\leq$`ng`), and each part solved with time step `dtf`=`dt/msplit`. These parts are distributed more or less uniformly between radiation groups. This strategy is called "subcycling". By the other hand, laser deposition can produce strong temperature oscilations during the time step. For example, laser can be almost absorbed in one cell with small mass, producing a sharp temperature peak. Later, this peak will be diffused by heat/radiation transport and a smooth temperature profile recovered. To avoid this phenomena, laser deposition is not applied inmediately, but split in fractions (with weight factors `wa(1:ng+1)`) that are injected into radiation and heat conduction processes. These coeficients are recomputed by subroutine `wctrl` as function of temperature variations during the subprocesses (`wb(1:ng+1)`) with the aim of reduce the oscilations in the next time step. As `wa` are required by the integration algorithm, they are considered part of the state vector. More details of this feedback mechanism are given in [1, 2, 3]. Temperature and pressure are computed as function of density and internal energy. To avoid excesive CPU use by the interpolation subroutines, these values are computed only `msplit` times during the time-step, and linealization is assumed between evaluations. The actual algorithm is as follows (See also Fig. 3.3):

1. Allocate temporal structures and initialize variables. State vector is copied from `s1` (preserved initial value) to `s2` (value at the begining of the subcycle/final value).
2. Start point of `msplit` loops (cycles).
3. Routine `eosall` evaluates the thermodynamic state `t2`: temperatures, pressures, their thermodynamic derivatives, and ionization. Ionization is assumed constant during the loop, while temperature and pressure are assumed to be linear functions of density and specific internal energy.
4. The state at the beginning of this loop (`s2`, `t2`) is copied into working storage (`s3`, `t3`). These values will be advanced by subprocess subroutines.
5. The laser deposition `d` (power per unit of mass) is computed by subroutine `laser`.
6. The subroutine `fusion` advances thermonuclear reactions. The molar fraction of tritium (`s3.f`), initially $\frac{1}{2}$ in DT fuel, is decremented by the nuclear reactions. The produced α-particles feed the corresponding density of energy (`s3%ea`). The same subroutine computes the difussion of `s3%ea`, giving place to a power deposition (per unit of volumen) into ion (`dai`) and electron (`dae`) species. A fraction of α-particle power (`e%alpha1`) is lost through the external boundary.

7. The specific powers to be coupled to ions `qi` (nuclear reactions) and to electrons `qe` (nuclear reaction plus a fraction of the laser) are computed.
8. The subroutine `heat` solves together electron heat transport and ion-electron relaxation with external depositions `qe` and `qi`. The specific energies (`s3%ee`,`s3%ei`) and temperatures (`t3%te`,`t3%ti`) are advanced.
9. If requested, the subroutine `hydro` advances hydrodynamics using a conservative Lagragian algorithm. Interface position and velocity, energies of ion, electrons, and α-particles are advanced.
10. If requested, the subroutine `heation` advances heat diffusion by ion transport. Only `s3%ei` and `t3%ti` are affected.
11. If requested, radiation transport for a subset of frequency groups is advanced by calls to subroutine `group`. A fraction of the specific laser deposition is coupled to each group. Only `s3%ee` and `t3%te` are modified.
12. The maximum relative variation of density, ion temperature, and electron temperature during the loop are computed. In the case of temperature, the reference temperature is the highest between the cell temperature and the temperatures of its two neighbors. These variations are normalized by user specified variations and the maximum value `var` is returned to calling subroutine, where it will be used to control the time step.
13. The current state (`s3`) is copied to (`s2`) and a new loop started (point 2).
14. During the process, temperature variations at the place where laser deposition is maximum (`wb`) have been monitorized for electron heat transport and radiation transport subprocesses. The subroutine `wctrl` recomputes, from `wb`, the coefficients `wa` to be used in the next time step.

3.9 Thermodynamic properties (`eosall`)

The subroutine `eosall` locates groups of adjacent cells with the same material and call interpolation subroutines `eos`, for electron and ion equations-of-state, and `zbr`, for ionization. These subroutines call low level ones `eoslin` and `eosbin`. If, for the given density, the specific energy is below the "cold" energy (i.e., negative temperature) the "cold" energy is taken. If the ionization table has been not loaded, the ionization is taken equal to the atomic number of the material. The ionization is forced to be above a specified minimum (`p%zmin`).

3.10 Laser deposition (`laser`)

The subroutine `laser` returns the laser deposition into cells as power per unit of mass. This subroutine calls one of the following subroutines:

1. `laser_wkb` - normal incidence rays in the WKB (Wentzel, Kramers, and Brillouin) approximation in planar, cylindrical, or spherical symetries. Physical basis and numerical algorithm are described in [2, 3].
2. `laser_maxwell` - resolution of Maxwell's equations in planar geometry with oblique incidence and "p" or "s" polarization. Physical basis and numerical algorithm are described in [2, 4].
3. `laser_3d` - 3D ray tracing in spherical geometry in the WKB approximation. Physical basis and numerical algorithm are described in [1].

The subroutine `pulse` is called by these subroutines to provide the laser intensity at a given time. Only one of the above options can be active at the same time. If none is active, zero deposition is returned.

3.11 Thermonuclear package (fusion)

Implements reaction $D_2 + T_3 \rightarrow He_4 + n$. Reactivity and α-particle diffusion algorithm are described in detail in [1]. The subroutine `tridia` is called to solve a tridiagonal system of linear equations.

3.12 Electron heat transport and i-e energy exchange (heat)

The physical model and the numerical discretization are described in [1, 2]. The conductivity and relaxation coefficents are determined in terms of the electron collision frequency computed by subroutine `collfreq`. The implicit equations for electron and ion temperatures are reduced to a tridiagonal system of linear equations that is solved by subroutine `tridia`.

3.13 Ion heat diffusion (heation)

This subroutine is similar to `heat`, but simpler. Usually, in a plasma, ion heat flux is negligible in comparison with electron heat flux (for the same temperature gradient, the quotient between both fluxes is basically equal to the square root of the quotient between electron to ion masses). Nevertheless, the ion thermal flux is crucial in IFE targets to treat the singularity of ion temperature that occurs in spherically convergent shock waves. Because this occurs in a very hot plasma, classical plasma collision frequencies are used.

3.14 Hydrodynamics (hydro)

An implicit Lagrangian conservative algorithm has been implemented, with separate energy equations for electrons, ions, and α-particles. The mechanical work dissipated by the artificial viscous pressure is coupled to ions. The equationes are reduced to a tridiagonal system of linear equations and solved by subroutine `tridia`. The current implementation is a particularization of the method analyzed in detail in [5].

3.15 Radiative transport (group)

The subroutine `group` solves the radiation transport by photons inside a given spectral interval. The algorithm is described in [2]. This routine calls routine `opacit` to obtain frequency averaged opacities (Planck and Rosseland averages) and normalized emissivities. A Planckian external source can be prescribed. The corresponding time dependent temperature is supplied by subroutine `pulse`. Also subroutine `tridia` is used here.

3.16 Electron collision frequency (collfreq)

The subroutine `collfreq` computes the collision frequency between electrons and ions, needed for laser absorption, electron–ion energy transfer, and electronic heat conduction. For high temperature, weakly coupled, fully ionized plasmas, the classical theory is applied. For warm dense matter (WDM) and metal-like solids, two models are implemented: the first one based in electron–phonon interaction, the second one within the Drude–Sommerfeld model. For high temperature, both models approach to the classical theory. A detailed description is given in [2]. Subroutines `fusion` and `heation`, that manage physical processes that are relevant only at high temperature, use their own evaluation of the classical collision frequency.

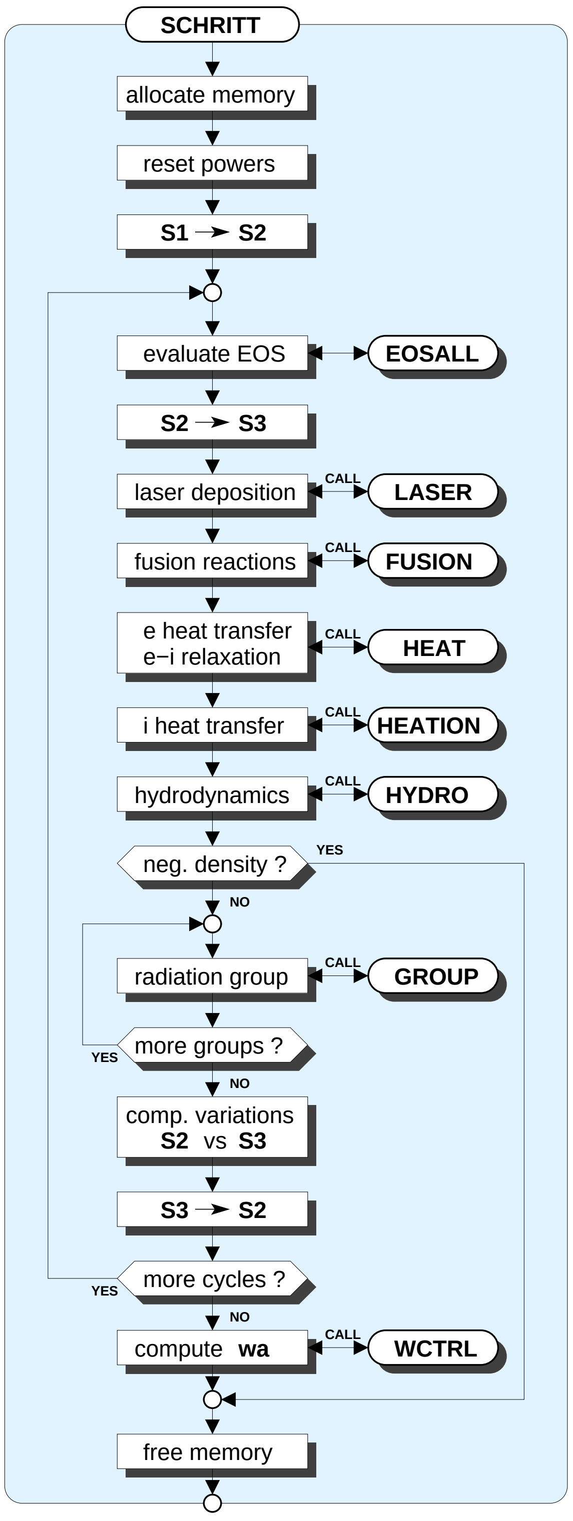


Figure 3.3: Integration subroutine `schritt`.

Chapter 4

Input file format

The **MULTI-IFE** program is controlled by input data file `fort.12`, that contains a set of FORTRAN namelist blocks. Some of them are mandatory, other are optional, and some are mutually incompatible (e.g., `pulse_wkb`, `pulse_maxwell`, and `pulse_3d`). Namelist blocks, except `layer` ones, can be appear in arbitrary order in the input file. Comments (everything in a line after exclamation (!) character) are allowed. Blocks not referenced by the code can be present in the input file. Such dummy blocks are usefull to debug and document applications. For example, to switch out provisionally laser absorption, just rename "laser_wbk" to a dummy name.

WARNING

No default values are supplied for variables (except in `material` blocks). So that special care should be taken to write the input file. Not defined items can take arbitrary values at execution time!

4.1 namelist block `parameters`

This mandatory block contains the modelling, numerical, and output options. The parameters marked by (F) can take values 1 (the statement is true) or 0 (it is false).

	problem definition (general)
`igeo`	1, 2, 3, for planar, cylidrical, and spherical geometries, respectively
`xmin`	initial coordinate of first interface
`texit`	end time of the interval to be simulated
`nfuel`	the leftmost `nfuel` cells contain DT fuel
	problem definition (boundary conditions)
`ileft`	(F) first interface is a rigid boundary
`iright`	(F) last interface is a rigid boundary
`alphal`	fraction of radiation reflected at the first interface
`alphar`	fraction of radiation reflected at the last interface
`betal`	radiacion pulse fraction coupled on the first interface
`betar`	radiacion pulse fraction coupled on the last interface

Table 4.1: Namelist block `parameters`

`trad`	radiation pulse temperature T_R
`tau`	radiation pulse duration τ
`itype`	radiacion pulse shape:
	1: $T_R \sin^2(\pi t/2\tau)$, for $0 < t < 2\tau$
	2: T_R, for $0 < t < \tau$
	4: use table defined by `'pulse_shape'` namelist block
	problem definition (modeling options)
`iradia`	(F) enable radiation
`ihydro`	(F) enable hydrodynamics
`iheation`	(F) enable ion heat conduction
`inegpre`	(F) allow negative pressure
`model`	electron collision model
	0: classical
	1: electron–phonon
	2: Drude–Sommerfeld
`fheat`	coefficient (for model 1) for heat transport
`fei`	coefficient (for model 1) for ion-electron relaxation
`flaser`	coefficient (for model 1) for laser absorption
`zmin`	enforced minimum value of ionization
`flf`	flux limit factor for electron heat transport
	numerical options
`nsplit`	number of subcycles (between 1 and the number of groups)
`dtmin`	minimum time step
`dtmax`	maximum time step
`dtinit`	initial time step
`dtrvar`	nominal relative variation of density in one subcycle
`dttevar`	nominal relative variation of electron temperature in one subcycle
`dttivar`	nominal relative variation of ion temperature one subcycle
`dtbreak`	tolerance factor to validate an integration step
`dtfactor`	maximum relative increments of time step
`nexit`	maximum number of integration steps
`iwctrl`	distribution of laser energy between subprocesses
	0: equal parts
	1: adjusted dynamically
	output control
`dt_aout`	write ASCII output each `dt_aout` time interval
`ns_aout`	write ASCII output each `ns_aout` integration steps
`dt_bout`	write binary output each `dt_bout` time interval
`ns_bout`	write binary output each `ns_bout` integration steps
`irad_left`	spectra is computed at this interface for photons with negative angle
`irad_right`	spectra is computed at this interface for photons with positive angle
`nreduce`	Reduced output. If 0 or 1, write all values, otherwise write:
	one value each `nreduce` interfaces (for interface values)
	averaged values over `nreduce` cells (for cell values)

Table 4.2: Namelist block `parameters` (continuation)

4.2 namelist block `material`

One `material` block is needed for each material. Although not forbidden, blocks for unused materials should be avoided: tables are loaded into memory and can cause collateral effects (all loaded tables contribute to the definition of frequency intervals for radiation). To disable temporarily a material, change “`material`” to a dummy name. The blocks make reference to material tables by giving the file name and an identifier number. If this number is zero, the first table in the file is taken. By default the file name is `fort.13` (the file used in **MULTI-fs** code [2]) and the identifier is cero. If the identifier is set to a negative number, the table is not loaded. This is allowed only for ionization (the atomic number is used) or for emissivity (LTE is assumed). The user has two basic alternatives: put all tables in a few files (or a unique file) and use identifier codes, or put each table in a file and ignore identifiers. The parameters marked by (T) are character strings (up to 80 characters).

`name`	(T) material name as appears in `'layer'` blocks
`zi`	atomic number
`ai`	mass number
`eeos_file`	(T) file containing electron EOS table
`eeos_id`	identifier number of the electron EOS table
`ieos_file`	(T) file containing ion EOS table
`ieos_id`	identifier number of the ion EOS table
`z_file`	(T) file containing ionization table
`z_id`	identifier number of the ionization table
	(if `z_id`=-1, full ionization is assumed)
`planck_file`	(T) file containing Planck opacity table
`planck_id`	identifier number of the Planck opacity table
`ross_file`	(T) file containing Rosseland opacity table
`ross_id`	identifier number of the Rosseland opacity table
`eps_file`	(T) file containing normalized opacity table
`eps_id`	identifier number of the normalized opacity table
	(if `eps_id`=-1, local thermal equilibrium is assumed)

Table 4.3: Namelist block `material`

4.3 namelist block `layer`

The target initial configuration is defined by a stack of layers, each defined by a `layer` block. The target is build from left to right, piling layers in the same order as they appear in the input file. The material name (T) is a character string (up to 80 characters).

4.4 namelist block `pulse_wkb`

This block defines a laser beam with normal incidence laser, in the WKB approximation. The parameter `pimax` (in CGS units) is the intensity in planar geometry (1 W cm^{-2} = 10^7 g s^{-3}), the power per unit of length in cylindrical geometry, or the total power in spherical geometry. The pulse time shape is defined by parameter `itype`. For `itype`=1 or 2, simple shapes are assumed. In both cases, `pimax` is the peak intensity and `pitime` is the FWHM. For `itype`=4, a tabulated profile is used (see Sec. 4.7). In that case, `pimax` and `pitime` define the units used in the table. With normal incidence, the laser light propagates until the critical surface, where it can be completely reflected (`delta`=0), completely absorbed (`delta`=1), or partially absorbed (0<`delta`< 1).

`thick`	thickness
`material`	(T) name of the material, as it appears in `'material'` blocks
`nc`	number of cells
`r0`	initial density
`te0`	initial electron temperature
`ti0`	initial ion temperature
`zonpar`	gridding parameter. Cell thicknesses of two succesive cells are related by:
	if `zonpar`> 0,
	$\Delta x_{i+1} =$`zonpar`$\times \Delta x_i$
	if `zonpar`< 0,
	$\Delta x_{i+1} = \vert$`zonpar`$\vert \times \Delta x_i$ in the first half of grid and
	$\Delta x_{i+1} = \vert$`zonpar`$\vert^{-1} \times \Delta x_i$ in the second half

Table 4.4: Namelist block `layer`

`inter`	laser incidence: 1 from RHS, -1 from LHS
`wl`	laser wavelength
`pimax`	laser intensity I (planar geometry)
	laser power per unit of length I (cylindrical geometry)
	laser power I (spherical geometry)
`pitime`	pulse duration τ
`itype`	pulse shape:
	1 - $I \sin^2(\pi t/2\tau)$, for $0 < t < 2\tau$
	2 - I, for $0 < t < \tau$
	4 - use table defined by `'pulse_shape'` namelist block
`delta`	fraction absorbed at critical density

Table 4.5: Namelist block `pulse_wkb`

4.5 namelist block `pulse_maxwell`

This block defines a laser treated as an EM wave with oblique incidence. This option is only available in planar geometry. The intensity is defined in the direction of laser propagation. The power per unit of target area is thus `pimax`$\times$ cos(`angle`). The solver subroutine (`laser_maxwell`) uses a subgrid where each cell is divided in `idep` parts. The polarization `pol` is given as a one character string.

`inter`	laser incidence: 1 from RHS, -1 from LHS
`wl`	laser wavelength
`pimax`	laser intensity I (relative to propagation direction)
`pitime`	pulse duration τ
`itype`	pulse shape:
	1 - $I \sin^2(\pi t/2\tau)$, for $0 < t < 2\tau$
	2 - I, for $0 < t < \tau$
	4 - use table defined by `'pulse_shape'` namelist block
`idep`	internally each cell is divided in `'idep'` subcells
`angle`	incidence angle (in degrees)
`pol`	polarization: either 'p' or 's' (one character string)

Table 4.6: Namelist block `pulse_maxwell`

4.6 namelist block `pulse_3D`

Defines a laser beam with finite width, incident over a spherical target. The beam is divided in `nr` rays with impact parameter between 0 and `rmax`, with a super-Gaussian intensity distribution defined by diameter `fwhm` and by exponent `bexp` (`bexp`=2 for a Gaussian profile). Each ray is solved in the WKB approximation taking into account light refraction.

`wl`	laser wavelength
`pimax`	laser power
`pitime`	pulse duration
`itype`	pulse shape: 1 - $I\sin^2(\pi t/2\tau)$, for $0<t<2\tau$ 2 - I, for $0<t<\tau$ 4 - use table defined by `'pulse_shape'` namelist block
`nr`	number of rays
`rmax`	maximum radius
`fwhm`	beam diameter
`bexp`	super-Gaussian exponent

Table 4.7: Namelist block `pulse_3d`

4.7 namelist block `pulse_shape`

Defines the pulse shape, for laser or for external radiation, as a $\{x_i, y_i\}$ table (maximum size 100). This block defines only the shape, the absolute values of time and power/temperature are determined by multipliing tabulated values by the reference values given in `parameters` or `pulse_*` block. Three interpolation methods have been implemented. The items marked by (V) are vectorial quantities, a set of numbers separated by commas.

`ntab`	number of values in `ttab` and `ptab`
`mtab`	interpolation mode: 1 - linear interpolation in `ptab`^{expo} 2 - linear interpolation in log(`ptab`) 3 - linear interpolation in exp(`ptab`)
`expo`	used only if `mtab`=1
`ttab`	(V) time divided by τ
`ptab`	(V) radiation temperature, power, or intensity divided by T_R or I

Table 4.8: Namelist block `pulse_shape`

4.8 Example of input file

The remaining details of the format of the input file are illustrated by the following input file, corresponding to last case in Chap. 8.

```
!---------------------------------------------------------------------------+
!  Rafael Ramis                                                             !
!  Universidad Politecnica de Madrid                                        !
!  March 4, 2015                                                            !
!---------------------------------------------------------------------------+
!  NIF target with Be ablator                                               !
!  configuration taken from J. Lindl, Phys. Plasmas 2 (11) 3933 (1995)      !
!  geometry described in Fig. 110 (pag. 4010)                              !
!  radiation pulse similar to the one from Fig. 111 (pag. 4010)            !
!  (line marked ">10 MJ")                                                  !
!  Pulse rises from 80 eV to 250 eV from 3.3 ns to 14.2 ns  (instead 6/14 ns) !
!--------------------------------------------------------------------------+
&parameters
!--problem definition (general)
   igeo       = 3,                         ! geometry (spherical)
   xmin       = 0.0,                       ! initial left boundary coordinate
   texit      = 20e-9,                     ! exit time
   nfuel      = 100,                       ! cells with DT
!--problem definition (boundary conditions):
   iright     = 0,                         ! free right boundary
   ileft      = 1,                         ! rigid left boundary (center)
   alphal     = 1.,                        ! reflected rad. frac. at L
   alphar     = 0.,                        ! reflected rad. frac. at R
   betal      = 0.,                        ! rad. pulse coupling at L
   betar      = 1.,                        ! rad. pulse coupling at R
   itype      = 4,                         ! rad. pulse shape (tabulated)
   tau        = 1e-9,                      ! rad. pulse duration scale (1 ns)
   trad       = 1.,                        ! rad. pulse temp. scale (1 eV)
!--problem definition (modeling options)
   iradia     = 1,                         ! enable radiation
   ihydro     = 1,                         ! enable hydrodynamics
   iheation   = 1,                         ! enable ion heat conduction
   model      = 0,                         ! electron collisions model (class.)
   fheat      = 8.0,                       ! coefficient (if model==1)
   fei        = 2.0,                       ! coefficient (if model==1)
   flaser     = 26.0,                      ! coefficient (if model==1)
   zmin       = 0.1,                       ! minimum value of Z
   flf        = 0.10,                      ! flux limit factor
   inegpre    = 1,                         ! allow negative pressure
!--numerical options
   nsplit     = 10,                        ! requested num. of subcycles
   dtmin      = 1e-18,                     ! minimum dt
   dtmax      = 100e-12,                   ! maximum dt
   dtinit     = 100e-12,                   ! initial dt
   dtrvar     = 0.025,                     ! nominal density var.
   dttevar    = 1.e12,                     ! nominal elec. temp. v. (disabled)
   dttivar    = 1.e12,                     ! nominal ion temp. var. (disabled)
   dtbreak    = 2.0,                       ! break step factor
   dtfactor   = 2.0,                       ! dt increment factor
   nexit      = 10000,                     ! maximum number of steps
   iwctrl     = 1,                         ! optimize deposition factors
!--output control
   dt_aout    = 1000000,                   ! ascii output time interval
   ns_aout    = 1000000,                   ! output step interval
   dt_bout    = 1000000,                   ! output time interval
   ns_bout    = 5,                         ! output step interval
   irad_left  = 101,                       ! left spectra interface
   irad_right = 201,                       ! right spectra interface
   nreduce    = 2,                         ! display one half of values
/
!--------------------------------------------------------------------------
```

```
&layer
   nc          = 40,                        ! number of cells
   thick       = 0.098,                     ! thickness
   r0          = 0.0003,                    ! density
   te0         = 0.0001,                    ! electron temperature
   ti0         = 0.0001,                    ! ion temperature
   zonpar      = 1.0,                       ! uniform grid
   material    = 'DT',                      ! thermonuclear fuel
/
&layer
   nc          = 60,                        ! number of cells
   thick       = 0.005,                     ! thickness
   r0          = 0.22053,                   ! density
   te0         = 0.0001,                    ! electron temperature
   ti0         = 0.0001,                    ! ion temperature
   zonpar      = 1.05,                      ! variable grid size
   material    = 'DT',                      ! thermonuclear fuel
/
&layer
   nc          = 25,                        ! number of cells
   thick       = 0.0024,                    ! thickness
   r0          = 1.860,                     ! density
   te0         = 0.0001,                    ! electron temperature
   ti0         = 0.0001,                    ! ion temperature
   zonpar      = 1.0,                       ! uniform grid
   material    = 'Cocktail',                ! dopped Be (4.86% Na, 2.1% Br)
/
&layer
   nc          = 75,                        ! number of cells
   thick       = 0.0136,                    ! thickness
   r0          = 1.860,                     ! density
   te0         = 0.0001,                    ! electron temperature
   ti0         = 0.0001,                    ! ion temperature
   zonpar      = 1.0,                       ! uniform grid
   material    = 'Be',                      ! pure beryllium
/
!-------------------------------------------------------------------------------
&material
   name        = 'DT',                      ! name
   zi          = 1,                         ! atomic number
   ai          = 2.5,                       ! mass number
   eeos_id     = 52711,                     ! electron EOS table (id)
   eeos_file   = 'tables/DT',               ! electron EOS table (file)
   ieos_id     = 52712,                     ! ion EOS table (id)
   ieos_file   = 'tables/DT',               ! ion EOS table (file)
   z_id        = -1,                        ! use zi for ionization
   planck_file = 'tables/DT20keV.PLANCK',   ! Planck opacity table (file)
   ross_file   = 'tables/DT20keV.ROSS',     ! Rosseland opacity table (file)
   eps_id      = -1,                        ! LTE emmissivity
/
&material
   name        = 'Cocktail',                ! name
   zi          = 4.,                        ! atomic number
   ai          = 9.,                        ! mass number
   eeos_file   = 'tables/BE_eos_e',        ! electron EOS  table (file)
   ieos_file   = 'tables/BE_eos_i',        ! ion EOS  table (file)
   z_id        = -1,                       ! use zi for ionization
   planck_file = 'tables/Mix20keV.PLANCK', ! Planck opacity  table (file)
   ross_file   = 'tables/Mix20keV.ROSS',   ! Rosseland opacity  table (file)
   eps_file    = 'tables/Mix20keV.EPS',    ! emmissivity  table (file)
/
&material
   name        = 'Be',                     ! name
   zi          = 4.,                       ! atomic number
   ai          = 9.,                       ! mass number
   eeos_file   = 'tables/BE_eos_e',        ! electron EOS table (file)
   ieos_file   = 'tables/BE_eos_i',        ! ion EOS table (file)
   z_id        = -1,                       ! use zi for ionization
   planck_file = 'tables/Be20keV.PLANCK',  ! Planck Opacity table (file)
   ross_file   = 'tables/Be20keV.ROSS',    ! Rosseland Opacity table (file)
   eps_id      = -1,                       ! LTE emmissivity
/
!-------------------------------------------------------------------------------
&pulse_shape
   ntab        = 9,                        ! number of values in ptab and ptab
   mtab        = 1,                        ! linear interpolation in ptab**expo
   expo        = 1,                        ! expo=1 => linear interpolation
   ttab= 0, 3.3, 4.6625, 6.025, 7.3875, 8.75, 10.1125, 11.475, 14.2,
   ptab=80,  80,     87,    96,    111,  130,     159,    186,  250,
/
!-------------------------------------------------------------------------------
```

Chapter 5

Output file

5.1 File structure

As in indicated above, the code dumps output quantities in file `fort.10`. This file is structured as indicated in Chap. 3, Sec. 6. The actual stored data depends on the code version and can be modified by the user. For example, by including line

```
call write_vect(i,icount,nr,'EE      ',n,s%ee)
```

in routine `write_output_record`, the specific energy of electrons is written in `fort.10` under the name `EE`. Undesired quantities, for example, the fusion energy in laser irradiated planar targets, can be swiched out by commenting the corresponding line in `write_output_record`. The number N_t of stored time frames depend on the options `dt_bout` and `ns_bout` given in namelist block `parameters`. For example, if `ns_bout=1`, data is stored in each time step. The number of stored values of space dependent variables (N or $N+1$) depends on parameter `nreduce` in namelist block `parameters`. Use `nreduce=1` to store all values, and `nreduce>1` to perform accurate simulations without creating too big output files. The physical dimensions of some quantities depend on the geometry. Energy is given erg in spherical simulations, erg cm^{-1} in cylindrical simulations, and erg cm^{-2} in planar simulations. The output of the standard version is described in the following table (for spherical geometry).

		Static values (section 1)
Name	Values	Description
`CMC`	N	mass coordinates at cells (g)
`NC`	N	number of each cell
`CMI`	$N+1$	mass coordinates at interfaces (g)
`NI`	$N+1$	number of each interface
`FREC`	N_g	central frequency of each radiation group (eV)
`FREI`	N_g+1	boundary frequencies between radiation groups (eV)
		Dynamic values (section 2)
Name	Values	Description
`TIME`	N_t	simulation time (s)
`ISTEP`	N_t	number of integration steps
`LASER`	N_t	incident laser energy (erg)
`ALPHAS`	N_t	energy lost by escaping α-particles (erg)
`FUSION`	N_t	fusion energy (erg)
`NEUTRONS`	N_t	number of neutrons

Table 5.1: Output file data

		Dynamic values (section 2)
Name	Values	Description
`ENIA`	N_t	energy of α-particles (erg)
`ENIE`	N_t	energy of electrons (erg)
`ENII`	N_t	energy of ions (erg)
`ENKI`	N_t	kinetic energy (erg)
`DELAS`	N_t	absorbed laser energy (erg)
`DERAD`	N_t	absorbed radiation energy (erg)
`DEFUS`	N_t	fusion energy coupled to plasma (erg)
`ENRL1`	N_t	radiated energy through the left boundary (erg)
`ENRL4`	N_t	radiated energy through the right boundary (erg)
`ENRG1`	N_t	radiation energy incident on the left boundary (erg)
`ENRG4`	N_t	radiation energy incident on the right boundary (erg)
`REF`	N_t	laser reflection factor
`TRA`	N_t	laser transmission factor
`VIMPLO`	N_t	average DT fuel implosion velocity (cm/s)
`RHORDT`	N_t	confinement parameter ($\int \rho dr$) for DT fuel (g cm^{-2})
`RHORHS`	N_t	confinement parameter for DT fuel with $T_i > 5$ keV
`IFAR`	N_t	In-Fight-Aspect-Ratio (layer with $\rho < \max(\rho)/e$)
`ALPHA`	N_t	Isentropic parameter (mass average $(P_e + P_i)/P_{Fermi}$ for DT cells with $\rho > \max(\rho)/10$)
`R`	$N \times N_t$	density (g cm^{-3})
`PT`	$N \times N_t$	pressure (dyn cm^{-2})
`TE`	$N \times N_t$	electron temperature (eV)
`PE`	$N \times N_t$	electron pressure (dyn cm^{-2})
`ZI`	$N \times N_t$	ionization
`XC`	$N \times N_t$	center of cell coordinate (cm)
`D`	$N \times N_t$	specific laser deposition (erg g^{-1} s^{-1})
`DENE`	$N \times N_t$	electron number density (cm^{-3})
`EA`	$N \times N_t$	α-particle energy density (erg cm^{-3})
`F`	$N \times N_t$	molar fraction of tritium
`TI`	$N \times N_t$	ion temperature (eV)
`PI`	$N \times N_t$	ion pressure (dyn cm^{-2})
`AD`	$N \times N_t$	laser deposition per unit of volume (erg cm^{-3} s^{-1})
`X`	$(N+1) \times N_t$	interface position (cm)
`V`	$(N+1) \times N_t$	velocity (cm s^{-1})
`TR`	$(N+1) \times N_t$	radiation temperature (eV)
`S+`	$(N+1) \times N_t$	outwards radiation flux times area (erg s^{-1})
`S-`	$(N+1) \times N_t$	inwards radiation flux times area (erg s^{-1})
`S`	$(N+1) \times N_t$	total radiation flux times area (erg s^{-1})
`I-`	$N_g \times N_t$	inwards radiation spectra at interface `irad_left` (erg s^{-1} eV^{-1})
`I+`	$N_g \times N_t$	outwards radiation spectra at interface `irad_right` (erg s^{-1} eV^{-1})

Table 5.2: Output file data (continuation)

Chapter 6

Table file format

Tables for equations of state, ionization, opacity and emissivity data are stored as plain text ASCII files composed of lines with a maximum length of 60 characters. Each line contains up to 4 fields of 15 characters (either a text string or a floating point number) and ends with the LF character (ASCII code 10)[1]. A file can contain several consecutive tables, each formed by a block of lines. The head line of each table defines type and structure of the table. For most of the tables, data is given for the nodes of a regular grid of size `NR`$\times$`NT` with density and temperature as independent variables. Data is sorted by increasing values of density and temperature: the first `NR` items correspond to the first value of temperature, the items from `NR`+1 to 2$\times$`NR` correspond to the second value of temperature, etc. For equations of state, pressure and temperature are given as functions of density and $\Delta e = e - e_0$; the specific internal energy above the cold energy $e_0(\rho) \equiv e(\rho, T_{ref})$, where T_{ref} is a sufficiently small reference temperature. The one-dimensional function $e_0(\rho)$ is included as part of the table. The first field of the head line is a table identifier `ID`. This value, as other integer values (`NR`, `NT`), can be coded in the file as a floating point number, but it is truncated to integer when the table is read. The structure of the rest of the table depends on table type.

6.1 Inverted equation of state tables

- Head line:
  - Field 1 - Identifier
  - Field 2 - Ignored (the material density (g cm^{-3}) at normal conditions is usually stored here)
  - Field 3 - NR: the number of tabulated densities
  - Field 4 - NE: the number of tabulated energies
- Subsequent lines[2]:
  - NR values of the density (g cm^{-3})
  - NE values of Δe (Mbar cm^3 g^{-1})
  - NR values of e_0 (Mbar cm^3 g^{-1})
  - NR$\times$NE values of pressure[3] (Mbar)
  - NR$\times$NE values of temperature (K)

[1]CR characters (ASCII code 13) after LF are ignored.
[2]Four values in each line, except in the last one.
[3]1 Mbar = 10^{12} g cm^{-1} s^{-2}.

6.2 Ionization tables

- Head line:
  - Field 1 - Identifier
  - Field 2 - Table type: `'␣0.60000000E+01'` (a string of 15 characters)
  - Field 3 - NR: the number of tabulated densities
  - Field 4 - NT: the number of tabulated temperatures
- Subsequent lines:
  - NR values of the decimal logarithm of ρ (g cm^{-3})
  - NT values of the decimal logarithm of T_e (in keV[4])
  - NR$\times$NT values of the decimal logarithm of Z

6.3 Opacity/emissivity tables

These tables are composed of several consecutive subtables whose format is

- Head line:
  - Field 1 - Identifier (the same for all subtables)
  - Field 2 - Table type:
    `'PLANCK␣1␣␣␣␣␣␣␣'` or `'␣0.70000000E+01'` - One group Planck opacity
    `'ROSSELAND␣1␣␣␣␣'` or `'␣0.40000000E+01'` - One group Rosseland opacity
    `'EPS␣1␣␣␣␣␣␣␣␣␣␣'` - One group emissivity
    `'PLANCK␣M␣␣␣␣␣␣␣'` - Multigroup Planck opacity
    `'ROSSELAND␣M␣␣␣␣'` - Multigroup Rosseland opacity
    `'EPS␣M␣␣␣␣␣␣␣␣␣␣'` - Multigroup emissivity
  - Field 3 - NR: the number of tabulated densities
  - Field 4 - NT: the number of tabulated temperatures
- Second line (only for multigroup tables):
  - Field 1 - FA: lower frequency boundary ν_k (eV)
  - Field 2 - FB: higher frequency boundary ν_{k+1} (eV)
- Subsequent lines:
  - NR values of the decimal logarithm of ρ (g cm^{-3})
  - NT values of the decimal logarithm of T_e (eV)
  - NR$\times$NT values of the decimal logarithm of the quantity. The tabulated opacities are expressed in cm^2 g^{-1}, the normalized emissivity is dimensionless

Subtables can be given in arbitrary order. For one-group tables, only a subtable is needed and the frequency range is assumed to cover the full spectra ($\nu_1 = 0$, $\nu_2 = \infty$).

[4]Notice that temperature is expressed in eV in the opacity tables but in keV in the ionization tables!

Chapter 7

Installation

The **MULTI-IFE** code can be installed and used in two alternative modes.

7.1 Self standing mode

All components of the code are packed in a *tar* file called `MULTI-IFE-2015`. All files should be installed in a user defined directory. For example, in a Linux system with the *tar* file present in the current directory, type:

```
mkdir multi
cd multi
tar xvf ../MULTI-IFE-2015
```

The following files are unpacked:

- `m150304.f95`: FORTRAN source code (version March 4, 2015)
- Input file examples: `2015CPC.input`, `2015PRE.input`, `2015IND.input`, and `2015DIR.input`.
- Output files for the examples: `2015CPC.output`, `2015PRE.output`, `2015IND.output`, and `2015DIR.output`.
- Subdirectory `tables` containing files with the material tables used in the examples.
- Subdirectory `cases` containing the same examples, but in individual *tar* files.
- `2015-MULTI-IFE.pdf`: this document.
- `MODINFO`: basic information.
- `README`: additional information.
- `FILELIST`: list of packed files.
- `CHECKSUM`: two checksum values for each packed file (as returned by Linux command `sum` with options `-s` and `-r`, respectively).

First, the code must be compiled. In Linux systems use

```
gfortran m150304.f95
```

to generate the executable (`a.out`). Then, copy one of the examples into input file `fort.12`[1]

```
cp 2015IND.input fort.12
```

[1]in some oparating systems one has to modify path names in `fort.12`, changing "/" by "\".

and execute the code

```
./a.out
```

Files `fort.10` and `fort.11` contains the results. The output on the screen should be equal to the contents of file `2015IND.output`.

7.2 Part of the MULTI environment

This is only available for Linux systems.

7.2.1 r94 system

The **MULTI-IFE** is a member of a family of codes that can be run on a common Graphic User's Interface (GUI), that includes graphic post-processors and programing tools. For example, the figures in Chap. 8 have been produced by that software. The interface is based on the modular **r94**-programing system. That system is installed as set of subdirectories of a base directory:

- `archive/` - contains *modules* as *tar* archives.
- `r94/` - contains installed *modules* in separate subdirectories: individual files with source code, data, executables, etc.
- `cases/` - simulation cases, each one in a separate subdirectory.
- `doc/` - documentation files.
- `work/` - working directory.

The base directory contains shell procedures to carry out basic tasks (installation, checking, and start interfaces). The *tar* file `MULTI-IFE-2015` is just a *module* more of this system, available in `archive/` subdirectory.

7.2.2 CPC package

Here we discuss the installation of the package (**r94**-system plus hydrocode **MULTI-IFE**) as originaly submitted to CPC. Future releases can differ in some details. The definitive name of the package file has to be assigned by the CPC library. Let us assume here that it is called `package.tar.gz` and it is present in the directory where you want to install the code (typically your home directory).

1. Unpack the package by typing:

   ```
   tar xvfz package.tar.gz
   ```

   A subdirectory called `multi_base` will be created.

2. This directory can be optionally moved or renamed (e.g., to have several concurrent installations).

3. Move to the base directory

   ```
   cd multi_base
   ```

4. Read file `README` and take a look to documentation files in `multi_base/doc/`.

5. Perform installation by typing:

```
./INSTALL
```

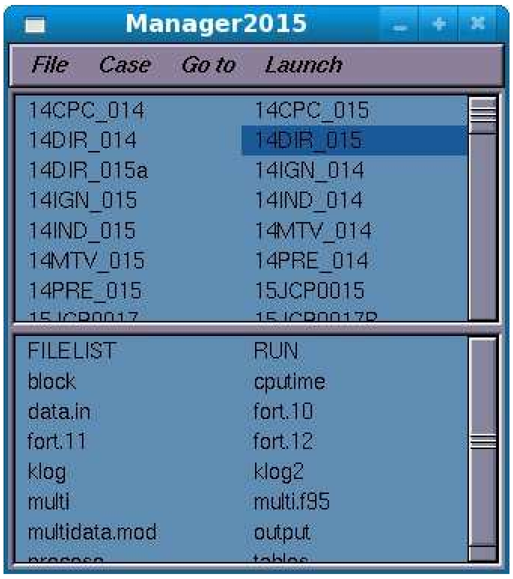
This procedure calls procedures called `install_*` to install individual sub-packages. In principle, these procedures trry only to install missing components. So that `./INSTALL` can be used also to complete a non-finished installation, to repair a damaged system (after removing damaged components), and to check the installation. Some problems can appear during installation, the known ones are:

- You are using a non supported operating system (like MS-Windows). In that case, you can use the code only in the self standing mode described in the previous section. The tar file `MULTI-IFE-2015` is available in `archive/`.
- The current directory is not writable or hard disk is full (about 30 Mb are required).
- Some components of the Linux system are missing (C-compiler, libraries, include files, commands, . . .). The operating system should be reinstalled or upgraded as "developing system". The utility procedure `check_system_2015.exe` can be used to identify missing components.

6. Start the graphic interface by typing

```
./MULTI
```

a window like the one shown in Fig. 7.1 should appear. Document `doc/2015-doc-GUI.pdf` explains how to run simulations and display results. A problem appear if the "`xterm`" command has not been installed, because the interface uses by default "`xterm`" shells to edit files and execute programs. Although the menu option **File→Options** allows to select "`gnome-terminal`" as shell, it is strongly recomended to install the "`xterm`" command.



```
14CPC_014      14CPC_015
14DIR_014      14DIR_015
14DIR_015a     14IGN_014
14IGN_015      14IND_014
14IND_015      14MTV_014
14MTV_015      14PRE_014
14PRE_015      15JCP0015

FILELIST       RUN
block          cputime
data.in        fort.10
fort.11        fort.12
klog           klog2
multi          multi.f95
multidata.mod  output
```

Figure 7.1: GUI Main window

Chapter 8

Examples

In this section, four examples of application of the code are given. Because the code **MULTI-IFE** contains all the physics included in codes **MULTI** [3] and **MULTI-fs** [2], the examples given in the corresponding publications can be reproduced by **MULTI-IFE**, and are included in the present examples. Together with a short description, two figures, one with a 3D representation of mass density as a function of cell number and time, and the other with a 2D plot of interface positions as functions of time, are given for each case. The CPU times are for a Dell Inspiron Notebook XPS M1330 with Intel Core 2 Duo T8300 processor at 2.4 GHz.

8.1 Au/Al foil submitted to a 300 ps pulse

This case (2015CPC) was given as example in [3]. A sandwich foil, 4 μm of aluminium covered by 0.2 μm of gold, is irradiated by a 300 ps FWHM pulse with a peak intensity of 3×10^{14} W cm^{-2} and 0.44 μm wavelength. The main difference with the simulation discussed in [3], is that **MULTI-IFE** uses separated temperatures for ions and electrons. CPU time is 1.56 seconds.

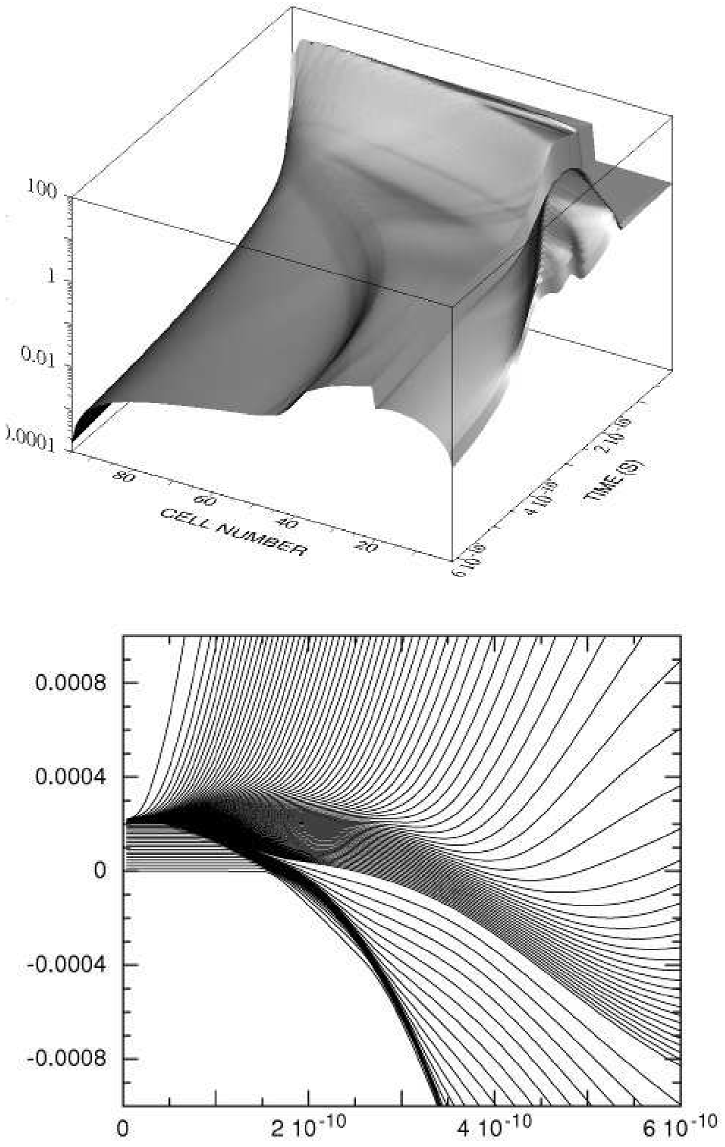


Figure 8.1: Case **2015CPC**: mass density $\rho(j,t)$ and interface motion $x_j(t)$.

8.2 Al foil submitted to a 300 fs laser pulse

This case (2015PRE) was given as example in [2]. This case was originally part of the results reported in [4]. A 150 nm thick aluminium foil is irradiated by a intense 10^{15} W cm^{-2} laser pulse with 150 fs FWHM duration and 400 nm wavelenght. CPU time is 11.13 seconds.

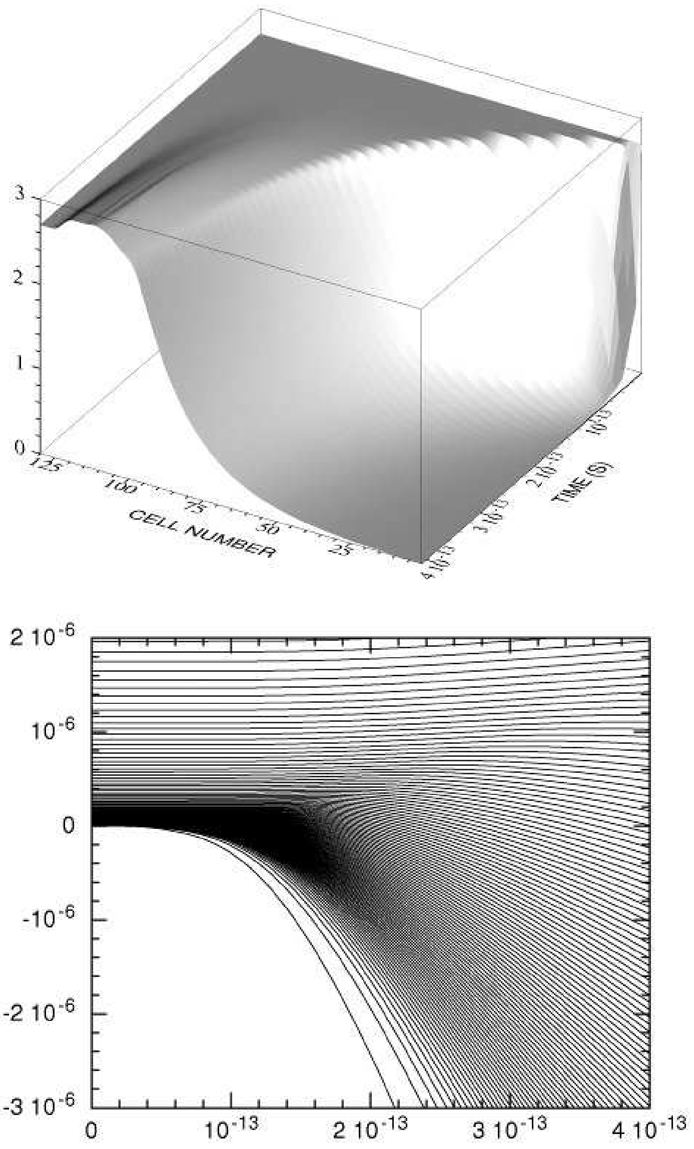


Figure 8.2: Case **2015PRE**: mass density $\rho(j,t)$ and interface motion $x_j(t)$.

8.3 Directly Driven IFE Target

This cases (2015DIR) is one of the examples in [1]. Described in detail in Chap 3. of [6] and in [7]. It represents a typical laser-fusion capsule. With an external radius of 2 mm, it uses plastic ablator covering a hollow shell of DT fuel (1.67 mg). Laser energy is 1.7 MJ and thermonuclear yield is 140.5 MJ. CPU time is 12.89 s.

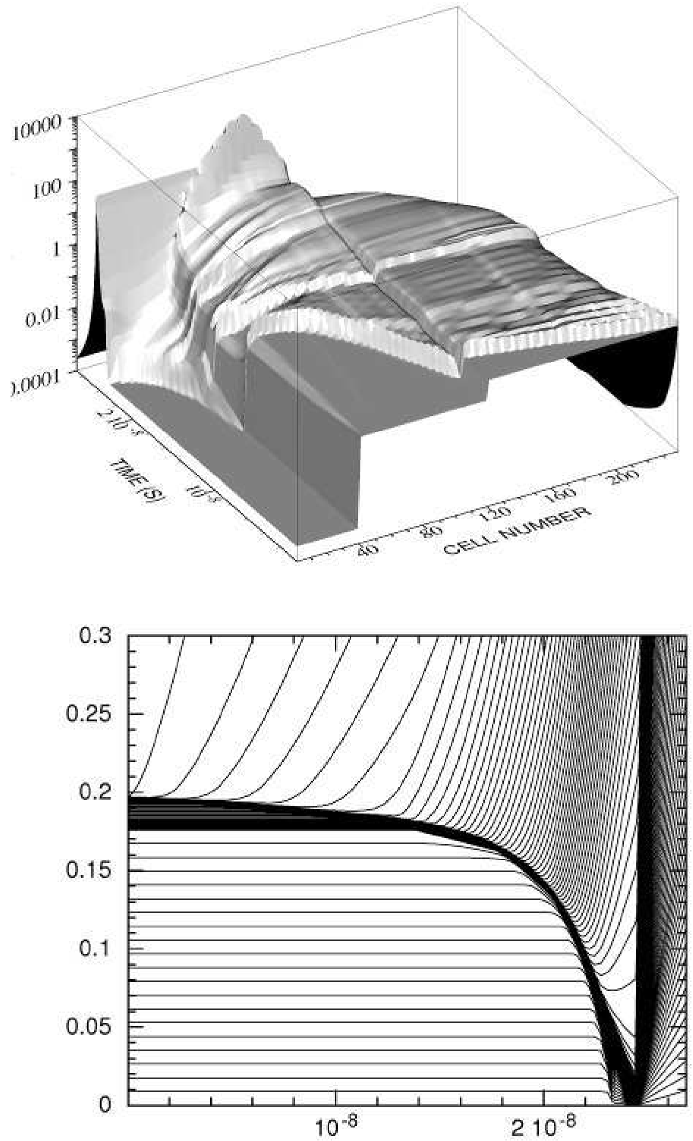
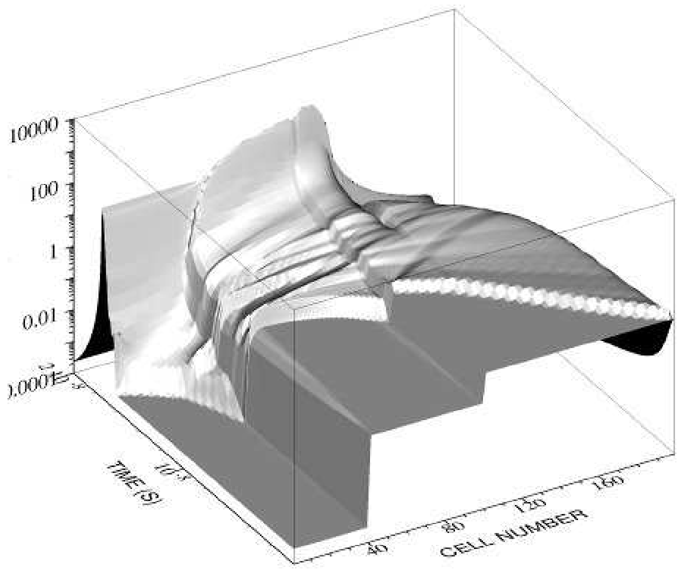


Figure 8.3: Case **2015DIR** mass density $\rho(j,t)$ and interface motion $x_j(t)$.

8.4 Indirectly Driven IFE Target

This case (2015IND) is one of the examples in [1]. This configuration is taken from [8]. The target geometry was described in Fig. 110 (pag. 4010). The radiation pulse used here is similar to the one from Fig. 111 (pag. 4010, line marked “>10 MJ”). This was one of the capsules initially proposed to be ignited at the NIF facility. It has an external radius of 1.19 mm and contains 0.14 mg of DT fuel. It uses beryllium ablator driven with a thermal bath of up to 250 eV. The absorbed radiation energy is 182 kJ and the yield is 13.4 MJ. CPU time is 17.86 s.



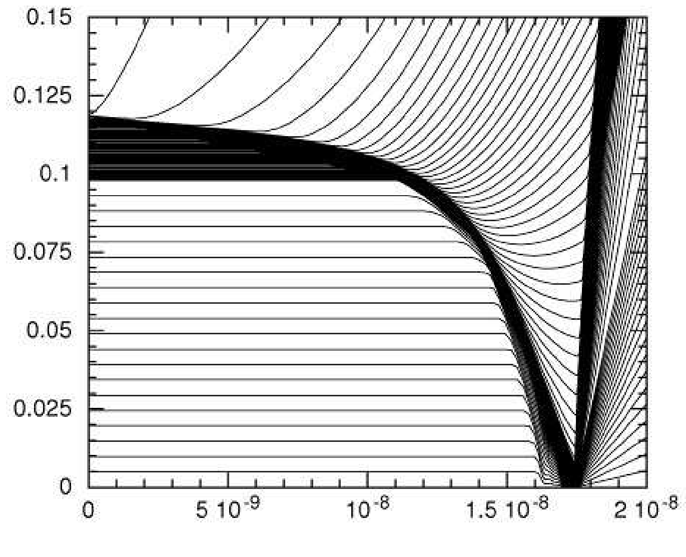


Figure 8.4: Case **2015IND**: mass density $\rho(j,t)$ and interface motion $x_j(t)$.

Bibliography

[1] R. Ramis and J. Meyer-ter-Vehn, *MULTI-IFE - A One-Dimensional Computer Code for Inertial Fusion Energy (IFE) Target Simulations*, Submitted to Computer Physics Communications.

[2] R. Ramis, K. Eidmann, J. Meyer-ter-Vehn, and S. Hüller, *MULTI-fs - A Computer Code for Laser-Plasma Interaction in the Fentosecond Regime*, Computer Physics Communications 183 (2012) 637-655.

[3] R.Ramis, R.J.Schmalz, and J.Meyer-ter-Vehn, *MULTI A Computer Code for One-Dimensional Multigroup Radiation Hydrodynamics*, Computer Physics Communications **49** (1988) 475-505.

[4] K. Eidmann, J. Meyer-ter-Vehn, T. Schlegel, and S. Hüller, *Hydrodynamic simulation of subpicosecond laser interaction with solid-density matter*, Phys. Rew. E, **62** (2000) 1202-1214.

[5] R. Ramis, *An implicit one-dimensional Lagrangian hydrodynamic method for inertial confinement fusion*, in preparation, to be submitted to Journal of Computational Physics.

[6] S. Atzeni and J. Meyer-ter-Vehn, *The physics of inertial fusion*, Claredon Press - Oxford (2004).

[7] R.L. McCrory and C.P. Verdon, *Inertial Confinement Fusion*, Proceedings of the Course and Workshop, 1988, (Eds. Caruso and E. Sindoni), pp. 83-123.

[8] J. Lindl, *Development of the indirect-drive approach to inertial confinement fusion and the target physics basis for ignition and gain*, Phys. Plasmas **2** (1995) 3933-4024.