Manual for FSPS v2.0

1. Overview

The collection of fortran routines contained in this package allows the user to compute simple stellar populations (SSPs) for a variety of IMFs and metallicities, and for a variety of assumptions regarding the morphology of the horizontal branch, the blue straggler population, the post–AGB phase, and the location in the HR diagram of the TP-AGB phase. From these SSPs the user may then generate composite stellar populations (CSPs) for a variety of star formation histories (SFHs) and dust attenuation prescriptions. Outputs include the 'observed' spectra and magnitudes of the SSPs and CSPs at arbitrary redshift. In addition to these fortran routines, we provide a collection of IDL routines that allow easy manipulation of the output.

The user is strongly encouraged to read Conroy et al. 2009a for an overview of stellar population synthesis (SPS) and for details regarding this collection of routines. As of v2.0, this code has been extensively calibrated against a suite of observational data (for details see Conroy & Gunn 2010). The code package is a in essence a highly flexible SPS code and has therefore come to be called FSPS.

The rest of this manual is organized as follows. In §2 we briefly discuss the overall philosophy of the code structure. In §3 we present the routines, how they are used, and provide the user with a basic program that demonstrates the use of the routines. §4 discusses the IDL routines provided to read and manipulate the outputs from the fortran package. In §5 we discuss a variety of questions the user may have, and §6 highlights improvements that will be made in the near-term.

1.0 Environment variables

As mentioned on the webpage, you will need to set an environment variable called SPS_HOME that points to the root SPS directory (i.e. the directory that contains the src, OUTPUTS, etc. directories)

1.1 A note on previous versions

The TP-AGB empirical spectral library does not extend past the rest-frame K-band and so in previous versions the integrated spectra were not reliable beyond $\lambda \approx 2.4 \mu m$. In the current version we have extrapolated the empirical TP-AGB spectra with the BaSeL library for models at the same $T_{\rm eff}$ and the lowest surface gravities available in the BaSeL library. Integrating over the wavelength now produces a bolometric luminosity in good agreement with the $L_{\rm Lbol}$ available in the outputs.

1.2 Units

This code produces two types of files. The first are spectral files (*.spec). The spectra are in units of $L_\odot$/Hz, i.e. they are f_ν. Integrating over frequency generates the bolometric luminosity of the spectrum. Wavelengths are in angstroms, and the wavelength array can be found in the file `BaSeL3.1/basel.lambda`. The second type of file contains magnitudes in a variety of filters. The magnitude zero point (i.e., AB or Vega) is set by the variable `compute_vega_mags` described below. Both of these output files contain information on the age, mass, bolometric luminosity, and star formation rate. All of these quantities are in the log (base-10), with units of years, $M_\odot$, $L_\odot$, and $M_\odot$ yr^{-1}, respectively.

1.3 Redshifting

The code allows the user to specify a redshift (see Section 3.1.2 below). Specifying the redshift only impacts the computed magnitudes. In other words, the magnitudes are computed for a spectrum that has been redshifted (not taking into account dimming due to the luminosity distance). The output *spectrum* is, however, not redshifted. We assume that the user could translate this spectrum to the redshift of his or her choice with relative ease. We emphasize again that the distance modulus is *not* included. See the parameter `redshift_colors` for more details.

1.4 Filters

The current release contains 47 filters, which are listed in the file `FILTER_LIST`, which can be found in

the source directory. The output magnitudes are listed in this order. The user is cautioned to use whenever possible the exact filter transmission curve appropriate for the data being considered. There is, for example, no such thing as THE $B-$band filter. Differences of a few hundredths of a magnitude are common between different definitions of a given filter such as B.

1.4 Interpretation of the Δ_L and Δ_T parameters

As described below, the user may modify the bolometric luminosity and effective temperature of the TP-AGB phase by applying overall shifts in log(L) and log(T) via the parameters Δ_L and Δ_T. In previous versions, these parameters were with respect to the default Padova model calculations circa 2008. As of v2.0, these parameters represent shifts with respect to the best-fit values found in Conroy & Gunn 2010. In other words, leaving these values set to 0.0 means that the user adopts the calibrations described in Section 3.1.3 in Conroy & Gunn 2010.

2. How to use this code, generally speaking

The code provided in this package is optimally designed to be integrated into a larger fortran program. The code reads all of the libraries into memory and then utilizes various routines to compute SSPs and CSPs. The benefit of the structure of this package is that it allows the user to produce very large numbers of models relatively quickly. It also allows one to easily integrate SPS into particular science tasks.

For those more interested in getting quick results, rather than using the more flexible aspects of the code, we also provide a routine that generates results after prompting the user for input. This program is called `autosps.exe`.

Note that all outputs are placed in the `OUTPUTS` directory, included in the tarball, by default.

3. Details of Fortran Routines

3.1 The `sps_vars.f90` Module: Common Variables, Parameters, and Structure Set-up

It is recommended that the user consider carefully the `sps_vars.f90` module. This module must be called at the beginning of every program that uses the routines described below. This module sets up two types of variables: common and parameters. Both parameters and common variables can be seen by all routines and thus do not have to be explicitly passed during a call to a routine. The difference between the two is that the parameters are defined once in `sps_vars.f90` and thereafter cannot be changed. The common variables, by contrast, can be changed by various routines (see § 3.1.1). Be careful when changing common variables! The variables in this module are well-documented. Most parameters should not be changed.

The `sps_vars.f90` module also sets up structures (for those who are not familiar with fortran structures, they are similar in many ways to structures in IDL). There are two main structures that are used extensively in many routines. One of these is defined to conveniently handle the output of the `compsp.f90` routine; it should not be of much concern for most users. The second structure defines the parameter set. It is sufficiently important to warrant a more detailed discussion (see § 3.1.2).

3.1.1 Common Variables & Parameters

We briefly describe several of the most important and most used common variables and parameters. These parameters are defined in `sps_vars.f90`.

- `verbose`

Parameter that controls output to screen. If set to 1 a lot of output will be printed to screen, if set to 0 the programs will be silent.

- `dust_type`

Common variable defining the extinction curve for dust around old stars:

  - **0:** power–law with index dust_index set in the parameter structure
  - **1:** Milky Way extinction law (with $R \equiv A_V/E(B-V)$ value defined in parameter set; see below) parameterized by Cardelli et al. 1989.
  - **2:** Calzetti et al. 2000 attenuation curve. Note that if this value is set then the dust attenuation is applied to all starlight equally (not split by age), and therefore the only relevant parameter is `dust2` (defined below), which sets the overall normalization.
  - **3:** allows the user to access a variety of attenuation curve models from Witt & Gordon 2000. See the parameters `wgp1` and `wgp2` in §3.1.2.

- `tuniv`

Common variable setting the age of the Universe. Used when computing SFH (see below).

- `compute_vega_mags`

A switch that sets the zero points of the magnitude system:

  - **0:** AB system
  - **1:** Vega system

- `imf_type`

Common variable defining the IMF type:

  - **0:** Salpeter 1955
  - **1:** Chabrier 2003
  - **2:** Kroupa 2001
  - **3:** van Dokkum 2008

- `isoc_type`

Switch to use either the Padova isochrones (`isoc_type='pdva'`) or BaSTI isochrones (`isoc_type='bsti'`). Be aware that the parameter `pz` needs to reflect the user's isochrone choice.

- `redshift_colors`

  - **0:** Magnitudes are computed at a fixed redshift specified in the parameter set (see below)
  - **1:** Magnitudes are computed at a redshift that corresponds to the age of the output SSP/CSP (assuming a redshift–age relation appropriate for a WMAP5 cosmology). This switch is useful if the user wants to compute the evolution in *observed* colors of a SSP/CSP.

3.1.2 The Parameter Set

The parameter set is a structure defined in the `sps_vars.f90` module. It must be defined at the beginning of every program for the various routines to properly work. This structure acts as the primary interface between what the user would like to compute and the various subroutines that actually do the work. Simply defining the structure at the beginning of the program will set all structure elements to their default values (see the example program `simple.f90` below). The elements of the structure, and the default values, are described below. The parameters are organized by topic, e.g., star formation history, IMF, dust, and stellar evolution parameters.

- `zred` Redshift. If this value is non-zero the magnitudes will be computed for the spectrum placed at redshift `zred`. *Default value is 0.0.*
- `zmet` Metallicity. The metallicity is specified as an integer ranging between 1 and 22 for the Padova isochrones and between 1 and 10 for the BaSTI isochrones. A lookup table for the actual metallicities corresponding to the integer values is provided at the end of this manual. Note that $Z_\odot = 0.0190$. *Default value is 1.*
- `sfh` defines the type of star formation history, normalized such that one solar mass of stars is formed over the age of the Universe. *Default value is 0.*
  - **0:** Compute an SSP
  - **1:** Compute a five parameter SFH (see below).
  - **2:** Compute a tabulated SFH defined in a file called sfh.dat that must reside in the same directory as the executable. The file must contain three rows. The first column is time since the Big Bang in gigayears, the second is the SFR in units of solar masses per year, the third is the absolute metallicity. An example is provided in the src directory.
- `tau` Defines e-folding time for the SFH, in Gyr. Only used if `sfh`= 1. *Default value is 1.0*, range is $0.01 < \tau < 10^2$.
- `const` Defines the constant component of the SFH. This quantity is defined as the fraction of mass formed in a constant mode of SF; the range is therefore $0 \leq C \leq 1$. Only used if `sfh`= 1. *Default value is 0.0.*
- `tage` Defines the age of the system, not weighted by the SFH. In other words, the SF starts at $t = 0$ and ends at $t =$`tage`. Only used if `sfh`= 1. *Default value is 0.0.* If the default value is set, then the output will sample time logarithmically from 0.0 to $T_{\rm univ}$ (this value is set in `sps_vars.f90`). If `tage`> 0 then only one time will be output at the time `tage`. This may seem like a rather awkward way to handle `tage`; it is chosen for ease when coupling this code to an MCMC. Basically, if you want to run a model with time sampled from 0.0 to a value T, then set $T_{\rm univ} = T$ and `tage`= 0.0. if you want to run a model with a single output at time T, then set `tage`$= T$.
- `fburst` Defines the fraction of mass formed in an instantaneous burst of star formation. Only used if `sfh`= 1. *Default value is 0.0.*
- `tburst` Defines the age of the Universe when the burst occurs. If `tburst`>`tage` then there is no burst. Only used if `sfh`= 1. *Default value is 0.0.*
- `imf1` Logarithmic slope of the IMF over the range $0.08 < M < 0.5\,M_\odot$. Only used if `imf_type`= 2. *Default value is 1.3.*
- `imf2` Logarithmic slope of the IMF over the range $0.5 < M < 1.0\,M_\odot$. Only used if `imf_type`= 2. *Default value is 2.3.*
- `imf3` Logarithmic slope of the IMF over the range $1.0 < M < 100\,M_\odot$. Only used if `imf_type`= 2. *Default value is 2.3.*

- `vdmc` IMF parameter defined in van Dokkum 2008. Only used if `imf_type`= 3. *Default value is 0.08.*
- `dust_tesc` Stars younger than `dust_tesc` are attenuated by both `dust1` and `dust2`, while stars older are attenuated by `dust2` only. Units are log(yrs). *Default value is 7.0.*
- `dust1` Dust parameter describing the attenuation of young stellar light, i.e. where $t \leq$`dust_tesc` (for details, see Conroy et al. 2009a). *Default value is 0.0.*
- `dust2` Dust parameter describing the attenuation of old stellar light, i.e. where $t >$`dust_tesc` (for details, see Conroy et al. 2009a). *Default value is 0.0.*
- `dust_clumps` Dust parameter describing the dispersion of a Gaussian PDF density distribution for the old dust. Setting this value to -99.0 sets the distribution to a uniform screen. *Default value is -99.0.* See Conroy et al. 2009b for details.
- `frac_nodust` Fraction of starlight that is not attenuated by the diffuse dust component (i.e. that is not affected by `dust2`). *Default value is 0.0.*
- `dust_index` Power–law index of the attenuation curve. Only used when `dust_type`=0. *Default value is -0.7.*
- `mwr` The ratio of total to selective absorption which characterizes the MW extinction curve: $R \equiv A_V/E(B-V)$. Only used when `dust_type`=1. *Default value is 3.1.*
- `uvb` Parameter characterizing the strength of the 2175Åextinction feature with respect to the standard Cardelli et al. determination for the MW. Only used when `dust_type`=1. *Default value is 1.0.*
- `wgp1` Integer specifying the optical depth in the Witt & Gordon 2000 (WG00) models. Values range from $1-18$, corresponding to optical depths of $0.25, 0.50, 0.75, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00, 5.50, 6.00, 7.00, 8.00, 9.00, 10.0$. Note that these optical depths are defined differently from the optical depths defined by the parameters `dust1` and `dust2`. See WG00 for details.
- `wgp2` Integer specifying the type of large–scale geometry and extinction curve. Values range from $1-6$, corresponding to MW+dusty, MW+shell, MW+cloudy, SMC+dusty, SMC+shell, SMC+cloudy. MW= Milky Way extinction; SMC= Small Magellanic Cloud extinction. Dusty, shell, and cloudy specify the geometry and are described in WG00.
- `wgp3` Integer specifying the local geometry for the WG00 dust models. A value of 0 corresponds to a homogeneous distribution, and a value of 1 corresponds to a clumpy distribution. See WG00 for details.
- `dell` Shift in $\log(L_{\rm bol})$ of the TP-AGB isochrones. Note that the meaning of this parameter and the one below has changed to reflect the updated calibrations presented in Conroy & Gunn 2009. That is, these parameters now refer to a modification about the calibrations presented in that paper. *Default value is 0.0.*
- `delt` Shift in $\log(T_{\rm eff})$ of the TP-AGB isochrones. *Default value is 0.0.*
- `sbss` Specific frequency of blue straggler stars. See Conroy et al. 2009a for details and a plausible range. *Default value is 0.0.*
- `fbhb` Fraction of horizontal branch stars that are blue. The blue HB stars are uniformly spread in $\log(T_{\rm eff})$ to 10^4 K. See Conroy et al. 2009a for details and a plausible range. *Default value is 0.0.*
- `pagb` Weight given to the post–AGB phase. A value of 0.0 turns off post–AGB stars; a value of 1.0 implies that the Vassiliadis & Wood 1994 tracks are implemented as–is. *Default value is 1.0.*

3.2 Description of Fortran Routines

We now discuss the purpose and syntax of the routines in this package. The first two, `ssp_gen.f90` and `compsp.90`, are the only two routines that most users will need to call. The other routines are primarily used internally, although they can integrated into other routines at the users discretion. At the end of this section we provide the user with a simple routine that demonstrates the syntax and calling sequence for these routines. The user should be able to modify this routine for most straightforward purposes.

- `SSP_GEN(pset,mass_ssp,lbol_ssp,spec_ssp)`

This routine takes as input the parameter set, `pset`, and outputs the time-dependent mass, `mass_ssp`, bolometric luminosity, `lbol_ssp`, and spectrum, `spec_ssp`, of an SSP defined by the parameter set. Each of these input variables must be properly defined at the beginning of the main routine. See the example below.

- `COMPSP(write_compsp,nzin,outfile,mass_ssp,lbol_ssp,spec_ssp,pset,ocompsp)`

This routine takes as input `mass_ssp`, `lbol_ssp`, and `spec_ssp`, which are the outputs of the routine `ssp_gen.f90`. The user must also provide the parameter set `pset` and filename for output in `outfile`, if output is desired (a blank string may be specified if no output is desired). There are four possible outputs, specified by `write_compsp`: No output (0), output magnitudes (1), output spectra (2), output magnitudes and spectra (3). The magnitude and spectra outputs are written to files with the output filename with ".mags" and ".spec" appended.

A variety of output from this routine is saved in the `ocompsp` variable, which must be a structure as defined in `sps_vars.f90`. Again, see the example routine below.

The `nzin` parameter sets the number of metallicity points passed. For standard, single metallicity calculations, set this value to one and simply pass the outputs from `ssp_gen.f90`. However, if one wants to compute spectra for an evolving metallicity (i.e. when computing a tabulated SFH, the metallicity history may be specified), one needs to set this parameter to the number of metallicity elements available (in the default release this is 22). One then must take care to pass the metallicity–dependent `ssp_gen.f90` outputs. Currently the code only allows the specification of a metallicity history when passing tabulated SFH.

- `GETMAGS(zred,spec,mags)`

The input is the redshift `zred` and spectrum `spec`. The output is an array of magnitudes `mags` for the redshifted spectrum.

- `SPS_SETUP(zin)`

This routine must be called at least once before running any routines. It reads in all of the isochrones and spectral libraries and stores them in a common block. If the user requires only one metallicity, then that metallicity can be specified as `zin`. The metallicity must be specified as an integer corresponding to the look-up table at the end of this manual. If the user wishes to read in all metallicities, then `zin` should be set to -1.

- `VELBROAD(lambda,spec,sigma)`

This routine broadens the input spectrum `spec` with corresponding wavelength array `lambda` by a velocity dispersion `sigma` measured in km/s. Note that the velocity broadening is only approximate in that we are ignoring the variation in $d\lambda$ with λ within each integration step.

- `GETINDX(lambda,spec,indices)`

This routine computes the spectral indices for the input spectrum `spec` with corresponding wavelength array `lambda`, returning the indices in an array `indices`. The `indices` array must be defined with an array length specified by the variable `nindsps`, which specifies the number of indices. The indices are defined in the file allindices.dat in the src/ directory. These indices are adopted from the MPA/JHU database definitions.

3.2.1. Example Routines

The code package contains several routines that highlights many of the features of the model. The routine `simple.f90` demonstrates the basic syntax required to generate simple models. In addition, there is a less simple routine located in the src directory, called `lesssimple.f90`, that highlights some more advanced features of the code.

4. Details of IDL Routines

- `res = read_mags(file)`

This function takes as input the magnitude file (*.mags) produced by `compsp.f90`. The output is a structure with elements including the some of the magnitudes listed in `FILTER_LIST` and computed by `compsp.f90`. Note that not all magnitudes computed and listed in the *.mags file are contained in the resulting structure. This routine can be trivially modified to include other/all of the magnitudes computed.

- `res = read_spec(file)`

This function takes as input the spectra file (*.spec) produced by `compsp.f90`. The output is a structure with elements including the time-dependent spectrum computed by `compsp.f90`.

5. How do I....

5.1 add additional filters?

Adding additional filters is straightforward. There are three things the user must do: 1) modify the `nbands` parameter in the `sps_vars.f90` routine; 2) add the filter to the `allfilters.dat` file located in the src directory. Follow the format: there must be a line starting with a # sign, followed by two columns, the first being the wavelength in angstroms, the second being the total throughput. The filter can be of any resolution, and need not be properly normalized. Finally, the user would be wise to add details of the filter to the `FILTER_LIST` file located in the src directory, although this is not, strictly speaking, necessary for proper functioning of the code.

We would appreciate it if the user would email us if they add a filter so that we can include this filter in later releases (thereby saving others the trouble of adding filters).

6. The Miles empirical spectral library

The Miles empirical spectral library has been incorporated into FSPS, although this library is not included in the public release due to memory issues (that will hopefully be resolved soon). If the user is interested in computing models with the Miles library, please email `cconroy@astro.princeton.edu`. We would be happy to pass along SSPs constructed with the Miles library.

Table 1: Lookup table of metallicity values

zmet	Z [$\log(Z/Z_\odot)$] for Padova	Z [$\log(Z/Z_\odot)$] for BaSTI
1	0.0002 (-1.98)	0.0003 (-1.80)
2	0.0003 (-1.80)	0.0006 (-1.50)
3	0.0004 (-1.68)	0.0010 (-1.28)
4	0.0005 (-1.58)	0.0020 (-0.98)
5	0.0006 (-1.50)	0.0040 (-0.68)
6	0.0008 (-1.38)	0.0080 (-0.38)
7	0.0010 (-1.28)	0.0100 (-0.28)
8	0.0012 (-1.20)	0.0200 (+0.02)
9	0.0016 (-1.07)	0.0300 (+0.20)
10	0.0020 (-0.98)	0.0400 (+0.32)
11	0.0025 (-0.89)	
12	0.0031 (-0.79)	
13	0.0039 (-0.69)	
14	0.0049 (-0.59)	
15	0.0061 (-0.49)	
16	0.0077 (-0.39)	
17	0.0096 (-0.30)	
18	0.0120 (-0.20)	
19	0.0150 (-0.10)	
20	0.0190 (+0.00)	
21	0.0240 (+0.10)	
22	0.0300 (+0.20)	

We assume $Z_\odot = 0.0190$.