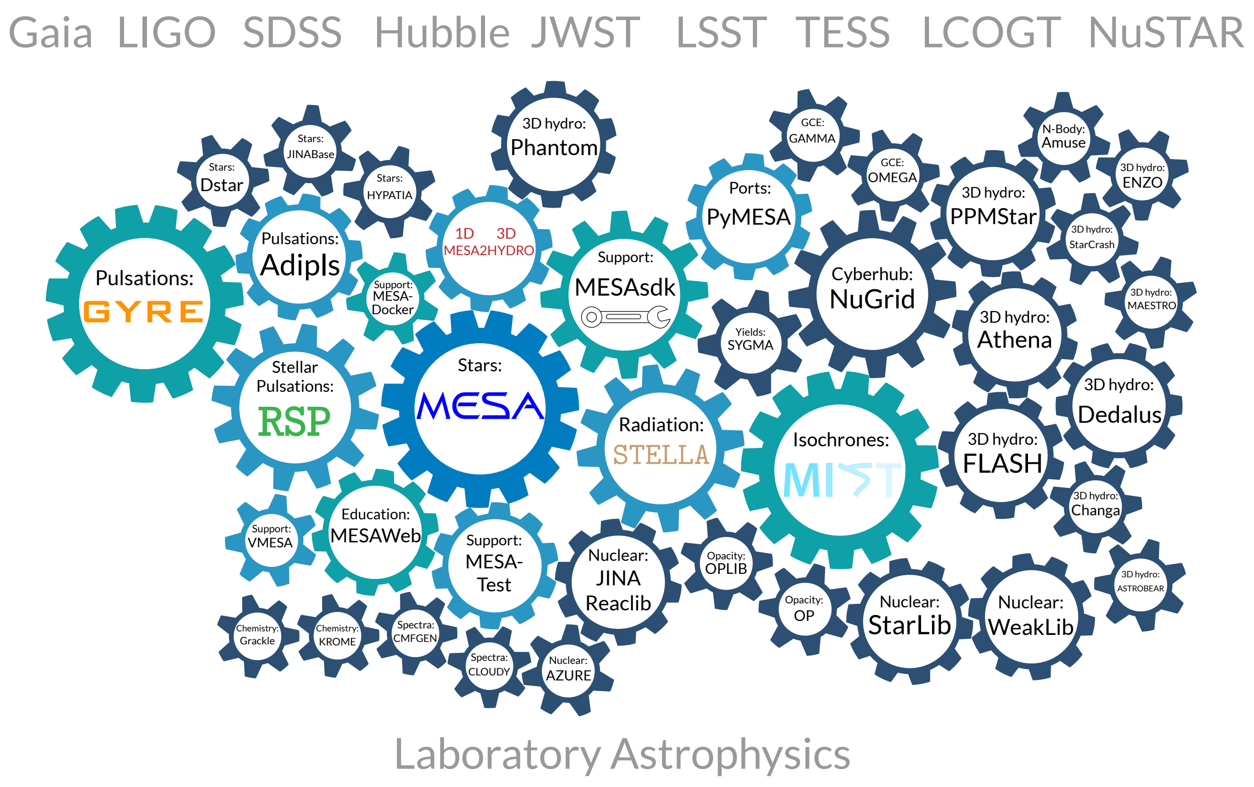


$^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$

The User's Guide

Meridith Joyce[1,2] & Lianne Lairmore[3]

[1]Research School of Astronomy & Astrophysics, Australian National University, Canberra, Australia
[2]Department of Physics & Astronomy, Dartmouth College, New Hampshire, USA
[3]Robotics Engineering Department, KeyMe, New York, NY, USA

Contents

1 Declaration of Use

If you use this package or any of its components, please cite the paper:

```
@article{MESA2HYDRO,
  doi = {10.3847/1538-4357/ab3405},
  url = {https://doi.org/10.3847%2F1538-4357%2Fab3405},
  year = 2019,
  month = {sep},
  publisher = {American Astronomical Society},
  volume = {882},
  number = {1},
  pages = {63},
  author = {M. Joyce and L. Lairmore and D. J. Price and S. Mohamed and T. Reichardt},
  title = {Density Conversion between 1D and 3D Stellar Models with 1DMESA2HYDRO3D},
  journal = {The Astrophysical Journal}
}
```

2 Introduction

The ^{1D}MESA2HYDRO^{3D} package can be used to render a one-dimensional (1-D) stellar density profile as a three-dimensional (3-D) particle distribution representing some or all of a model star. Using MESA and our Python interface, the user can create custom stellar models and convert those directly to hydrodynamical initial conditions (ICs). In this guide, we provide instruction on installing and operating the software and the necessary details required to reproduce the test cases laid out in Joyce et al. (2019).

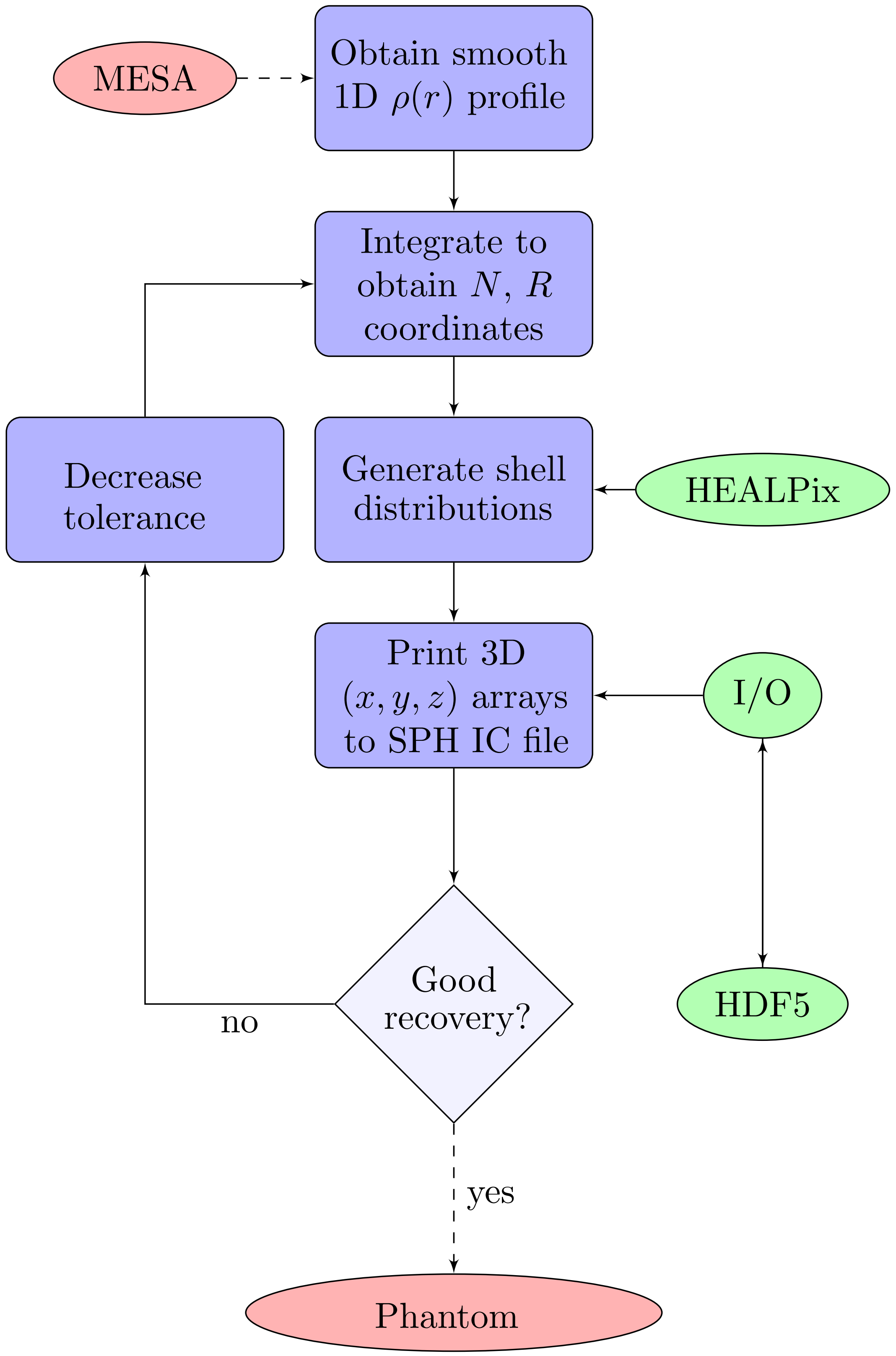
3 Algorithm

The mapping occurs by spatially parameterizing a 1-D density profile generated by a MESA, or other stellar structure and evolution code (SSEC), as a set of N, R coordinates. The profile must be given in the form of discrete $\rho(r), r$ data (density as a function of radius). N, R coordinates, named for "number" and"radius," describe the count and relative locations of particles representing some shellular cross section of a star. The 3-D coordinates for each particle are imparted using the HEALPix spherical tessellation algorithm (described below).
The algorithm for computing a set of shell radii proceeds as follows:

- A discrete density profile $r, \rho(r)$ corresponding to some percentage of the mass or radial distribution (as specified by the user) is extracted from a smooth MESA profile or similarly formatted file. The region representing the core mass is separated from the region to be rendered as SPH particles.
- At the base of the region to be translated, we search for a solution to the equality

$$m_{\text{shell}} = \int_{r_l}^{r_u} 4\pi r^2 \rho(r)\, dr = (12N^2) m_p, \tag{1}$$

by shifting the upper bound on the mass shell integral, $r_{u,0}$, surface-ward from its local position until the integrated mass and mass from the summation of HEALPix particles are equal to within some user-defined tolerance, δ_{TOL}.
In equation (1), m_p is the mass per particle and N is the HEALPix integer, both of which are set by the user. The choice of m_p and δ_{TOL} have the largest effect on the computation time: higher


MESA
Obtain smooth 1D $\rho(r)$ profile
Integrate to obtain N, R coordinates
Decrease tolerance
Generate shell distributions
HEALPix
Print 3D (x, y, z) arrays to SPH IC file
I/O
Good recovery?
no
HDF5
yes
Phantom

values of m_p translate to less frequent solutions and hence a lower resolution profile and shorter computation times, whereas lower values of δ_{TOL} correspond to increased precision on the location of r_u and thus longer computation times. The default tolerance is $\delta_{\mathrm{TOL}} = 0.01$.

- When one instance of equality (1) is satisfied, the coordinates N and $r_{\mathrm{mid}} = (r_u + r_l)/2$ are recorded in a standard text file with the prefix "NR." For other physical quantities, such as internal energy E or temperature $\log T$, $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ searches the MESA data directly for the r values bordering r_{mid} and linearly interpolates between them to produce approximate values for $E(r_{\mathrm{mid}})$, $\log T(r_{\mathrm{mid}})$, etc., as desired.

Following the computation of one such r_u, the subsequent lower bound $r_{l,1}$ is set to $r_{u,0}$, and the process repeats until $r_{l,1}, r_{u,1}$ again satisfy equation (1).

- The calculation of placement radii r_{mid} continues until $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ has subdivided the profile into k regions of variable size $(r_u - r_l)_j$, where $j = 1, ..., k$. Each region j is then uniquely characterized by its N, R coordinate pair. The generation of an NR file can take anywhere from several minutes to several hours depending on the choice of $m_p, \delta_{\mathrm{TOL}}$, and the penetration depth. The completed NR file is then passed to HEALPix via **healpy**.

- For each shell k, HEALPix distributes $n_{p,k}$ particles across the surface of a sphere with radius $R_k = r_{\mathrm{mid},k}$ using the equal cell method described in Górski et al. (2005).

- Having obtained $12N^2$ sets of (x, y, z) coordinates for the associated particles, the shells are stacked concentrically to form a 3-dimensional, hollowed sphere by normalizing each HEALPix shell by its placement radius relative to the total stellar radius.

- Each shell is arbitrarily rotated with respect to its neighbors in order to avoid ordered particle alignments. The rotated coordinates $(x', y', z')_k$ are computed via the multiplication of $(x, y, z)_k$ by the unit matrices

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

with θ, ϕ and ψ pseudo-randomly generated over the interval $[0, 2\pi]$. The pseudo-random number generator used is Python's **random.random**, a wrapper for the Mersenne Twister algorithm (Matsumoto & Nishimura, 1998), which uses the current timestamp as a seed. New values of θ, ϕ and ψ are computed for every k but not for every particle; otherwise, the distribution provided by HEALPix would not be preserved.

- The final set of k stacked, rotated, concentric sets of (x, y, z) coordinates and the MESA attributes assigned to them particle-by-particle are output to a file with the prefix "IC." These arrays can be passed to subroutines that organize the data into file structures compatible with various hydrodynamics codes directly. Currently, $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ supports output in the GADGET-2 unstructured binary and hdf5 formats, the Phantom binary format, and a simple ASCII text file. The user may control the precision with which the numerical data are written to the IC file, as well as the format of that file, using flags in the configuration file. For example, `filetype=phantom_binary` will produce a binary file in the Phantom format.

- $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ can reload the 3-D data it has generated directly and reduce it to a 1-D $r, \rho(r)$ curve using binning parameters specified by the user.

The integral in equation (1) is solved using a fourth-order Runge–Kutta scheme with adaptive step size refinement (Runge, 1895). This method was found to be more well-suited to our problem than, for example, Python's scipy.integrate function, due to the large variation in radial width that can correspond to a fixed shell mass. One can provide an initial guess for the integration step size in the configuration file, though this will be adjusted automatically as necessary depending on the

local shape of the density profile and on the particle mass and solution tolerance provided by the user. An inappropriate choice in step size may prolong the first few shell calculations, but it will not have a large impact on the computation time. Changes in the solution tolerance, however, scale linearly with computation time.
The ^{1D}MESA2HYDRO^{3D} workflow is subdivided into two main procedures: the first translates a 1-D density profile to an NR file by calculating the shell placement radii, and the second translates an NR file to an IC file using the radial spacings and the HEALPix tessellation. As the former conversion takes much longer than the latter, the subroutines are written to be executable in isolation. Within these subroutines, many other components of the workflow can be isolated by manipulating the appropriate flags in a configuration file.

4 Input

^{1D}MESA2HYDRO^{3D} reads input data and run time parameters from text files organized into `keyword, value` pairs. These are called "configuration files" and use a ".cfg" extension. The format of a configuration file follows the convention of a Fortran namelist, as is commonly used in other stellar modeling packages. The user can also specify configurable values through the command line directly.

4.1 Configuration Files

In a .cfg file, keywords are on the left side of an equals sign and the desired values are on the right. Each *name, value* pair must be on its own line. Boolean values must be *true* or *false*, but the case does not matter (TRUE/FALSE, True/False, true/false all are legal values).
At execution,the –config-file flag is used to tell run.py the name and location of the configuration file. The complete path to the configuration file from the current directory is needed.
To issue a run with arguments passed via configuration file (e.g. mainsequence.cfg), the command line should read

```
./run.py mainsequence.cfg
```

or

```
python run.py mainsequence.cfg
```

The contents of a configuration file should look similar to this:

```
# This is a sample for a config file for run.py
MESA_file = /home/mjoyce/MESA2HYDRO/data/sample_MESA_output/profile_ms.data
check_MESA_profile = True
make_NR_file = True     # Have M2H make a new NR file
make_IC_file = True     # Have M2H make a new IC file
new_NR_filename=     out/NR_files/NR_ms_test.dat
new_IC_filename=     work/IC_ms_test
# loaded_NR_filename = out/NR_files/NR.ms.dt0.01.R0.00.mp1m5.dat
# loaded_IC_filename=  work/IC_ms.dt0.01.R0.00.mp1m5
IC_format_type = phantom_binary     # data format for output IC (recommended)
# masscut = 0.999     # corresponds to 5% depth cut by mass
r_depth = 0.96        # depth cut by radius...always overrides depth by mass
N = 8                 # HEALPix integer
mp = 1e-7             # solar mass units
stepsize = 1.0e8      # initial guess for step size
                      # at start of each shell integral
TOL=0.01              # solution tolerance
```

Configuration (`*.cfg`) files understand Python comments (#) and revert to default values if the user-given value is not understood.

4.2 Default Parameter Values

The default parameter values can be found in the `SCRIPT_CONFIGS` dictionary in **run.py**. New parameters and default values can also be added there.

```
SCRIPT_CONFIGS = {
    'check_MESA_profile': True,
    'MESA_file': 'data/sample_MESA_output/profile_mainsequence_logE.data',
    'make_NR_file': False,
    'loaded_NR_filename': 'work/NR_files/saveNR_ms.dat',
    'new_NR_filename': 'latest_NR.dat',
    'make_IC_file': False,
    'loaded_IC_filename': 'ms_logE',
    'new_IC_filename': 'latest_IC',
    'IC_format_type': 'phantom_binary',
    'masscut': 0.95,
    'r_depth': 0.5,
    'N': 8,
    'mp': 1e-7,
    'mp_cgs': 1.988e26,
    'stepsize': 2.45e6,
    'which_dtype':'f',
    'TOL':0.01}
```

There are a few cases where parameter specifications may conflict. Given competing values for particle mass `mp`, ^{1D}MESA2HYDRO^{3D} will default to solar mass units. Given competing values for the radial/mass depth, `masscut` and `r_depth`, ^{1D}MESA2HYDRO^{3D} will default to the "mass contained" within `r_depth`. Given conflicting instructions to make a new file or use an old one, ^{1D}MESA2HYDRO^{3D} will load the existing NR (IC) file unless `make_NR_file` (`make_IC_file`) is set to True. If no value is provided for some parameter in the configuration file, the value listed here will be assumed.

4.3 Parameter Description and Options

Table 4.3 gives the configuration file parameters alongside their allowed values, Python data type, and description of function.

4.4 Command Line Arguments

The user may also set configuration values via command line arguments. The help flag

```
./run.py --help
```

will list all command line configurable options. Boolean options are set by specifying a flag alone, while all other options are specified with a flag followed by the desired value:

```
./run.py --check-MESA --N 16 --r_depth 0.80 --make-IC-file
```

This example sets `check_MESA` and `make_IC_file` to **True**, sets `N` to 16, and sets `r_depth` to 0.80. The order of the flags does not matter as long as the intended value immediately follows the flag. Currently, the configuration file flag and the configuration setting flags cannot be used together. Any value not set in either place will assume a default value. A complete list of configurable values can be found with the –help flag. The –defaults flag will print the internal defaults and exit.

Table 1: Configuration File Parameters

Name	Options	Data Type	Description
`check_MESA_profile`	True/False	boolean	load a plot of the input profile before calculation
`MESA_file`	file name	string	name of the MESA profile to convert
`make_NR_file`	True/False	Boolean	make new or use old NR file (default: old)
`loaded_NR_filename`	True/False	Boolean	name of existing NR file to be read
`new_NR_filename`	True/False	Boolean	name of new NR file to be generated
`make_IC_file`	True/False	Boolean	make new or use old IC file (default: old)
`loaded_IC_filename`	file name	Boolean	name of existing IC file to be read
`new_IC_filename`	file name	Boolean	name of new IC file to be generated
`IC_format_type`	'phantom_binary'	flag	Phantom structured binary
	'binary'	flag	GADGET-2/GIZMO unstructured binary format 1
	'hdf5'	flag	GADGET-2 hdf5 binary format
	'text'	flag	ascii text file; default
`masscut`	number	float	depth of penetration from surface by mass, 0 to 1
`r_depth`	number	flag	depth of penetration from surface by radius, 0 to 1
`N`	number	integer	HEALPix integer, $N \in \{2^x\}$
`mp`	number	float	mass per particle, $M_\odot$ (overrides cgs)
`mp_cgs`	number	float	mass per particle, cgs
`stepsize`	number	float	initial guess for Runge-Kutta step size, cm
`which_dtype`	'f'/'d'	string	float 'f' or double 'd' precision
`TOL`	number	float	integration solution tolerance

4.5 Input MESA/SSEC Data Format

While it is not (in theory) necessary to use the inlists, profile_columns, and history_columns prescriptions for MESA provided with this suite, some subroutines assume a particular format for the MESA data. Recovery has only been verified when this format was used.
The header of a MESA profile that is readable by ^{1D}MESA2HYDRO^{3D} is formatted as follows:

```
(line 1) 1  2  3  4  5  6  7  8  9 10 11 12 3 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
(line 2) model_number    num_zones initial_mass    initial_zstar_age    time_step
Teffphotosphere_Lphotosphere_r    center_eta    center_h1    center_he3    center_he4
center_c12    center_n14    center_o16  center_ne20    star_mass    star_mdot
star_mass_h1star_mass_he3star_mass_he4star_mass_c12star_mass_n14star_mass_o16
star_mass_ne20 he_core_mass  c_core_mass  o_core_mass si_core_mass fe_core_mass
neutron_rich_core_mass   tau10_mass tau10_radius  tau100_masstau100_radius
dynamic_time kh_timescalenuc_timescale   power_nuc_burn power_h_burnpower_he_burn
power_neu    burn_min1    burn_min2 time_seconds
(line 3)            57785    1.0000000000000000E+000    2.0000000000000000E-002
  5.0000000000000000E+009    2.0202602043678632E+008    5.7596827540378790E+003
  1.0132344316590454E+000    1.0123173502173932E+000   -1.5113464067642879E+000
  3.1060348935066984E-001    5.9709952550457791E-006    6.6892475065480950E-001
  2.3136476321419881E-005    5.7240751976072694E-003    8.5287930921241241E-003
  2.0994599496981497E-003    1.0000000000000000E+000    0.0000000000000000E+000
  6.5823041794771087E-001    7.6530927338120010E-004    3.2085402682590725E-001
  2.5315830371697599E-003    2.0798345475335099E-003    9.3490441350858634E-003
  2.0994599496976882E-003    0.0000000000000000E+000    0.0000000000000000E+000
  0.0000000000000000E+000    0.0000000000000000E+000    0.0000000000000000E+000
  0.0000000000000000E+000    9.9999999989740374E-001    1.0121039705504904E+000
  9.9999999982281740E-001    1.0119857705574893E+000    1.0197849350849037E+004
  2.2885105513658181E+007    9.8693843078607616E+009    1.0134436805771265E+000
  1.0134436805771267E+000    3.4867849786949279E-043    2.5223359611021012E-002
  5.0000000000000000E+001    1.0000000000000000E+003    1.5779074992000000E+017
(line 4)
(line 5)   1  2  3  4  5  6 7  8  9 10
```

```
(line 6)   zone    mass    logR    logT  logRho    logP logE
   x_mass_fraction_H    y_mass_fraction_He  z_mass_fraction_metals
(line 7)   1    1.0000000000000000E+000   5.3166803327612203E-003   3.7606022794428209E+000
        -6.8095799232935716E+000  4.7554327493173005E+000   1.2313339024408560E+001
         7.0000000000000007E-001  2.8000000000000008E-001   1.9999999999999796E-002
```

where "(line x)" is not part of the structure. Lines 1 through 3 specify MESA model attributes which are not needed by ^{1D}MESA2HYDRO^{3D} but may be needed for additional processing with MESA, and lines 4 and 5 are purely organizational; however, ^{1D}MESA2HYDRO^{3D} assumes a header containing 5 lines, which it will pass. Line 6 contains the attribute names ^{1D}MESA2HYDRO^{3D} understands, and lines 7 through the end of the profile file contain the stellar structure data. The user may use any SSEC code (e.g. Dotter et al. (2008); Demarque et al. (2004); VandenBerg et al. (2006); Pietrinferni et al. (2004); Cordier et al. (2007)) to generate an input profile, so long as it is formatted as above. This profile header is pulled from the MESA profile of a solar-like main sequence model, included in our test suite.

5 Output

^{1D}MESA2HYDRO^{3D} relies on two critical data structures: the **NR file** and **IC file**. Either of the subprocesses that generate these files can be turned off via flags at the command line or in the configuration file. The NR file can take several hours to compute from the MESA profile, depending primarily on particle mass and solution tolerance. The NR $\rightarrow$ IC process takes no more than a few seconds for reasonable particle numbers.

5.1 NR Files

The "heavy lifting" that this software performs is the generation of an NR coordinate file from an SSEC model. An NR file is a basic text file containing 4 columns of numerical data:
(1) HEALPix integer N (dimensionless),
(2) radii r (cm),
(3) masses M (grams),
(4) internal energy E (ergs)
The first column contains the integer used by HEALPix to determine the number of particles n_p distributed over a spherical shell. In order to divide the surface of a sphere into equal-area regions, the HEALPix tessellation requires $n_p = 12N^2$ and that N be a power of 2. Allowed values of N are thus 2,4,8,16, etc. (see Górski et al. (2005) for more information).
The radii represent midpoint values $r_{\text{mid}} = (r_u + r_l)/2$, in physical units (cm), corresponding to the radius at which a given shell must be placed in order to recast the mass contained in a region $(r_u - r_l)$ as a shellular distribution of particles. The third column gives the mass contained in this interval (in $M_\odot$). The fourth gives the internal energy (ergs). The length of the file corresponds to the number of shells needed to reconstruct the sampled profile as a nest of concentric surface particle distributions.
The only quantities ^{1D}MESA2HYDRO^{3D} technically requires to build a particle distribution are mass, radius, and density. However, additional physical information is necessary to build the appropriate arrays on the 3D end. In order for the distributions to achieve hydrostatic equilibrium (HSE), pressure and temperature (equivalently, internal energy per unit particle) must also be propagated.
Any number of additional physical quantities can be tracked in the NR files by adding the MESA column name to the NR writing routine. Some physical quantities and their recognized names are

Table 2: Allowed Values for Input Profile Columns

No.	Physical quantity	Column Name	Required?
1	Radial shell number	zone	no
2	Mass contained in shell ($M_\odot$)	mass	yes
3	$\log_{10}$ Radius (cm)	logR	yes
4	Local Temperature $\log_{10}$T (K)	logT	no
5	$\log_{10}\rho$ (g/cm^3)	logRho	yes
6	Pressure $\log_{10}$P (cgs)	logP	yes for HSE
7	Internal energy $\log_{10}$E (ergs)	logE	yes for HSE
8	Hydrogen abundance (fractional)	x_mass_fraction_H	no
9	Helium abundance (fractional)	y_mass_fraction_He	no
10	Metal abundance (fractional)	z_mass_fraction_metals	no

Number in the first column corresponds to the column number in the example header provided

given in Table 2.
If one forgets to track a particular quantity during the writing of the NR file, the value at the appropriate R coordinate can be found after-the-fact by interpolating the input MESA profile data. ^{1D}MESA2HYDRO^{3D} uses this backward-hunting method to source the appropriate values of, e.g., pressure during the writing of the IC files in any case. There should not be a difference between pressure values passed during NR computation versus those found after, based on equivalent radii and input profiles.
The header, first few lines, and footer of an NR file look like this:

```
(line 1) ## fname /home/usr/MESA2HYDRO/data/sample_MESA_output/profile_mainsequence_logE.data
masscut 0.9832621098341146  N 8   mp (Ms) 1e-07   mp (g) 1.988e+26   stepsize 1.0000000e+08
(line 2) #N    (ru+rl)/2 (cm)    Mcontained in shell ru-rl     u(rmid)
(line 3) 8 52836985702.89638 1.52715912738e+29 386768540555151.5
(line 4) 8 52869798202.89638 1.52334100923e+29 386768540555151.5
(line 5) 8 52902610702.89638 1.51950987585e+29 386768540555151.5
(line 6) 8 52935423202.89638 1.51566571402e+29 380430917840439.9
.
.
.
(line 223) 8 69881907577.89638 0.0 15375658296637.578
(line 224) #
(line 225) #
(line 226) # runtime:  2669.62379599  seconds
```

5.2 IC Files

^{1D}MESA2HYDRO^{3D} optionally writes the 3-D particle distributions directly to some SPH-compatible initial conditions ("IC") file formats. This I/O feature has been tested most rigorously for compatibility with Price et al. (2018)'s *Phantom* smoothed-particle hydrodynamics code, but options to write output in the style of GADGET-2's binary format 1, GADGET-2 HDF5, and basic ascii text files are also available. Where formatting directly to the desired IC type is not available, it is likely that Price (2007)'s SPLASH can perform the appropriate conversion via, e.g.,

```
  ssplash phantom_file.tmp to ascii
```

where `phantom_file.tmp` is the phantom_binary–formatted output from ^{1D}MESA2HYDRO^{3D}, and `ascii` is the keyword SPLASH understands for converting to a basic text file. Options for conversions within SPLASH can be found in the SPLASH documentation. As of the current release, this is probably the safest way to obtain initial conditions that are not in the Phantom format directly, but one must **be aware that the definitions for smoothing lengths differ between Phantom and GADGET-2**. Module **io_lib.py** handles this via slightly different definitions of `hsml` (smoothing length) in the write routines for Phantom versus GADGET-2.

6 Prerequisites

^{1D}MESA2HYDRO^{3D} is written in Python 2.7. **It will be re-released in Python 3 at the end of October, 2019** Components that interface with MESA have been developed using MESA version 10398. MESA-v10398's protocols and capabilities are approximately associated with the third instrument paper (Paxton et al., 2015), though MESA versions are released much more frequently than instrument papers.
^{1D}MESA2HYDRO^{3D} is threadsafe so long as the user does not point concurrent instances to the same output filename.
^{1D}MESA2HYDRO^{3D} evaluates the mass integral at each shell using RK4 (4th order Runge-Kutta, Runge 1895) with an adaptive step size, where the user provides a "best guess" for the initial integration step size, in cm (defaulting to a working value for the Sun if not otherwise specified). **A "good guess" will vary dramatically from problem to problem; see the tables here and Joyce et al. (2019) for suggestions.**
The efficiency of evaluation will change somewhat depending on the initial guess, as this is reset at each new shell. This method was selected over other schemes (e.g. **scipy.integrate**, Newton-Raphson) because of its flexibility in dealing with the high variance in scale in a stellar profile.
Faster integration times over restricted domains could certainly be achieved using pre-built integrators written in C or Fortran (or perhaps some of Python's own); our choice was designed primarily to deal with the fact that the distance scales covered in a single model star can vary, from shell to shell, by orders of magnitude, and thus allowing for large changes in step size within a given run is necessary to make computing times tractable. We are confident there are more sophisticated numerical techniques available to address this problem and would like to improve our methods—if you are familiar with any such solutions, please let us know.
The main algorithm implements a pseudorandom number generator via Python's **random.random**, a wrapper for the Mersenne Twister algorithm (Matsumoto & Nishimura, 1998), which uses the current timestamp as a seed.
^{1D}MESA2HYDRO^{3D} is not currently parallelized, but this is a goal of future releases.
In addition to standard, built-in Python libraries, the following Python 2.7 (or higher) packages are required by ^{1D}MESA2HYDRO^{3D}:

- argparse
- cython
- h5py (**optional:** only necessary if using HDF5 data format)
- HDF5lib (**optional:** only necessary if using HDF5 data format)
- healpy
- matplotlib.pyplot
- numpy
- python-tk
- random

- scipy.interpolate
- scipy.optimize
- tables (**optional:** only necessary if using HDF5 data format
- time

These can be installed from the command line via

```
sudo apt-get install python-tk
```

or similarly via, e.g., `pip install numpy`. We recommend that the user try to install these additional libraries via command line rather than through pip.
To run the test case, the user must have numerical and other libraries required by healpy/HEALPix

`https://healpy.readthedocs.io/en/latest/`

installed on their machine. The installation command for healpy is:

```
pip install healpy
```

More information on dealing with healpy can be found at: `https://healpy.readthedocs.io/en/latest/install`
Optionally, HDF5

`https://support.hdfgroup.org/HDF5/`

can be used. ^{1D}MESA2HYDRO^{3D} produces `phantom_binary` and ASCII output without involving HDF5, and if an HDF5 installation can be avoided, it should be.
If the user wishes to test their results using Phantom or another SPH code, they must install these codes separately. Phantom is available at

`https://bitbucket.org/danielprice/phantom/wiki/Home`

MESA itself requires mesasdk, the MESA software development kit. Installation instructions for this are available at

`http://mesa.sourceforge.net/`

This project was verified using MESA version 10398.
Lists of parameters that can be specified in MESA inlists are given in the "controls.defaults" and "star_job.defaults" files, which include variable names and short explanations of what each parameter controls physically. MESA output is controlled by two additional attribute lists called "history_columns.list" and "profile_columns.list," from which the user can specify the quantities for MESA to log throughout the model star's temporal evolution or as a function of radial depth, respectively. The MESA software suite and documentation provide glossaries of the available input physics options, as well as all of the calculated quantities the user can request. Parameters are categorized by which aspect of the model they affect, e.g. atmosphere, convective vs diffusive regions. These categories are indexed at the top of the list files, and variable naming strives to be somewhat intuitive. More information on MESA controls can be found in section 11.
This suite does not require an SPH code to run, but the user should use an SPH viewer, such as **SPLASH** (Price, 2007) or **gadgetviewer** (Helly, 2003), to verify properly the recovery of a density profile from SPH-compatible file types. Approximate verification is possible with the Python routines included with ^{1D}MESA2HYDRO^{3D}, but these do not invoke smoothing kernels and hence may be subject to normalization errors (see Price (2012)).

7 Installation

7.1 Via pip

^{1D}MESA2HYDRO^{3D} is installable via Python's **pip** tool. To install via pip, type into the command line:

```
  pip2 install MESA2HYDRO
```

To upgrade to a newer version, run

```
pip install --upgrade MESA2HYDRO==0.1.24
```

where 0.1.24 should be replaced with the lastest version number on

`https://pypi.org/manage/project/MESA2HYDRO/releases/`

Sometimes the error

```
ERROR: Cannot uninstall 'MESA2HYDRO'. It is a distutils installed project and thus we
cannot accurately determine which files belong to it which would lead to only a partial uninstall.
```

is triggered by an attempt to use `pip install --upgrade`. As a workaround, remove any current installation of MESA2HYDRO, followed by

```
  pip install MESA2HYDRO==0.1.24
```

or

```
  pip install MESA2HYDRO
```

for the most recent version.

7.2 Via GitHub

Alternatively, one may clone the git repository

```
  https://github.com/mjoyceGR/MESA2HYDRO
```

unpack it, and run the following from the top level directory:

```
  python2 setup.py install
```

to set up all dependencies manually. In this case, the user may have to install additional dependencies from the command line (`sudo apt-get`) or pip manually.

8 Setup

If the package has been installed via pip, pip will place the MESA2HYDRO home directory with other Python packages, most likely somewhere like

```
  /usr/share/bin/python2.7/
```

If the user has failed to specify pip2 and also has more recent versions of Python installed, it may end up somewhere like

```
~/anaconda/lib/python3.5/site-packages/
```

In any case, it is advised to copy or move the entire MESA2HYDRO root directory somewhere more sensible, since the user will need to work out of it directly:

```
mv MESA2HYDRO/ /home/your_name/
```

From there, one should set the `$MESA2HYDRO_ROOT` environment variable in their `.bashrc` to point to the root directory. `$MESA2HYDRO_ROOT` can also be set manually in `.bashrc`, or exported in the command line via:

```
export MESA2HYDRO_ROOT=/home/usr/YOUR_PATH_TO/MESA2HYDRO
```

If `$MESA2HYDRO_ROOT` is not set in `.bashrc`, this export command will need to be issued before operation every time a new terminal is opened.
If planning to use ^{1D}MESA2HYDRO^{3D} to interface with MESA directly, one should also set

```
export MESASDK_ROOT=/home/usr/mesasdk
export MESA_DIR=/home/usr/mesa-r10398
export OP_NUM_THREADS=8
```

in `.bashrc`.
Then,

```
cd $MESA2HYDRO_ROOT/
```

If one has installed ^{1D}MESA2HYDRO^{3D} via pip, no additional set up should be necessary.
If one has installed the package via tarball, direct downlod, or GitHub, ^{1D}MESA2HYDRO^{3D} can be configured by running

```
python setup.py install
```

in the MESA2HYDRO directory.

9 Running a Test Case

To successfully run a basic MESA2HYDRO instance, the user must have, **at minimum**, the following Python and external packages installed on their machine:

```
argparse
cython
healpy
matplotlib.pyplot
numpy
python-tk
random
scipy.interpolate
scipy.optimize
time
```

If they were not automatically installed via pip or the setup procedure, these can be installed from the command line via, e.g.,

```
sudo apt-get install python-tk
```

or similarly via, e.g.,

```
pip install numpy
```

The authors have had better results installing these additional libraries via command line rather than through pip.
In particular, one must have numerical and other libraries required by healpy/HEALPix (`https://healpy.readthe` installed on their machine. The installation command for healpy is:

```
pip install healpy
```

More information on dealing with healpy can be found at `https://healpy.readthedocs.io/en/latest/install.`

9.1 Basic Operation

A basic execution of ^{1D}MESA2HYDRO^{3D} is issued via the following command in the `MESA2HYDRO/work/` repository

```
./run.py --config-file mainsequence.cfg
```

or simply

```
./run.py  mainsequence.cfg
```

This creates an instance of ^{1D}MESA2HYDRO^{3D} for a solar-like main sequence star, with parameters from the "mainsequence.cfg" namelist (see 4.3 for discussion of configuration files) using pre-generated MESA profile data in the example format.

9.2 Using run.py

The script **run.py** contains all actions required to obtain a quick-and-dirty (i.e. computed without a smoothing kernel) density profile recovery from an input MESA profile file. Using routines encapsulated in `/lib/mainlib.py`, the **run.py** script performs the following actions after parsing either a configuration file or set of command line arguments:

1. **check_MESA_profile** generates a pop-up figure of the loaded MESA profile data. Close this to proceed.
2. **make_NR_file** generates an NR data file by numerically integrating regions of the loaded MESA density profile to determine sets of bounding radii $r_{\text{lower}}, r_{\text{upper}}$ that correspond to fixed mass per shell. This mass is determined by the user-specified number of particles per shell (N) and mass per particle (m_p). Fourth-order Runge-Kutta integration is used to obtain shell mass from the MESA density profile. The internal energy of the model star at the MESA radius $r_{\text{mid}} = (r_{\text{lower}} + r_{\text{upper}})/2$ is tracked as a fourth parameter in the NR file.

   Because the MESA data are discrete, linear interpolation is used to connect nearest neighbors. The resulting file contains sets of N values, radial midpoints, and shell masses that serve as input conditions to the HEALPix algorithm. Internal energy is tracked for use in constructing the SPH IC files; any other quantity may be tracked this way with minor modification. The number of entries in this file is the number of shells which will be used to generate the 3D distribution. An NR file producing smooth recovery may take up to a few hours to generate. The time it takes to achieve reasonable agreement varies with star type and desired solution tolerance. The ones provided in our sample took between 1 and 3 hours to generate (see Table 12).

3. **make_IC_file** loads the NR data generated in (1) into healpy (the Python interface to HEALPix) shell by shell, from which $12N^2$ sets of x, y, z coordinates are produced per entry. These values are iteratively concatenated to a global array. Each shell is rotated by an arbitrary angle θ and normalized by the placement radius given in the NR file. The resulting randomized, normalized array contains the set of particle positions corresponding to the entire radial span of the loaded MESA data. The resulting file should be an SPH-compatible IC file, written with data organization structures specified in the configuration file. For a reasonable NR file, this component should execute in a few seconds, maximum.

4. **try_reload** This gives a very rough indication of recovery by loading the IC file generated in (2) back into Python arrays containing the recovered radii and densities. Plots showing the loaded versus recovered profile in log and linear space are generated. A multiplicative offset (in linear space) manifests as a linear offset in the semilog plots, and can be caused by overshooting the target shell mass during numerical integration in (1)—this can be resolved by setting a lower solution tolerance on the mass integral. If this is the case, the recovered data may present slightly higher densities than the MESA data, but the curvature of the profile will be preserved.

10 Components

The suite installs by default as MESA2HYDRO. This will be the name of the parent repository, containing the *lib*, *work*, *data*, and *out* subdirectories. The *work* repository is where the user should operate the run.py script, the *lib* repository contains the subroutines (the bulk of the software), the *data* repository contains MESA control settings and sample input and output data, and the *out* repository contains sample output in the form that that a successful $^{1\mathrm{D}}$MESA2HYDRO$^{3\mathrm{D}}$ run should produce. This repository can also serve as a dump for data generated by the user.

10.1 MESA2HYDRO/DOCUMENTATION

This manual and the academic paper are stored here.

10.2 MESA2HYDRO/lib

The *lib* repository contains four original function libraries and tertiary components required by large data format reading and writing routines, as implemented by other authors. Our libraries are summarized as follows:

mainlib.py

This module contains the primary components of the workflow, including routines to generate the NR data, convert the NR data to large data format IC files, and load and examine the recovered density profile within python. Note that successful recovery in Python does not guarantee that the generated IC file will be compatible with Phantom, GADGET-2, or other IC readers—users must test their distributions with an SPH reader or hydrodynamics code, as we have done for all cases using Phantom.

converge_funcs.py

This module contains the numerical routines used to generate the NR data, including numerical integrators, routines which interface with HEALPix via healpy, and basic mathematical routines.

MESAhandling.py

This module contains MESA data handling routines, including construction of dictionaries from MESA output column headers, MESA inlist production, data selection by keyword, etc.

io_lib.py

This module contains routines to deal with the conversion of NR data to various initial conditions file formats. Many of the GADGET IC reading and writing routines are standard scripts which are publicly available. The origin and authors of scripts that are not the original work of Joyce et al. (2019) are noted in the comments.

cfg_parser.py

This module contains routines to parse command line arguments.
Other scripts in this directory are dependencies of the I/O handling routines included in io_lib.

10.3 MESA2HYDRO/work

The user should operate ^{1D}MESA2HYDRO^{3D} from this directory. The critical component is **run.py**, whose operation is detailed in 9.1. Some minor additional verification scripts are included.

confirm_density_profile.py

Generates a plot of density versus radius from a user-provided MESA profile via command line interaction.

confirm_mass_profile.py

Generates a plot of cumulative mass versus radius from a user-provided MESA profile via command line interaction.

11 External MESA data

The algorithm has been verified on a number of test MESA profiles, which encompass a range of model stars with a wide spectrum of physical properties. The MESA control files used to generate these specific MESA models are included with ^{1D}MESA2HYDRO^{3D} in the *data* directory. All MESA data used in the test suite were generated using MESA version 10398. All components necessary to reproduce the results in Joyce et al. (2019) are described as follows:

11.1 MESA2HYDRO/data

The subdirectory `MESA2HYDRO/data/MESA_controls/` contains files for MESA version 10398 which control output settings for profile and history data. These are specifically set to ^{1D}MESA2HYDRO^{3D}'s reading specifications. The files are

```
star_job.defaults
profile_columns_testsuite.list
profile_columns.list
history_columns_testsuite.list
history_columns.list
controls.defaults
```

- `history_columns_testsuite.list` dictates which physical quantities are retained by MESA throughout the star's temporal evolution, stored in a history.data file.
- `profile_columns_testsuite.list` dictates which physical quantities are stored in MESA's radial profiles, which are snapshots of the star's internal structure at a fixed time. These are stored in profile.data files. **These are the files that ^{1D}MESA2HYDRO^{3D} integrates to construct NR files.**

Use caution if designing your own `profile_columns.list` or `history_columns.list` files. So long as the essential parameter quantities (see Section 4.2) are tracked using the correct column keyword—as defined in `profile_columns.list`—alternative profiles should work, but proper operation is definitely not guaranteed.

- `MESA2HYDRO/data/inlists/` contains MESA inlist files corresponding to each validation case and additional MESA files (e.g. model `.mod` files) needed to run those cases.
- `MESA2HYDRO/data/sample_MESA_output/` contains the MESA profile and history output data used to generate our test suite (Note: ^{1D}MESA2HYDRO^{3D} does not make direct use of `history.data` files, but these are very useful for verification that the physics of your model makes sense)
- `MESA2HYDRO/data/config_file_examples/` contains example .cfg files for use with **run.py**

11.2 MESA2HYDRO/out

`MESA2HYDRO/data/sample_MESA_output/NR_files/` contains the NR data files created from the MESA profile data for each of the five test cases:

```
main sequence        NR_ms_TOL0.01_0.75R_mp1em7.dat
red giant            NR_rg_TOL0.01_0.75R_mp1em7.dat
extended supergiant  NR_OB_TOL0.01_0.75R_mp1em6.dat
TP-AGB               NR_agb_TOL0.01_0.75R_mp5em7.dat
white dwarf          NR_wd_TOL0.01_0.75R_mp1em7.dat
```

Any of these can be converted to an IC file directly with appropriate settings in the configuration file.

12 Test Suite

For information on the astrophysics of the test suite, we refer the user to the academic paper. Tables 12 and 12 from Joyce et al. (2019) provide quick references for viable parameter choices. They are reproduced here. Figures 4 through 8 in Joyce et al. (2019) are reproducible using between 22-30 bins for the ^{1D}MESA2HYDRO^{3D}-rendered particle data and the associated run-time parameters given here.

Table 3: $^{1\mathrm{D}}$MESA2HYDRO$^{3\mathrm{D}}$ Run Time Parameters and Recovery for Test Models

Star	N	m_p	$R_\star$	$M_\star$	$\Delta_{r,\text{initial}}$	N_{shells}	n_p	t_{gen}	σ_{rms}
Standard Solar	8	1×10^{-7}	0.75	0.0167	$1.00{\times}10^{8}$	441	338,688	1.311	0.018
Solar Red Giant	8	1×10^{-7}	0.75	0.0650	$1.00{\times}10^{8}$	852	654,336	4.039	0.022
Red Supergiant	8	1×10^{-6}	0.75	0.0034	$1.00{\times}10^{11}$	453	347,904	6.671	0.020
TP-AGB	8	5×10^{-7}	0.75	0.0449	$1.00{\times}10^{10}$	298	228,864	6.652	0.118
White Dwarf	8	1×10^{-7}	0.75	0.0330	$1.00{\times}10^{5}$	556	427,008	5.128	0.045

Computational and physical features of each test model are shown for $r_{\text{depth}} = 0.75R_\star$ and $\delta_{\text{TOL}} = 0.01$, corresponding to Figures 4 through 8 in Joyce et al. (2019). Particle masses are in units of $M_\odot$. N is the HEALPix integer. Initial step sizes are in physical units (cm). t_{gen} is the generation time for the NR coordinates, in hours. Generation times for IC files are negligible, typically on the order of 5 to 10 seconds.

Table 4: Parameters for $^{1\mathrm{D}}$MESA2HYDRO$^{3\mathrm{D}}$ Distributions Tested with Phantom

Star	m_p	N_p	$M_\star$ (g)	$R_\star$ (cm)	ρ_{avg} (g/cm^3)	t_{dyn} (s)	Phantom EOS
Standard Solar	10^{-5}	102,144	1.98×10^{33}	7.02×10^{10}	1.37×10^{0}	1.80×10^{3}	Adiabatic; $\gamma = 5/3$
Solar Red Giant	10^{-5}	102,144	1.98×10^{33}	2.40×10^{11}	3.46×10^{-2}	1.13×10^{4}	Adiabatic; $\gamma = 5/3$
Red Supergiant	10^{-4}	64,512	1.78×10^{35}	1.18×10^{14}	2.57×10^{-8}	1.31×10^{7}	Adiabatic; $\gamma = 5/3$
TP-AGB	10^{-5}	193,536	5.05×10^{33}	8.69×10^{12}	1.84×10^{-6}	1.55×10^{6}	Adiabatic; $\gamma = 5/3$
White Dwarf	10^{-5}	134,400	1.98×10^{33}	2.78×10^{8}	2.20×10^{7}	4.48×10^{-1}	Helmholtz

In all cases, $\delta_{\text{TOL}} = 0.01$. r_{depth} is approximately zero for all models, with the exception of the supergiant, which uses $r_{\text{depth}} = 0.35$ due to the difficulty of resolving the extreme densities in the stellar core using a particle mass that is suitable for the rest of the layers. When $r_{\text{depth}} = 0.0$ is used for the supergiant, the $^{1\mathrm{D}}$MESA2HYDRO$^{3\mathrm{D}}$ solution time jumps from approximately 1.6 hr to 19 hr, and the particle number increases from ≈ 65000 to more than 1 million.

13 Current Limitations and Known Issues

• The authors have encountered some bugs surrounding **cython**. In some cases, the setup procedure fails (segfault) if **cython** is installed via pip rather than via the command line directly (i.e. `sudo apt-get install cython`). The cause of this issue is currently unknown, but may be related to conflicting versions of Python. Purging **cython** and installing it from the command line has (sometimes) fixed this.

• Only Phantom ICs have actually been evolved and had their stability numerically assessed on the 3D side. Performing additional verification with other SPH codes would be very helpful! If 3D SPH is your expertise, do it!

• $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ is not currently parallelized

• $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ does not use a smoothing kernel in its built-in recovery tests

• $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ is written in Python 2.7 (yes, I know...) because development began a long time ago. Optimistically, it will be converted to Python 3 by the end of October 2019.

• $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ was developed on Linux machines using various iterations of Ubuntu. It has been successfully installed on a computing cluster and various laptops and Linux-running workstations. Its functionality in a Mac or Windows environment has not yet been fully explored.

14 How to Contribute

$^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ is publicly available and can be modified as the user sees fit. M Joyce will do her best to integrate any corrections or improvements in a timely manner. Some components of $^{\mathrm{1D}}$MESA2HYDRO$^{\mathrm{3D}}$ fall under the domain of the MESA collaboration, who maintain an active community of developers and participants.
For now, requests for incorporating additional functionality should be emailed to **Meridith.Joyce@anu.edu.au**
If you use this package or any of its components, please cite the paper!

14.1 Authors' Statement

This package is presented with the hope that it will help others do science.
Most of the authors are not professional software developers. In the spirit of collaboration and to avoid discouraging others—especially women and members of other groups underrepresented in software development—from sharing their code publicly, it is expected that all criticism, corrections, and suggestions for improvement will be made respectfully and in good faith.

References

Cordier, D., Pietrinferni, A., Cassisi, S., & Salaris, M. 2007, AJ, 133, 468

Demarque, P., Woo, J.-H., Kim, Y.-C., & Yi, S. K. 2004, ApJ Suppl., 155, 667

Dotter, A., Chaboyer, B., Jevremović, D., et al. 2008, ApJ Suppl., 178, 89

Górski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ, 622, 759

Helly, J. C. 2003, gadgetviewer

Joyce, M., Lairmore, L., Price, D. J., Mohamed, S., & Reichardt, T. 2019, The Astrophysical Journal, 882, 63

Matsumoto, M., & Nishimura, T. 1998, ACM: Transactions on Modeling and Computer Simulation, 8, 3

Paxton, B., Marchant, P., Schwab, J., et al. 2015, ApJ Suppl., 220, 15

Pietrinferni, A., Cassisi, S., Salaris, M., & Castelli, F. 2004, ApJ, 612, 168

Price, D. J. 2007, PASA, 24, 159

—. 2012, Journal of Computational Physics, 231, 759

Price, D. J., Wurster, J., Tricco, T. S., et al. 2018, PASA, 35, e031

Runge, C. 1895, Mathematische Annalen, 46, 167

VandenBerg, D. A., Bergbusch, P. A., & Dowler, P. D. 2006, ApJ Suppl., 162, 375