eZND Documentation

Release 1.0

Kevin Moore

September 19, 2016

Contents:

CHAPTER 1

Introduction to eZND

The eZND code solves for steady-state detonations where post-shock expansion is important in determining the structure of the detonation. It is a 1D code, and therefore relies on adequate parametrization of the various multidimensional expansive effects as a function of distance behind the shock front.

It was written to compute the structure of surface helium detonations on accreting white dwarfs, presented in this paper. It also contains solvers for other auxiliary detonation properties (eg. CJ solutions, standard ZND structure, hydrostatic white dwarfs).

eZND uses MESA for its microphysics (equations of state, reaction networks, integrators, etc.) so you must have MESA installed on your system. For this reason, the code was written in Fortran and tested primarily with the build environment provided by the MESA SDK.

CHAPTER 2

Tutorial

eZND uses a Fortran namelist to specify inputs, just like the inlist file used in MESA. It is called **inlist_znd**, and lives in the same directory as the source code. Unless you're doing some heavy-duty modifications (eg. changing the post-shock evolution equations) then you shouldn't need to ever recompile the code.

All of the variables in the inlist have default values, listed in the inlist_controls section along with a detailed description of their function. Thus, if you provide an empty file for the inlist, then eZND will still run - computing its default calculation, finding the ZND structure of a CJ detonation in pure helium at $\rho_0 = 5 \times 10^5$ g/cm^3 and $T_0 = 10^8$ K.

This tutorial goes over the basic setup and output for some of the most common use cases.

2.1 Setting your MESA path

eZND uses MESA's microphysics modules, so needs to know where your local MESA installation lives. By default, it will look in your $MESA_DIR environment variable but if you have multiple MESA installations, you may want to tell eZND to use a different version. In that case you can specify the (full) path to the MESA installation with the inlist command,

```
my_mesa_dir = '/Users/Kevin/mesa_5271'        !MESA installation directory to use
```

where the full path is contained in quotes.

2.2 Setting a nuclear reaction network

The reaction networks are set up in MESA format, using data from NACRE, Reaclib, and other sources (see the MESA instrument papers for more detail). To specify a reaction network file, use

```
net_file = '88_iso.net'       !Nuclear reaction network to use
```

where the network you want is in quotes. This can be a relative or absolute path to the network file. MESA's default networks are stored in the installations **data/net_data/nets** folder.

2.3 Setting the initial conditions

The initial conditions for eZND consist of thermodynamics (ρ_0 and T_0) and composition. The thermodynamic conditions can either be set directly, or if the calculation is to be mapped to a white dwarf surface are set implicitly by setting the white dwarf core and envelope masses.

If the thermodynamic conditions are specified directly, the following commands are used

```
rho_0 = 5d5   !Initial density (g/cm^3)
T_0 = 1d8     !Initial temperature (K)
```

If the thermodynamic conditions are set implicitly from a white dwarf core/envelope mass then these are set using

```
m_c = 1.0             !white dwarf core mass (solar masses)
m_env = 0.05          !white dwarf envelope mass (solar masses) - need to actually implement this!
```

The mass fractions of the initial isotopes is specified in a few steps. First, you set the number of different isotopes in the initial composition with

```
num_isos_for_Xinit = 3       !Number of isotopes in initial composition
```

where this integer can be anything between 1 and the number of isotope species in the reaction network. Each isotope is then identified and assigned a mass fraction with the following two lines (per isotope in the initial mixture)

```
names_of_isos_for_Xinit(1) = 'he4'    !Isotope name
values_for_Xinit(1) = 0.8             !Isotope mass fraction
```

The names of the isotopes can be found in the **isotopes.data** file located in MESA's **data/chem_data** folder.

2.4 Selecting run mode

eZND can be run in a variety of modes, from calculating CJ conditions as a function of initial density to calculating generalized CJ velocities for a variety of white dwarf envelope masses. From here, the inlist controls vary enough that we will go through each runtime mode separately.

2.4.1 Finding naive CJ velocities as a function of initial density

This mode sweeps through a variety of initial densities, calculating the CJ conditions for each. It does not do a post-shock integration so does not calculate the post-shock structure for any of the detonations. Currently, an NSE solver is not implemented so the CJ state is calculated from an assumed final burning state. If the final state is not known, then the CJ state can be iteratively solved for using post-shock integration, see *Finding generalized CJ velocities as a function of initial density*.

In this case expansive effects cannot be included (hence the 'naive'), so any specification of blowout and curvature source terms will be ignored. The controls for sweeping through densities are described in the section *Sweeping through initial densities*.

2.4.2 Finding generalized CJ velocities as a function of initial density

This mode is similar to *Finding naive CJ velocities as a function of initial density*, sweeping through a variety of initial densities, except here the post-shock integrals are performed so the CJ (or generalized CJ) solution can be found iteratively.

2.5 Sweeping through initial densities

Several run modes involve sweeping though an initial density, ρ_0, directly:

- *Finding naive CJ velocities as a function of initial density*

- *Finding generalized CJ velocities as a function of initial density*

To set the densities to use,

```
rho_sweep_log_min = 5.0              !Lowest log-density to use
rho_sweep_log_max = 7.0              !Highest log-density to use
```

while the number of steps to take between the maximum and minimum values (inclusive) is set with

```
rho_sweep_num_steps = 50             !Number of steps in density to take
```

The number of steps must be at least 2 since the min and max values must be calculated. The code will take equally spaced steps in log-density specified by the formula:

$\rho_0 = 10^{(\mathrm{rho\_sweep\_log\_min}+(\mathrm{rho\_sweep\_log\_max}-\mathrm{rho\_sweep\_log\_min})*(i-1)/(\mathrm{rho\_sweep\_num\_steps}-1))}$

where i is the integer index in the range 1..rho_sweep_num_steps.

If expansive effects are enabled, then the thickness-scale height relation can be specified with

```
d_hs_scaling = 0.5    !H = d_hs_scaling*H_scale
```

2.6 Numerical controls

Various controls for the numerical solvers are available. Let's start with the generalized ZND equation integrator.

2.6.1 ZND Integration

The number of output steps (spaced equally in log-distance behind the shock front) is given by

```
     num_steps = 1000  !For typical production runs - these are data output points, not integration s
```

and the integration range (in cm) is specified by

```
     burn_time = 1d10          !cm for ZND if doing a spatial integration, sec if integrating in time
```

Each output step consists of a call to the stiff ODE integrator, isolve(). The number of steps that this routine can take internally can be set with

```
     max_steps = 1000          !Max number of internal integration steps per isolve() call
```

There are also a variety (7 as of MESA v5271) of implicit solvers used in the numerical integration. These are specified by integers, listed in MESA's **num/public/num_def.f** file. The solver is specified with

```
     which_solver = 4  !ros3pl_solver
```

Tolerances for accepting an internal step in the numerical integration (in between output points) are specified in absolute and relative terms. These are explained in MESA's **num/public/num_isolve.dek** file, and set by

```
     rtol_init = 1d-8          !Relative tolerance for accepting integration step
     atol_init = 1d-8          !Absolute tolerance for accepting integration step
```

Many of the implicit solvers can be sped up if the Jacobian can be evaluated analytically at the given point (rather than by finite differences). The switch that controls whether the analytic Jacobian is used is

```
     ijac = 0                  !0:numeric, 1:analytic (Jacobian for ZND integrator)
```

2.6.2 Pathological point traversal

There are a few controls for how the pathological point is traversed. It is usually possible to automate jumping over the pathological point if the generalized CJ velocity is determined to a high enough accuracy (I typically use a relative tolerance of 1d-5, but your milage may very of course).

Assuming you have determined the pathological point, you may

CHAPTER 3

Index