Manual for alf v2.0

1.1 Overview

alf is a program for fitting the absorption line optical-NIR spectrum of an old ($\gtrsim$ 1 Gyr) stellar system. The underlying models were first presented in Conroy & van Dokkum (2012a) and were specfically designed to constrain the stellar IMF in old massive galaxies. Over the years the models have evolved considerably (as described in Choi et al. 2014; Conroy et al. 2014; Conroy et al. in prep.) and now include many features, from theoretical age and metallicity-dependent response functions covering 19 elements, to several nuisance parameters meant to capture uncertainties in stellar evolution, to parameters capturing uncertainties in the data, including modeling telluric absorption and sky line residuals. The fitting is handled with the ensemble MCMC sampler emcee (Foreman-Mackey et al. 2013). alf is MPI parallelized and runs efficiently on many processors. Fitting data with alf is time intensive - models are often only fully converged after 4 hrs on 32 cores.

Initially alf was only capable of fitting populations with metallicities near solar (e.g., within a factor of $\sim$ 2). As of mid-2016, with the addition of new empirical SSPs computed by Alexa Villaume, and new theoretical response functions, alf is now capable of fitting stellar populations with metallicities from approximately -2.0 to +0.3 and performs well when fitting stellar populations ranging from metal-poor globular clusters to brightest cluster galaxies.

alf works in continuum-normalized space and so does not make any use of the shape of the continuum (nor of corresponding photometry). This is a place for possible future improvement, though marginalizing over issues such as flux calibration and dust attenuation are serious challenges at the high level of precision ($\lesssim$ 1%) that alf was designed to operate.

1.2 Setup, installation, and running the code

Download the code from git: github.com/cconroy20/alf. You will need to set an environment variable called ALF_HOME that points to the root alf directory (i.e. the directory that contains the src, bin, etc. directories). alf is written to run via MPI parallelization, and so requires the mpifort compiler.

The syntax for running the code is: mpirun -np N alf.exe infile.dat, where N is the number of processors and infile.dat is a file located in the indata directory that contains the input spectrum (see below).

1.3 The input file

All input spectra are kept in the indata directory. An example input file is included in the github distribution. The format is as follows: The first N lines contain header information for which wavelength ranges to include in the fit, e.g., "# 0.40 0.47" tells alf to fit a wavelength chunk from $0.40\mu m - 0.47\mu m$. Up to 10 wavelength chunks can be defined in this way. Within each wavelength chunk the input spectrum and model are continuum matched via a polynomial with one order per 100A (the above example wavelength range would therefore use a 7th order polynomial).

After the header, the columns are wavelength, flux, error, weight, instrumental resolution. Wavelengths are in vacuum and can either be in the observed or restframe. Flux can have arbitrary units and does not need to be flux calibrated. The weight column can be used to mask or otherwise down-weight regions of the spectrum for whatever reason. The instrumental resolution is in units of km/s (sigma of a Gaussian) and can be set to 0.0 if the instrumental resolution is not known. Currently the instrumental resolution is applied once to all of the models at setup in the model restframe, so if the data is at cosmological redshifts then this feature will not work properly.

1.3 The output files

alf produces up to three output files with extensions *.sum, *.mcmc, and *.bestspec. The first two files have the same format: the first column is the χ^2, the next N columns are the parameters, then three mass-to-light ratios in the r, I, and K bands, and then three mass-to-light ratios in the same bands but asssuming a fixed IMF (Kroupa in all cases except for imf_type=0, in which case Salpeter is the reference IMF). The order of the N parameters is given in the routine str2arr.f90. See also the IDL routine read_alf.pro in the

`scripts` directory.

The file *.sum contains summary information for each parameter. The rows are: mean of the posterior, parameters at $\chi^2_{\rm min}$, 1σ errors, 2.5%, 16%, 50%, 84%, 97.5% CLs, and lower and upper limits on the parameters (lower and upper priors). The *.mcmc contains the outputs of the MCMC chains (if `print_mcmc` is set, see below).

The file *.bestpsec contains the model and data spectra exactly as they were used to compute χ^2. The format is: wavelength, model spectrum, data spectrum, data S/N (including jitter and inflated errors around sky lines if applicable), and the polynomial that was multiplied by the original model to produce the model column in the file.

1.4 Common variables

These are global variables that the user can set at the beginning of `alf.f90` that will effect various aspects of the code:

- `nmcmc` Number of steps taken after the burn-in phase. This value does not need to be large (e.g., 100) presuming that the chains are in fact burned in.
- `nsample` Inverse of the sampling rate used to print walkers to the *mcmc file.
- `nburn` Number of steps taken during the burn-in phase. Suggested value is $> 10^4$.
- `nwalkers` Number of walkers used by `emcee`. Suggested value is 1024.
- `dopowell` Flag indicating whether Powell minimization is run at the beginning of `alf` in order to determine logage, velocity, sigma, and [Z/H]. Turned off by default. This step does not seem to be necessary to speed up the convergence and can cause the code to get stuck in undesirable parts of parameter space.
- `print_mcmc` Flag indicating whether the *.mcmc output file will be created (1=yes, 0=no).
- `fit_type` Flag determining the type of fitting to be performed:
  - **2:** super-simple mode. Only a single age, [Z/H], velocity, and velocity dispersion are included in the fit.
  - **1:** simple mode. In addition to the variables in super-simple mode, the abundances of the elements C,N,O,Mg,Si,Ca,Ti, and Na are fit.
  - **0:** All 42 parameters are included in the fit, including various nuisance parameters, emission lines, a two-component SFH, IMF, and various data-related issues including a jitter term, fitting for atmospheric transmission, and inflating the errors around sky lines.
- `mwimf` Flag indicating of the IMF is to be fixed to the MW value (Salpeter for `imf_type`=2, Kroupa otherwise). This flag is automatically set to 1 if fit_type=2 or 1.
- `observed_frame` Flag indicating of the input data are in the original observed frame or not. If yes, then atmospheric transmission and error inflation around sky lines are turned on.
- `smooth_trans` Variable specifying the degree of additional smoothing (in km/s) to be applied to the transmission function template. Set this to a non-zero value only if the input spectrum has been smoothed by an amount over and above the instrumental resolution.
- `imf_type` Flag indicating the type of IMF to be fit. Note that in all cases the IMF slope above 1 Msun is fixed to the Salpeter value (2.3). This parameter only applies if `fit_type`=0):
  - **0:** single power-law with cutoffs of 0.08 and 100 Msun (free parameter is `imf1`).
  - **1:** double power-law with cutoffs of 0.08 and 100 Msun (free parameters are `imf1` and `imf2`).
  - **2:** single power-law with variable lower cutoff (free parameters are `imf1` and `imf3`).

  - **3:** double power-law with variable lower cutoff (free parameters are `imf1`, `imf2`, and `imf3`).
  - **4:** non-parametric IMF with four free variables: `imf1`, `imf2`, `imf3`, and `imf4` specifying the log weight in each IMF mass bin. There is a regularization constraint requiring that the IMF be globally smooth. The intra-bin weighting can be chosen to be either flat or Salpeter and is set by the variable `nonpimf_alpha`.

- `atlas_imf` Choice of IMF used to construct the theoretical response functions. Options are 'krpa' and 'salp'.

- `nonpimf_alpha` Intrabin weighting for the non-parametric IMF. Choices include 0.0 and 2.3. Default is 2.3 (Salpeter).

- `fit_two_ages` Option to turn on/off the second age component in the full model. Default is on (1).

- `nonpimf_regularize` Option to force regularization of the non-parametric IMF. Default is on (1).

- `use_age_dep_resp_fcns` Flag indicating whether age-dependent response functions are used. Default is yes (1). If turned off, then the user must specify which age will be the reference through the variable `fix_age_dep_resp_fcns`, which is by default set to 10 Gyr.

- `use_z_dep_resp_fcns` Flag indicating whether metallicity-dependent response functions are used. Default is yes (1). If turned off, then the user must specify which metallicity will be the reference through the variable `fix_z_dep_resp_fcns`, which is by default set to [Z/H]=0.0.

1.5 The parameter set

The parameter set is a structure defined in the `alf_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. The elements of the structure, and the default values, are described below.

- `velz` Recession velocity in km/s.

- `sigma` Velocity dispersion in km/s.

- `logage` log(age) in Gyr of the dominant stellar population (the only age fit in simple and super-simple modes)

- `zh` Overall metallicity, [Z/H]. Determines which SSPs and which response functions to use.

- elements (`feh,ah,ch,nh,nah,mgh,sih,kh,cah,tih,vh,crh,mnh,coh,nih,cuh,srh,bah,euh`) Abundances of individual elements, determined by the theoretical response functions.

- `teff` Shift in $T_{\rm eff}$ relative to the fiducial isochrones.

- `imf1` Low-mass IMF slope. For `imf_type`=1,3 this is the slope from $m_{\rm cut} - 0.5M_{\odot}$, for `imf_type`=0 it is the slope from $0.08 - 1.0M_{\odot}$ and for `imf_type`=2 it is the slope from $m_{\rm cut} - 1.0M_{\odot}$. Log IMF weight for the non-parametric IMF, when `imf_type`=4.

- `imf2` Intermediate-mass IMF slope, from $0.5 - 1.0M_{\odot}$, for `imf_type`=1,3. Log IMF weight for the non-parametric IMF, when `imf_type`=4. Otherwise not used.

- `imf3` IMF cutoff, $m_{\rm cut}$, for `imf_type`=2,3. Log IMF weight for the non-parametric IMF, when `imf_type`=4. Otherwise not used.

- `imf4` Log IMF weight for the non-parametric IMF, only used when `imf_type`=4.

- `logfy` Log of the fraction by mass in a young component.

- `fy_logage` Age of the young component.
- `logm7g` Log fraction of light at $1\mu m$ (relative to a 13 Gyr $Z_\odot$ SSP) contributed by an M7III giant star (over and above the nominal predictions from the isochrones).
- `loghot` Log fraction of light at $1\mu m$ (relative to a 13 Gyr $Z_\odot$ SSP) contributed by a hot star (over and above the nominal predictions from the isochrones) whose temperature is specified by `hotteff`. This parameter is supposed to mimic the effects of hot HB stars, but has not been extensively tested in a long time.
- `hotteff` Temperature in Kk of the additional hot star component.
- `sigma2` Velocity dispersion of the emission lines.
- `velz2` Offset in velocity of the emission lines relative to the continuum.
- `logemline_h` Log emission line strength of the Hydrogen lines Hα, Hβ, Hγ, Hδ. The line ratios assume no dust and Case B recombination.
- `logemline_oiii` Log emission line strength of the [OIII] doublet (relative strengths of the doublet adopted from Cloudy models).
- `logemline_sii` Log emission line strength of the [SII] doublet (relative strengths of the doublet adopted from Cloudy models).
- `logemline_ni` Log emission line strength of the [NI] line.
- `logemline_nii` Log emission line strength of the [NII] doublet (relative strengths of the doublet adopted from Cloudy models).
- `jitter` The jitter term is a multiplicative factor applied to the observed errors.
- `logtrans` Log of the strength of the atmospheric transmission function. Currently the H_2O and O_2 lines are tied together, but future developments will allow these two components to vary separately. The transmission function is a model courtesy of Russell Smith, computed from the code available here: `http://rtweb.aer.com/`.
- `logsky` Log of the factor by which the error on the data are inflated around sky lines.

1.6 How to specify priors on the parameters

The upper and lower priors on each parameter are specified in the routine `set_pinit_priors.f90`. These priors can be modified at the beginning of `alf.f90` by e.g., `prhi%logm7g=-5.0`. This example effectively removes the parameter `logm7g` from the fit because the upper limit on the prior is such a small value. Be careful not to set the upper limit to a value that is lower than the lower limit (or visa versa!).

1.7 How to interpret the outputs

The velocity dispersion, σ_a, is simply the best-fit broadening applied to the models after instrumental broadening is included. Note that the models are already smoothed to a dispersion of 100 km/s. Therefore, if instrumental resolution is included in the input file, then the final dispersion of the source, σ_f is $\sqrt{\sigma_a^2 + 100^2}$. If instrumental broadening is not included in the input file, then instrumental broadening must be subtracted in quadrature from the above expression. Moreover, if instrumental broadening is comparable to or larger than 100 km/s and is not constant in km/s as a function of wavelength, then one should definitely include instrumental broadening in the input file!

Elemental abundances. TBD

When fitting in the full mode (`fit_type`=0) a two age component model is used for the SFH. The resulting parameters (`logage`, `logfy`, `fy_logage`) can be used to compute the mass-weighted age, *not* the light-weighted age.

Always be aware of what priors are being adopted and whether or not the posteriors on a parameter are running up against the upper/lower limit on the prior.

1.9 Ancillary programs

`write_a_model.exe` This program is a general purpose routine for computing models given a set of parameters. It includes noise (in the form of a constant S/N per Å), and prints a file that is ready to be directly fit by `alf`. There is also the option to write many identical models to file, in which different realizations of the noise are created. At present this routine is not terribly user-friendly, so email me if you have questions.

`spec_from_sum.exe` This program requires as input a results file (without the ".sum") and returns the corresponding spectrum. This is useful, even though the *.bestspec file also includes the best-fit model, because the output in *.bestspec only extends over the fitted wavelength range. This program also makes it easy to change single aspects of a best-fit model (e.g., turn off IMF variation, change the resolution).

1.9 Acknowledgements

ALF has been in active development since 2011 and has been supported by Packard and Sloan foundation fellowships, NASA grants NNX14AR86G and NNX15AK14G and NSF grant AST-1524161. CC thanks Dan Foreman-Mackey for porting `emcee` to Fortran 90, Ben Johnson for parallelizing `alf` , Alexa Villaume for generating new grids of SSPs, and Bob Kurucz for continually improving the atomic and molecular line lists without which the theoretical response functions would not exist.