MESA2GADGET: BRIDGING THE GAP BETWEEN 1-D AND 3-D STELLAR MODELS

M. Joyce$^{1,2}$, S. Mohamed$^{2,3}$, AND L. Lairmore$^4$

$^1$Department of Physics and Astronomy, Dartmouth College, Hanover, NH 03755
$^2$South African Astronomical Observatory, Observatory Road, Cape Town, South Africa
$^3$University of Cape Town, Rondebosch, Cape Town, South Africa and
$^4$Robotics Engineering Department, KeyMe, New York, NY, 10005

Abstract

Thermal pulsation is a complicated mixing event in AGB stars which involves interaction between several layers of the stellar interior, and which occurs on very short timescales compared to a star’s evolutionary timescale. The temporal constraint makes these events difficult to resolve in a 1-D stellar evolution codes, and instead they are typically studied in isolation through 3-D smoothed-particle hydrodynamical simulations. However, in their 2015 instrument paper, Paxton et al. demonstrated that the 1-D MESA (Modules for Experiments in Stellar Astrophysics) stellar evolution code could produce robust interior profiles of thermally pulsating AGB stars.

Following on from investigations done this year involving MESA models of massive stars, we present “MESA2GADGET,” a Python-based toolkit for automating the projection of 1-D stellar density profiles to 3-D particle distributions for use in smoothed-particle hydrodynamics simulations. The package facilitates the conversion of 1-D models of the interior structures of stars during exotic and time-sensitive evolutionary phases to initial conditions in 3-D codes, and establishes an accessible foundation for prescribing empirically informed, rather than ad hoc, evolutionary initial conditions for high-resolution stellar simulations.

Subject headings: computational methods: algorithms, workflows—stars: evolution, AGB, pulsation

1. INTRODUCTION

The wide use and versatility of synthetic stellar data in astrophysics make the accuracy of stellar calculations a protracted concern among astronomers. The maintenance, development, and validation of high-quality, theoretical stellar models gain particular urgency in light of the growing scope and accuracy of observational knowledge. It is thus increasingly important that we are able to characterize quantitatively the discrepancies between observations and stellar evolution calculations, and to improve the treatment of the formalisms they invoke through a thorough understanding of the impact of microphysical specifications on synthetic data across a range of evolutionary phases.

The DSEP (Dartmouth Stellar Evolution Program) code is a tool that has served these initiatives historically. The isochrones and stellar tracks produced with this code have provided powerful priors, boundary conditions, and data for a plethora of astronomical problems. In recent months, DSEP has reproduced observed stellar magnitudes to 1% accuracy, and the full extent of its predictive capacity is currently being investigated via Red Giant Branch Bump (RGBB) brightness estimates. DSEP’s ability to reproduce accurate RGBB parameters is especially noteworthy given that this feature is a direct, observable consequence of a mixing interaction between the convective and radiative zones in the interior of a star. Through such reproductions, we are pushing the boundary on constraining a highly nonlinear, turbulent mixing problem under a computationally tractable formalism.

While DSEP has served the astronomical community well for many years and continues to produce novel science, the future of 1-D stellar evolution models is the MESA (Modules for Experiments in Stellar Astrophysics) code. MESA is a suite of open source, thread-safe libraries developed in Fortran 95. It contains distinct modules for handling the equations of state, opacity, nuclear reaction rates, diffusion data, and atmospheric boundary conditions. Each module is a separate library with its own public interface, supports shared memory parallelism based on the OpenMP application program interface, and employs adaptive mesh refinement and time step control. Extensive testing of MESA indicates that the code accurately calculates the evolution of stars over a wide range of masses (from brown dwarfs to $M = 100 M_\odot$), and over evolutionary phases spanning from the pre-main sequence to the onset of core collapse in high-mass stars, or to the white dwarf stage in lower-mass stars.

Astronomers are only now entering an era with the computational resources to treat microphysical processes, such as nucleosynthesis and elemental diffusion, with sufficient accuracy to affect synthetic observables produced by stellar evolution codes. Between MESA and DSEP, we now have access to evolutionary models that include elemental diffusion, gravitational settling, fine-tuning of mixing events, and elaborate meshes for specifying the initial abundances of over 50 elements individually.

In their second instrument paper, Paxton et al. (2013) present MESA’s reproduction of stars through the end of their core He-burning and thermal pulsation phases. They introduce a treatment of radiation-dominated envelopes that allows the uninterrupted evolution of massive stars to core collapse, thus enabling the production of progenitor models for a variety of stellar deaths, including supernovae. In Paxton et al. (2015), the group
further unveils the ability to coevolve an interacting pair of stars undergoing mass transfer and angular momentum loss, with improvements in the modeling of advanced burning stages. The consequence of these capabilities in aggregate is the ability to model an entire massive-star life cycle, from pre-main sequence to core collapse, all within the purview of 1-D physics.

These capabilities have been duly recognized by the 1-D stellar modeling community, and MESA is widely considered to be the state-of-the-art among evolutionary codes. In addition, its vast array of user-specifiable control parameters makes it highly customizable and hence adaptable to a wide array of problems. Among these parameters are those that control metallicity, opacity, and the presence and strength dust-driven winds (DDW). Tashibu et al. (2017) investigate the impact of these evolutionary parameters on dust formation and mass loss around asymptotic giant branch (AGB) stars. More typically, however, the dynamic behavior of AGB stars is studied in 3-D environments, often using smoothed-particle hydrodynamics (SPH).

The question of AGB stars’ role in generating dust and enhancing the interstellar medium (ISM) with heavy elements is long-standing, and of course predates Tashibu et al. (2017)’s work. About fifteen years ago, Steffen & Schönberner (2000) posited the problem of describing the existence of thin, detached gas shells around AGB stars. Though they were able to construct physically reasonable models using a combination of two-wind interactions and short-episode periods of high mass loss following the helium flash, they concluded that improved stellar calculations—ones that could address mixing events, mass loss, pulsation, and the helium shell phase with considerably more accuracy than the calculations used at the time—could mitigate the need for some contrivance in their explanation. All of these processes can be “turned on” in MESA. Using stellar priors generated with MESA, we can determine the impact of chemical enhancement and improved treatment of mixing events, such as the dredge-up and helium flashes, on these stellar environments.

More recently, MESA has also attracted the attention of other stellar astrophysicists who have recognized its potential for integration with higher dimensional stellar simulations. This can be seen, for instance, in the work of Ohlmann et al. (2017), who use 1-D density profiles generated with MESA to inform 3-D hydrodynamical models of red giants.

Movi\v{b}ed by these investigations, we were inspired to consider two problems. Firstly, we are interested in the impact of high-fidelity, stellar evolutionary boundary conditions for 3-D SPH models of AGB stars. Secondly, and necessitated by the first, we would like to address the general problem of interfacing 1-D stellar evolution models with 3-D SPH models of stars.

2. THERMALLY PULSATING ASYMPTOTIC GIANT BRANCH STARS

Figure 1 shows an HR diagram of the evolutionary track of a 2.5 $M_\odot$ star, from the pre-main sequence to the tip of the asymptotic giant branch (AGB). MESA version 8118 was used to construct this model.

During the pre-main sequence (Pre-MS), the proto-stellar cloud undergoes gravitational collapse until core hydrogen burning is initiated. Following this, the star settles into the main sequence (MS), which can last anywhere from $10^7$ to $10^{10}$ years, depending on the initial mass of the star. This is followed by the main sequence turn off (MSTO) and ascent onto the sub giant and red giant branches. At this point, prior hydrogen burning has generated a core of helium ash, and burning continues in a hydrogen shell surrounding the core. Following this comes the ignition of helium burning in the core, which takes place along the early asymptotic giant branch (EAGB). This phase mimics the RGB phase, but includes an additional layer of shell burning. Following this, the star ascends onto the thermally pulsating asymptotic giant branch (TP-AGB).

The TP-AGB phase of a stars evolution is a late stage characterized by instability, and occurs only for stars below about $10^6 M_\odot$. During this phase, a low-mass star’s interior structure consists of a central and largely inert core of carbon and oxygen, surrounded by a helium-to-carbon burning shell, surrounded then by hydrogen-to-helium burning shell, and surrounded lastly by a large convective envelope. The “thermal pulsation” component refers to dynamic activity that involves interaction between several of these layers and pulsations which induce mixing.

Thermal pulsations are driven by thermal run-away when the helium shell becomes unstable. During these helium shell flashes, the helium shell luminosity increases dramatically, causing a sudden release of energy that creates a convective region between the helium and hydrogen layers. This, in turn, causes expansion which results in the extinguishing of the hydrogen burning shell. This extinction is followed by a drop in surface luminosity, then a release of energy from the core which increases the surface luminosity to its cyclical maximum. Following this, the star contracts, which reignites the hydrogen burning shell and reinstates hydrogen burning as the main source of luminosity. The process repeats for about $10^6$ years, with the time between thermal pulses lasting anywhere from a few thousand to a few hundred thousand years, depending on stellar mass.

Figure 2 shows MESA-generated models of thermal pulses for a 1.8$M_\odot$ star. The pulsations are shown as a function of stellar age for four quantities: effective tem-
MESA MODELS AS INITIAL CONDITIONS

MESA is operated by specifying the initial conditions of a model star via a central control file called an “inlist.” This contains modeling, physical, and plotting controls. All possible parameters which can be specified in the inlists are given in “defaults” files, which also include short explanations per parameter. The output is controlled by two “columns” lists, “history” and “profile,” in which the user can specify the quantities for MESA to track (1) throughout the model star’s temporal evolution or (2) as a function of radial depth at a given time, respectively. The “control” and “star_job” lists serve as glossaries of the available input physics options, and the history_columns list file serves as an index for all of the calculated information you can request from MESA. The variable naming strives to be somewhat intuitive.

MESA’s profile files store the state of the star, from core to surface, as a function of radius (or equivalently, mass contained) at a particular time. The user can control the frequency with which these configurations are captured and printed to files throughout an evolutionary run. The history file, on the other hand, tracks the evolution of the physical state variables, e.g. density, pressure, temperature, luminosity, etc—the dependent quantities in the standard stellar structure equations—over time. This is only “complete” at the end of the model’s run. The history file and its analog in other 1-D stellar evolution codes contains the data typically associated with a stellar structure and polytrope solvers, and is not especially difficult to produce compared to an accurate profile.

Because MESA stores highly accurate instances of the model star’s interior structure—similar to what GADGET users would call “snapshots”—at specified time intervals, we can recover the interior configuration of the model star during any evolutionary phase; if the code is robust enough to resolve thermal pulses, the structural data can be extracted at this point. Hence, to obtain the interior configuration of a TP-AGB star, we can simply generate a profile during the TP-AGB phase.

This feature is significant for the purpose of 1-D to 3-D interfacing because it allows for the recovery of density profiles which are empirically informed, thus alleviating the need to assume a 1/r density profile ad hoc in the construction of a 3-D stellar model. While not fully empirical, an evolutionary model tailored to the observable features of a specific AGB system, such as TT Cyg or R Scl, is able to capture the physics of actual stars with more fidelity, and is at least subject to differences in initial mass, elemental abundances, or a host of other variables, that could manifest in the density profile.

This method is especially powerful given the unprecedented customizability of MESA. With a routine system for converting MESA density profiles to SPH initial conditions, we have a way to encode and account for any of the ~10,000 possible stellar evolutionary initial conditions governing the model star into a 3-D particle distribution. With this tool, parameter studies can be functionally automated. And, although our scientific motivation concerns AGB stars specifically, such an interface is readily adaptable to other evolutionary phases.

4. METHOD

In developing the interface between 1-D MESA models and SPH initial conditions, we invoke roughly the same physical approximations as in [Ohlmann et al. (2017)], who use MESA in a similar way to inform 3-D models of red giants. We note that they use MESA version 6208 for thier models, whereas we use MESA version 9575.

In their construction, they use only a small percentage of MESA’s total density profile in order to represent the particle distribution associated only with the outer layers of the stellar atmosphere. The rest of the interior is represented as a single particle with a mass equal to the remaining stellar mass. This is necessary in order to avoid using particle numbers associated with authentic stellar densities. This representation, while approximate, is computationally tractable yet sufficient to preserve the region which exhibits the dynamic behavior. We use the term “depth cut” subsequently to refer to the percentage of the star, in terms of total mass, to which we fit the density profile. Setting an e.g. 5% depth cut refers to fitting the region containing the 95th mass percentile and outward (towards the surface) only.

The general method for conversion from 1-D stellar density profile to 3-D particle distribution proceeds as follows:

- We first extract the density profile ρ(r) corresponding to the outer ~5% of the mass distribution from a MESA snapshot generated during the model star’s AGB phase.
• We subdivide the profile into \( k \) regions of variable size \((r_u - r_l)\) such that the number of particles \( N \) (equivalently, the mass per particle \( m_p \)) is preserved across all regions \( k \). To maintain constancy in either of these variables, the mass per shell \( m_{\text{shell}} \) must be fixed. Since the mass contained per region \( k \) is a function of the width of \((r_u - r_l)\) via the density integral \( m_{\text{shell}} = \int_{r_l}^{r_u} 4\pi r^2 \rho(r) \, dr \), computing \( m_{\text{shell}} \) for each region requires either numerical integration or model fitting. The bottom panel of Figure 3 demonstrates the selection of a region \( r_u - r_l \) from the density profile.

• We distribute the \( N \) particles \( (m_{\text{shell}}/m_p) \) corresponding to the mass contained in each region \( k \) across the surface of a sphere with radius \( r_{\text{mid},k} = \sqrt{(u_k - l_k)} \). Care is required in selecting a particle distribution method that will minimize computational artifacts; this is detailed thoroughly in section XX. An example of one such shell is shown in Figure 4.

• Having computed the appropriate radii \( r_{l,k}, r_{u,k} \) for each shell and obtained \( N \) sets of \( x, y, z \) coordinates for the associated particles, we stack the shells concentrically to form a 3-dimensional, hollow sphere representing the particle distribution of an AGB star’s atmosphere. We must ensure that we arbitrarily rotate each shell with respect to its neighbors in order to avoid ordered particle alignments.

• The final set of \( k \) stacked, rotated, concentric sets of \( x, y, z \) coordinates constitute the desired particle distribution. These complete \( x, y, z \) arrays are what GADGET2 and similar codes understand as initial conditions.

• It is easy and necessary to validate our distribution. We may generate an initial conditions file from our coordinate arrays and load these data directly. We ensure that \( \rho(r_k) \) vs \( r_k = \sqrt{x^2 + y^2 + z^2} \) vs \( k \) is in strict agreement with the density profile originally collected from MESA.

Ohlmann et al. (2017) have demonstrated that this method is effective for depth cuts ranging from 1 to 10% in constructing their 3-D models.

4.1. Surface Particle Distribution and Shell Placement

We follow in large part the mathematical techniques of Pakmor et al. (2012) to construct the surface particle distribution per shell and to place the shells in their final orientation.

To construct the surface particle distribution, we first replicate Pakmor et al. (2012)’s use of the HEALPix, or the Hierarchical Equal Area iso-Latitude Pixelization, algorithm for the tessellation of the surface of a sphere (Górski et al. 2005). In constructing initial conditions for SPH simulations, it is important that the particle configuration does not give way to any preferred directionality once evolved; otherwise, the resulting simulation will be subject to computational artifacts with no physical basis. The HEALPix method is a canonical choice among distribution methods, providing both enough randomness and enough separation between the particles to minimize non-physical effects.

HEALPix works by subdividing the surface of a sphere into \( n_p \) quadrilateral regions, or cells, of equal area and placing exactly one particle in each cell. All that HEALPix requires to generate the desired set of particle coordinates \((x, y, z)\) is a number of particles, or equivalently, cells, \( n_p \). However, \( n_p \) must satisfy some fairly restrictive mathematical properties in order for the tessellation to work; namely, \( n_p \) must be such that \( n_p = 12x^2 \), with \( N = 2^x \) for some integer \( x \). To access the HEALPix algorithm, our workflow uses the Python interface healpy, developed by CITE. Figure 5 shows an example tessellation for \( n_p = 3072 \), or equivalently \( N = 16 \). Regions in this figure are labeled with an integer index beginning at 0.

The appropriate spacing among HEALPix shells is determined by the calculation of placement radii \( r_{\text{mid},k} = \sqrt{(u_k - l_k)} \).
Fig. 5.— Heirarchical equal area iso-latitude pixelization of a spherical surface for $N = 16$, generated with healpy.

\[ (r_{u,k} + r_{l,k})/2. \]

The widths of slices $r_{u,k} - r_{l,k}$ are dictated by the imposition of constant $n_p$, and by proxy, $m_{\text{shell}}$. As mentioned in the previous section, calculation of the $r_u$ values corresponding to the desired $m_{\text{shell}}$ requires evaluation of the density integral. This evaluation can proceed in two ways.

4.1.1. Analytic Fit vs Numerical Integration

Because we are modeling a relatively smooth section of the profile (see Figure 3), we can obtain a well-fitting, closed form expression for $m_{\text{shell}}$ from the function

\[ m_{\text{shell}} = \int_{r_1}^{r_u} 4\pi r^2 \rho(r) \]

by assuming a functional form for $\rho(r)$ of

\[ \rho(r) = A \left( \frac{1}{(r - B)} \right) + C \]

and applying standard optimization techniques. In this case, we use Python’s scipipy.optimize curve fitting package to compute values for fitting parameters $A$, $B$, and $C$. The resulting anti-derivative

\[ m_{\text{shell}}(r_u) = 4\pi \left\{ \frac{1}{2} A (r_u - r_l)^2 
+ AB (r_u - r_l) + AB^2 \ln (r_u - r_l) - B 
+ \frac{1}{4} C (r_u - r_l)^3 \right\} \]

is then an explicit function of known quantities. Calculation of $m_{\text{shell}}$ at this point is straightforward.

This method is fast computationally and removed from any limitations due to discrete radial sampling within the MESA profile data, but it involves an ad hoc assumption similar to the one this project was conceived to avoid. It likewise involves initial guesses for the fitting parameters $A$, $B$, and $C$. Defaults in the software are assigned as follows:

\[ A = 10.0^{-6} \quad (4a) \]
\[ B = 0.68 \quad (4b) \]
\[ C = 1.0 \quad (4c) \]

or can be specified by the user. Either way, these assignments are physically arbitrary and invoke external assumptions that should be avoided if possible.

To avoid this, we can also solve the density integral in the $m_{\text{shell}}$ expression numerically, using an arbitrarily small radial step size specified by the user. In this case, we are assigning MESA’s true value of $\rho$ at the desired $r_u$—or at least a value of $\rho$ as near to $\rho(r_u)$ as the discrete data will allow—rather than computing it independently. This better achieves our goal of physical fidelity, but the downsides to this method include its relatively longer run time as well as its dependence on MESA’s density sampling. The numerical technique has been validated over step sizes spanning from $10^{-3} = 10^{-6}$.

The numerical subroutines module in MESA2GADGET supports both of the outlined methods, and both have been validated on three AGB profiles. We recognize the potential also to adapt the analytical fitting method to serve as a means of local interpolation MESA grid points. This would provide a technique which is a hybrid of the current two options.

4.1.2. Convergence Method

Having now presented the means of obtaining $m_{\text{shell}}$ as a function of $r_u$, we delve into the details of selecting $r_u$.

We follow once again the procedure of Pakmor et al. (2012). The crux of their method is to impose two independent, geometric constraints and force their equality. Since we are searching for the width of a radial slice which optimizes certain mass values, we define two functions, $n_1$ and $n_2$, of $r_u$. Recall that HEALPix tessellates the surface of a sphere into $12n^2$ quadrilaterals for $n \in \{2^x\}; n, x \in \mathbb{Z}$. Let $n_1(r_u)$ be such that $n_p = \frac{m_{\text{shell}}}{m_p}$, and $m_{\text{shell}} = m_{\text{shell}}(r_u, r_l)$. Then,

\[ n_1(r_u) = \sqrt{\frac{m_{\text{shell}}(r_u, r_l)}{12n_p}}. \]

Now, let each quadrilateral piece on the tessellated surface have width $r_u - r_l$. The surface area of that quadrilateral is then $(r_u - r_l)^2$. Concurrently, the total surface area of a spherical shell $k$ is given by $S_{\text{tot}} = 4\pi r^2$. From our midpoint definition, this is equivalent to $S_{\text{tot}} = 4\pi r_u^2$. From HEALPix, we also have the requirement that there exist exactly one particle per quadrilateral, or $12n^2$ particles and $12n^2$ quadrilaterals. From this follows the equality $(r_u - r_l)^2 12n^2 = 4\pi \left( \frac{r_u + r_l}{2} \right)^2$. The resulting expression for $n_2$ is

\[ n_2(r_u) = \sqrt{\frac{\pi}{12}} \frac{r_u + r_l}{r_u - r_l}. \]

Because $n_1$ is a monotonically increasing function of $r_u$ and is zero-valued when $r_u = r_l$, and $n_2$ is monotonically decreasing and infinitely valued when $r_u = r_l$, there exists some value of $r_u$ for which $n_1 = n_2$. Depending on how $r_u$ is obtained (i.e. integration method), we will not necessarily recover the exact value, so it is sufficient numerically to search for the first test value of $r_u$ at which $n_1 > n_2$. Hence, $r_u$ is set via the requirement that $N = n_p = 12n_1^2$ when $n_1 \approx n_2$. Once $r_{u,k}$ is set
for shell \( k \), \( r_{l,k+1} \) is assigned to \( r_u \). The process repeats until the density drops to zero. Figure 6 demonstrates the iterative search for these \( r_u \) values, propagating from the most interior region of the star outward. Points where the green line intersects the magenta curve mark the \( r_u \) values at which crossover between \( n_1(r_u) \) and \( n_2(r_u) \) occurred. Since we aim to preserve \( N = n_p \), it is expected that the value of convergence for \( n_1 \) and \( n_2 \) remain constant across each shell. As we can see, it does not quite do so. This is because we can only satisfy \( n_1 > n_2 \), rather than the ideal case of \( n_1 = n_2 \), with discrete \( r_u \) values. Constancy in \( N \) can be better approximated with decreasing step size (and increasing computation time).

Figure 6 also demonstrates that the widths of the radial shells increase with increasing \( r_u \). This is what we should expect from a density function which decreases towards the stellar surface; we require cross sections of larger physical size to maintain constant particle number in lower density regions.

[Ohlmann et al. (2017)]'s method differs slightly from that of Pakmor et al. (2012) in that they do not invoke a cubical volume constraint (e.g., a distance between shells of \( r_u - r_l \)). Because [Ohlmann et al. (2017)] designed their distributions to work with AREPO (Springel 2011), equal-mass cells were not required. Our method does invoke the assumption of a fixed particle mass \( n_p \), but routines which do not require this constraint are in development and readily integrable with our framework.

5. RESULTS

The complete MESA2GADGET workflow encompasses the codes, subroutines, and linkings shown in Figure 7. Components represented in dark blue are the base codes: MESA and 3DESI-GADGET, or 3D Evolve Stellar Interact-GADGET, a 3-D SPH GADGET derivative designed specifically for modeling stellar behavior. Components given in light blue are externally composed software packages with which the code authored here interfaces. These include healpy, the Python-based interface to HEALPix; h5py and pygadgetreader (Thompson 2014), Python interpreters for HDF5 (a large data file type commonly used in SPH codes) and GADGET, respectively, and the HEALPix and the HDF5 libraries themselves. Components shown in green are the work of the authors exclusively. These modules include:

- **MESAlibjoyce**, a library for processing MESA data and MESA control files;
- **convergefuncs**, the module containing the numerical routines;
- **make_hdf5**, the interface which translates HEALPix shells to the file format understood by GADGET; this is available in either HDF5 or binary formats, though generating binary data does not require the installation of special libraries;
- **validateanalytic**, a module containing a sample implementation of the analytic method for computing shell radii \( r_u \); and
- **validatenumeric**, a module containing a sample implementation of the numerical method for computing shell radii \( r_u \).

One complete operating cycle of MESA2GADGET proceeds as follows: The MESA AGB density profile is generated by and post-processed with the handling routines in our MESA library. The extracted data are reduced to a set of \( k \) HEALPix parameters \( N, r_u \) via the methods contained in our numerical module. We use healpy to generate sets of \( x, y, z \) coordinates corresponding to each shell, and write the total \( k \) coordinate sets to an HDF5 file. This file is read by the SPH code, which builds final configuration. To ensure the desired profile was preserved throughout these processes, we back-project the 3-D distribution into a 1-D radial curve and compare this to the data generated initially. When a GADGET-ready initial conditions file is loaded into an SPH reader, such as SPLASH (Price 2011) or gadgetviewer (7), a successfully translated profile will produce a model similar to those shown in Figure 6. The top panel shows the final particle distribution; the bottom panel shows the same, but with a smoothed density projection applied. Color coding reflects density. Note that in the case of the smoothed distribution, the density coloring reflects the presence of the large point mass we have placed at the center of the star.
We have validated the density profile translations across a small range of post-main sequence stellar phases, a range of particle masses spanning $10^{-5}$ to $10^{-12}M_\odot$, a range of convergence $N$ values spanning 8 to 256 (though GADGET and similar codes cannot be expected to deal with $N$ values above 16, and this is degenerate with $m_p$), a range of step sizes spanning 1 to $10^{-8}$, a range of atmospheric penetration depths spanning 1% to 15%, and across the two integration techniques described in section XX. Figure 9 shows the results of two test cases using the analytical method.

The figures show four density curves as a function of radius. In small blue dots are the discrete data sampled directly from the MESA density profile. The thick, yellow-green solid line represents the analytical fit to those data via the closed form integral solution. The dashed black line shows the projected density function we should expect to recover from the 3-D particle distribution that comes directly from the HEALPix portion of the workflow. This curve is generated by passing $r = \sqrt{x^2 + y^2 + z^2}$ for HEALPix coordinates $(x, y, z)$ through the analytic function used to fit the raw MESA data, and is intended as a sanity check on the translation of the initial density profile up to this point in the routine. It is indistinguishable from the analytic fit, which is what we should expect if the density profile is well-preserved at this stage. Lastly, the large magenta dots represent the density values at each shell radius $r_{mid,k}$ recovered from loading the data from GADGET itself, after the initial conditions file has been read and interpreted by the code.

In the upper panel, we see that the final density profile recovered from GADGET does deviate from both the initial MESA radial distribution and the analytic fit, but the recovery is reasonably accurate. The accuracy is improved in the lower panel, where the sampling rate is increased considerably via a decrease in step size.

As is typically the case, the user must balance resolution against computation speed. Though an arbitrary degree of smoothness is theoretically possible using the analytic fit, note that key regions of divergence in the bottom panel of Figure 9 have not been significantly improved compared to the upper panel. Fundamentally, the analytical method is limited by the goodness of fit of the initial density function, and not by the resolution of the radial data used in the integral.

Figure 10 shows the results of validation tests using the numerical method. The small blue dots and thick, yellow-green solid line again represent the MESA data and analytic fit, respectively. The magenta dots likewise represent the density values recovered from loading the GADGET data directly, but in this case, the density values correspond to those collected from the initial MESA profile. The agreement between the MESA profile and GADGET recovery thereof is on par with the agreement found between the analytic fit and its GADGET recovery. Once again, decreases in both the particle mass and step size correspond to improvements in the profile recovery.

Table 1 provides values for key parameters used to generate the test models shown in Figures 9 and 10 as well as for other validation tests which are not shown. As the table more clearly demonstrates, computation time is most sensitively a function of step size in the analytic case, but imposing a very small particle mass $m_p$ can also increase the run time considerably. In the numeric case, it is lowering the particle mass which has the most significant impact on computation time.

### 6. SUMMARY AND FUTURE WORK

We have developed, tested, and presented an algorithm which facilitates a largely automated translation of 1-D density profiles generated with the stellar evolution code MESA to 3-D particle distributions which can be understood by SPH codes. We have verified the functionality of two profile translation methods and run a cursory analysis of the performance of the algorithm. We conclude that this method is very effective for preserving the semi-empirical data generated with MESA and fits thereof, and that it is computationally robust over the parameter space considered. Though our subsequent investigations will concern AGB stars most immediately, the scientific applicability of this tool is broad. The development of this algorithm lays critical groundwork for future enhancements, especially the implementation of e.g. additional integration and data interpolation methods. The potential for contribution to and refinement of the numerical methods used in MESA2GADGET is large.

On the scientific front, we aim to adapt this interface...
TABLE 1

<table>
<thead>
<tr>
<th>Method</th>
<th>$m_p$</th>
<th>Depth Cut</th>
<th>Step Size</th>
<th>Time (s)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytic</td>
<td>10$^{-6}$</td>
<td>5%</td>
<td>10$^{-6}$</td>
<td>8.2</td>
<td>shown in upper panel of 9</td>
</tr>
<tr>
<td>Analytic</td>
<td>10$^{-6}$</td>
<td>5%</td>
<td>10$^{-6}$</td>
<td>54.4</td>
<td>not shown</td>
</tr>
<tr>
<td>Analytic</td>
<td>10$^{-7}$</td>
<td>5%</td>
<td>10$^{-6}$</td>
<td>515.3</td>
<td>not shown</td>
</tr>
<tr>
<td>Analytic</td>
<td>10$^{-9}$</td>
<td>5%</td>
<td>10$^{-6}$</td>
<td>62.7</td>
<td>not shown</td>
</tr>
<tr>
<td>Analytic</td>
<td>10$^{-12}$</td>
<td>5%</td>
<td>10$^{-3}$</td>
<td>152.5</td>
<td>shown in lower panel of 6</td>
</tr>
<tr>
<td>Numeric</td>
<td>10$^{-6}$</td>
<td>5%</td>
<td>10$^{-3}$</td>
<td>728.2</td>
<td>not shown</td>
</tr>
<tr>
<td>Numeric</td>
<td>10$^{-8}$</td>
<td>5%</td>
<td>10$^{-3}$</td>
<td>20.6</td>
<td>not shown</td>
</tr>
<tr>
<td>Numeric</td>
<td>10$^{-12}$</td>
<td>5%</td>
<td>10$^{-6}$</td>
<td>763.4</td>
<td>not shown</td>
</tr>
<tr>
<td>Numeric</td>
<td>10$^{-8}$</td>
<td>5%</td>
<td>10$^{-3}$</td>
<td>76.4</td>
<td>shown in the lower panel of ??</td>
</tr>
</tbody>
</table>

Note. Parameter values associated with validation runs; see Figures 9 and 10. "Time" refers to computation time, in seconds. "Depth cut" refers to the depth of penetration from the surface into the stellar interior, in terms of mass.

Fig. 9.— Validation of the analytic method using two sets of parameters (top and bottom), as indicated in Table 1.

Fig. 10.— Validation of the numerical method using two sets of parameters (top and bottom), as indicated in Table 1.

to (1) investigate the impact of initial metallicity on the features of thermal pulsations along the asymptotic giant branch; (2) investigate the impact of metallicity on late stage stellar mass loss rates; and, crucially, (3) to investigate the formation, or lack, of detached gas shells around AGB stars.

7. ACKNOWLEDGEMENTS

This work is supported by computational resources from the South African Astronomical Observatory, grant AST-1211384 from the National Science Foundation (USA), and workspace accommodation from the South African Astronomical Observatory and the University of Cape Town. We would like to thank John Bourke and Conrad Buck for their insights regarding mathematical technicalities and software development, respectively.
8. APPENDIX

The software authored herein was developed in Python 2.7. All numerical routines are constructed using the numpy package where applicable, and hence exhibit computational performance on par with C. The software suite is currently stored in a private GitHub repository. It will be made publicly available after a proprietary period in which the authors will maintain exclusive access to pursue their science objectives. Contributions are ongoing.

8.1. Performance Analysis

The primary sources of computational overhead are I/O operations and vector rotations. Once the initial MESA data are generated, it does not take longer than several minutes to generate the corresponding particle distribution, with reasonable step size and particle mass selections. While the losses imparted by inefficient coding within any component of MESA2GADGET are negligible on the timescale it takes either MESA or GADGET—or any other SPH code—to run, there is certainly room for improvement concerning specific subroutines. In no reasonable case is the computational overhead imposed by these convergence routines significant compared to the overhead of running either 1-D or 3-D simulation. Rigorous performance testing is ongoing and will be necessary following any adjustments to our numerical methods.

8.2. System Prerequisites

MESA2GADGET requires the following packages and their dependencies to run:

- MESA, preferably version 8118 or later
  - mesasdk: MESA software development kit
- GADGET 2.0.7
  - GSL: GNU Scientific Library
  - FFTW: fast Fourier transform library
- HEALPix
  - Fortran 90
  - IDL/GDL
  - C++
  - C

We suggest the following additional packages:

- SPH viewer, such as SPLASH or gadgetviewer
- GNUPLT

Components of MESA2GADGET can be run in parallel. MESA2GADGET has been verified in three Linux environments: the supercomputing cluster “mensa” at SAAO, the high performance computer LMC at Dartmouth, and the author’s PC. Transferability of the entire suite is currently limited by the difficulty of installing and linking MESA2GADGET’s required external packages. Documentation to assist with this process is under development.

8.3. Additional Python Packages

The following Python 2.7 packages are required by MESA2GADGET:

- argparse
- codecs
- datetime
- h5py
- hdf5lib
- healpy
- math
- matplotlib.pyplot
- numpy
- os
- pygadgetreader
- random
- re
- scipy.optimize
- subprocess
- sys
- tables
- time

REFERENCES

Price, D. J. 2011, SPLASH: An Interactive Visualization Tool for Smoothed Particle Hydrodynamics Simulations, Astrophysics Source Code Library, ascl:1103.004
Thompson, R. 2014, pyGadgetReader: GADGET snapshot reader for python, Astrophysics Source Code Library, ascl:1411.001