User's Guide for SLUG v. 2.0

Mark Krumholz

August 13, 2014

Contents

1 License and Citations

This is a guide for users of the SLUG software package. SLUG is distributed under the terms of the GNU General Public License v. 3. A copy of the license notification is included in the main SLUG directory. If you use SLUG in any published work, please cite the SLUG method paper, da Silva, R. L., Fumagalli, M., & Krumholz, M. R., 2012, *The Astrophysical Journal*, 745, 145. A second method paper, describing the upgraded version 2 code and a set of ancillary tools, is in preparation at this time.

2 What Does SLUG Do?

SLUG is a stellar population synthesis (SPS) code, meaning that, for a specified stellar initial mass function (IMF), star formation history (SFH), cluster mass function (CMF), and cluster lifetime function (CLF), it predicts the spectra and photometry of both individual star clusters and the galaxies (or sub-regions of galaxies) that contain them. In this regard, SLUG operates much like any other SPS code. The main difference is that SLUG regards the IMF, SFH, CMF, and CLF as probability distributions, and the resulting stellar population as being the result of a draw from them. SLUG performs a Monte Carlo simulation to determine the PDF of the light produced by the stellar populations that are drawn from these distributions. The remainder of this section briefly describes the major conceptual pieces of a SLUG simulation. For a more detailed description, readers are referred to da Silva, Fumagalli, & Krumholz (2012).

2.1 Cluster Simulations and Galaxy Simulations

SLUG can simulate either a simple stellar population (i.e., a group of stars all born at one time) or a composite stellar population, consisting of stars born at a distribution of times. We refer to the former case as a "cluster" simulation, and the latter as a "galaxy" simulation, since one can be thought of as approximating the behavior of a single star cluster, and the other as approximating a whole galaxy.

2.2 Probability Distribution Functions: the IMF, SFH, CMF, and CLF

As mentioned above, SLUG regards the IMF, SFH, CMF, and CLF as probability distribution functions. These PDFs can be described by a very wide range of possible functional forms; see Section 6 for details on the exact functional forms allowed, and on how they can be specified in the code. When SLUG runs a cluster simulation, it draws stars from the specified IMF in an attempt to produce a cluster of a user-specified total mass. There are a number of possible methods for performing such mass-limited sampling, and SLUG gives the user a wide menu of options; see Section 6.

For a galaxy simulation, the procedure involves one extra step. In this case, SLUG assumes that some fraction f_c of the stars in the galaxy are born in star clusters, which, for the purposes of SLUG means that they all share the same birth time. The remaining fraction $1 - f_c$ of stars are field stars. When a galaxy simulation is run, SLUG determines the total mass of stars M_* that should have formed since the start of the simulation (or since the last output, if more than one output is requested) from the star formation history, and then draws field stars and star clusters in an attempt to produce masses $(1 - f_c)M_*$ and $f_c M_*$. For the field stars, the stellar masses are drawn from the IMF, in a process completely analogous to the cluster case. For star clusters, the masses of the clusters are drawn from the CMF, and each cluster is then populated from the IMF as in the cluster case. For both the field stars and the star clusters, the time of their birth is drawn from the PDF describing the SFH.

Finally, star clusters can be disrupted independent of the fate of their parent stars. When each cluster is formed, it is assigned a lifetime drawn from the CLF. Once that time has passed, the cluster ceases to be entered in the lists of individual cluster spectra and photometry (see next section), although the individual stars continue to contribute to the integrated light of the galaxy.

2.3 From Stars to Light

Once SLUG has drawn a population of stars, its final step is to compute the light they produce. SLUG does this in several steps. First, it computes the physical properties of all the stars present user-specified times using a set of stellar evolutionary tracks. Second, it uses these physical properties to compute the composite spectra produced by the stars, using a user-specified set of stellar atmosphere models. Third and finally, it computes photometry for the stellar population by integrating the computed spectra over a set of specified photometric filters. The results are then written to file.

For a cluster simulation, this procedure is applied to the star cluster being simulated at a user-specified set of output times. For a galaxy simulation, the procedure is much the same, but it can be done both for all the stars in the galaxy taken as a whole, and individually for each star cluster that is still present (i.e., that has not been disrupted). Again, the results are written to file.

2.4 Monte Carlo Simulation

The steps described in the previous two simulations are those required for a single realization of the stellar population. However, the entire point of SLUG is to repeat this procedure many times in order to build up the statistics of the population light output. Thus the entire procedure can be repeated as many times as the user desires, in order to build up statistics.

3 Compiling and Installing SLUG

3.1 Dependencies

The core SLUG program requires

- The boost C++ libraries
- The GNU scientific library

Compilation will be easiest if you install these libraries such that the header files are included in your CXX_INCLUDE_PATH and the compiled object files are in your LD_LIBRARY_PATH. Alternately, you man manually specify the locations of these files by editing the Makefiles – see below.

In addition to the core dependencies, the python parsing routines require numpy and scipy.

3.2 Compiling

If you have boost and GSL included in your CXX_INCLUDE_PATH and LD_LIBRARY_PATH environment variables, and your system is running either MacOSX or Linux, you should be able to compile simply by doing

```
make
```

from the main slug directory. To compile in debug mode, do make debug instead.

Alternately, you can manually specify the compiler flags to be used by creating a file named Make.mach.MACHINE_NAME in the src directory, and then doing

```
make MACHINE=MACHINE_NAME.
```

An example machine-specific file, src/Make.mach.ucsc-hyades is included in the repository. You can also override or reset any compilation flag you want by editing the file src/Make.config.override

Finally, note that SLUG is written in C++11, and requires some C++11 features, so it may not work with older C++ compilers. The following compiler versions are known to work: gcc $\geq$ 4.7, clang/llvm $\geq$ 3.3, icc $\geq$ 14.0. Earlier versions may work as well, but no guarantees.

4 Running a SLUG Simulation

Once SLUG is compiled, running a simulation is extremely simple. Just do

```
./bin/slug param/filename.param
```

where filename.param is the name of a parameter file, formatted as specified in Section 5. The code will write a series of output files as described in Section 7.

5 Parameter Specification

An example parameter file is included as `param/example.param` in the source tree. Parameter files for `SLUG` are generically formatted as a series of entries of the form

```
keyword    value
```

Any line starting with `#` is considered to be a comment and is ignored. Some general rules on keywords are:

- Keywords may appear in any order.
- Some keywords have default values, indicated in parenthesis in the list below. These keywords are optional and need not appear in the parameter file. All others are required.
- Keywords and values are case-insensitive.
- Unless explicitly stated otherwise, units for mass are always $M_\odot$, units for time are always yr.
- Any time a file or directory is specified, if it is given as a relative rather than absolute path, it is assumed to be relative to the environment variable `$SLUG_DIR`. If this environment variable is not set, it is assumed to be relative to the current working directory.

The keywords recognized by `SLUG` can be categorized as follows:

Basic keywords. These specify basic data for the run.

- `model_name` (`SLUG_DEF`): name of the model. This will become the base filename for the output files.
- `out_dir` (`$SLUG_DIR/output`): name of the directory into which output should be written.
- `verbosity` (`1`): level of verbosity when running, with 0 indicating no output, 1 indicating some output, and 2 indicating a great deal of output.

Simulation control keywords. These control the operation of the simulation.

- `sim_type` (`galaxy`): set to `galaxy` to run a "galaxy" simulation (a composite stellar population), or to `cluster` to run a "cluster" simulation (a simple stellar population)
- `n_trials` (`1`): number of trials to run
- `log_time` (`0`): set to 0 for logarithmic time step, 1 for linear time steps

- `time_step`: size of the time step. If `log_time` is set to 0, this is in yr. If `log_time` is set to 1, this is in dex (i.e., a value of 0.2 indicates that every 5 time steps correspond to a factor of 10 increase in time).
- `start_time`: first output time. This may be omitted if `log_time` is set to 0, in which case it defaults to a value equal to `time_step`.
- `end_time`: last output time, in yr. Note that not all the tracks include entries going out to times > 1 Gyr, and the results will become inaccurate if the final time is larger than the tracks allow.
- `sfr`: star formation rate. Only used if `sim_type` is `galaxy`; for `cluster`, it will be ignored, and can be omitted. If this is set to a positive value, it is taken to indicate a constant star formation rate. If it is set to a negative value, it is taken as a flag indicating that a star formation history file should be read.
- `sfh`: name of star formation history file. This file is a PDF file, formatted as described in Section 6. This is ignored, and can be omitted, if `sim_type` is `cluster`, or if `sfr` is set to a positive value.
- `cluster_mass`: mass of the star cluster for simulations with `sim_type` set to `cluster`. This can be omitted, and will be ignored, if `sim_type` is `galaxy`.
- `redshift` (`0`): place the system at the specified redshift. The computed spectra and photometry will then be computed in the observed rather than the rest frame of the system.

Output control keyword. These control what quantities are computed and written to disk.

- `out_cluster` (`1`): write out the physical properties of star clusters? Set to 1 for yes, 0 for no.
- `out_cluster_phot` (`1`): write out the photometry of star clusters? Set to 1 for yes, 0 for no.
- `out_cluster_spec` (`1`): write out the spectra of star clusters? Set to 1 for yes, 0 for no.
- `out_integrated` (`1`): write out the integrated physical properties of the whole galaxy? Set to 1 for yes, 0 for no. This keyword is ignored if `sim_type` is `cluster`.
- `out_integrated_phot` (`1`): write out the integrated photometry of the entire galaxy? Set to 1 for yes, 0 for no. This keyword is ignored if `sim_type` is `cluster`.
- `out_integrated_spec` (`1`): write out the integrated spectra of the entire galaxy? Set to 1 for yes, 0 for no. This keyword is ignored if `sim_type` is `cluster`.

- `output_mode` (`ascii`): set to `ascii` or `binary`. Selecting `ascii` causes the output to be written in ASCII text, which is human-readable, but produces much larger files. Selecting `binary` causes the output to be written in raw binary; see Section 7 for a description of the output format. Both modes of output can be read by the python parsing tools, though `binary` files are of course much faster to read.

Physical model keywords. These specify the physical models to be used for stellar evolution, atmospheres, the IMF, etc.

- `imf` (`$SLUG_DIR/lib/imf/chabrier.imf`): name of the IMF descriptor file; this is a PDF file, formatted as described in Section 6. Note that `SLUG` ships with the following IMF files pre-defined (in the directory `lib/imf`)
  - `chabrier.imf` (single-star IMF from Chabrier, 2005, in "The Initial Mass Function 50 Years Later", eds. E. Corbelli, F. Palla, & H. Zinnecker, Springer: Dordrecht, p. 41)
  - `chabrier03.imf` (single-star IMF from Chabrier, 2003, PASP, 115, 763-795)
  - `kroupa.imf` (IMF from Kroupa, 2002, Science, 295, 82-91)
  - `kroupa_sb99.imf` (simplified version of the Kroupa, 2002 IMF used by default by `starburst99`, `http://www.stsci.edu/science/starburst99/docs/default.htm`)
  - `salpeter.imf` (single-component power law IMF from Salpeter, 1955, ApJ, 121, 161)
- `cmf` (`$SLUG_DIR/lib/cmf/slug_default.cmf`): name of the CMF descriptor file; this is a PDF file, formatted as described in Section 6. The default selection is a power law $dN/dM \propto M^{-2}$ from $M = 10^2 - 10^7\ M_\odot$. This is ignored, and may be omitted, if `sim_type` is set to `cluster`.
- `clf` (`$SLUG_DIR/lib/clf/slug_default.clf`): name of the CLF descriptor file; this is a PDF file, formatted as described in Section 6. The default gives a power law distribution of lifetimes t with $dN/dt \propto t^{-1.9}$ from 1 Myr to 1 Gyr. Note that this corresponds to a cluster age distribution of slope -0.9. The `SLUG` source also ships with an alternative CLF file, `lib/clf/nodisrupt.clf`, which disables cluster disruption entirely (by setting the lifetime distribution to a δ function at 10^{300} yr).
- `tracks` (`$SLUG_DIR/lib/tracks/Z0140v00.txt`): stellar evolution tracks to use. The following tracks ship with `SLUG` (all in the directory `lib/tracks`):
  - `ZXXXXvYY.txt`: Geneva (2013) tracks; metallicities are Solar (`XXXX` = `0140`) and 1/7 Solar (`XXXX` = `0020`), and rotation rates are 0 (`YY` = `00`) and 40% of breakup (`YY` = `40`).
  - `modcXXX.dat`: Geneva tracks with standard mass loss, for metallicities of 2× Solar (`040`), Solar (`020`), 0.4× Solar (`008`), 0.2× Solar (`004`), and 0.05× Solar (`001`).

  - `modeXXX.dat`: same as `modcXXX.dat`, but with higher mass loss rates.
  - `modpXXX.dat`: Padova tracks with thermally pulsing AGB stars; metallicities use the same scale as `modcXXX.dat` files (i.e., `020` is Solar).
  - `modsXXX.dat`: same as `modpXXX.dat`, but without thermally pulsing AGB stars

- `atmospheres` (`$SLUG_DIR/lib/atmospheres`): directory where the stellar atmosphere library is located. Note that file names are hard-coded, so if you want to use different atmosphere models with a different format, you will have to write new source code to do so.

- `specsyn_mode` (`sb99`): spectral synthesis mode. Allowed values are:
  - `planck`: treat all stars as black bodies
  - `Kurucz`: use Kurucz atmospheres, as compiled by Lejeune et al. (1997, A&AS, 125, 229), for all stars
  - `Kurucz+Hillier`: use Kurucz atmospheres for all stars except Wolf-Rayet stars; WR stars use Hillier model atmospheres (Hiller & Miller, 1998, ApJ, 496, 407)
  - `Kurucz+Pauldrach`: use Kurucz atmospheres for all stars except OB stars; OB stars use Pauldrach model atmospheres (Pauldrach et al., 2001, A&A, 375, 161)
  - `SB99`: emulate the behavior of `starburst99`: use Pauldrach for OB stars, Hillier for WR stars, and Kurucz for all other stars

- `clust_frac` (`1.0`): fraction of stars formed in clusters

- `min_stoch_mass` (`0.0`): minimum stellar mass to be treated stochastically. All stars with masses below this value are assumed to be sampled continuously from the IMF.

- `metallicity`: metallicity of the stellar population, relative to solar. This may be omitted if `tracks` is set to one of the default sets of tracks that ships with `SLUG`, as the metallicities for these tracks are hardwired in. This keyword is provided to allow users to supply their own tracks.

- `WR_mass`: minimum starting mass that stars must have in order to pass through a Wolf-Rayet phase. This can be omitted if `tracks` is set to one of the default sets of tracks that ships with `SLUG`, as the WR cutoff masses for these tracks are hardwired in. This keyword is provided to allow users to supply their own tracks.

Photometric filter keywords. These describe the photometry to be computed. Note that none of these keywords have any effect unless `out_integrated_phot` or `out_cluster_phot` is set to 1.

- `phot_bands`: photometric bands for which photometry is to be computed. The values listed here can be comma- or whitespace-separated. For a list of available photometric filters, see the file `lib/filters/FILTER_LIST`.

- `filters` (`$SLUG_DIR/lib/filters`): directory containing photometric filter data
- `phot_mode` (`L_nu`): photometric system to be used when writing photometric outputs. Allowed values are:
  - `L_nu`: report frequency-averaged luminosity in the band, in units of erg/s/Hz
  - `L_lambda`: report wavelength-averaged luminosity in the band, in units of erg/s/Å
  - `AB`: report AB magnitude
  - `STMAG`: report ST magnitude
  - `VEGA`: report Vega magnitude

6 Probability Distribution Functions

7 Output Files and Format

8 The Python Output Parser