Spectral fitting with STARLIGHT

Roberto Cid Fernandes
Universidade Federal de Santa Catarina, Florianópolis, Brazil
April 11, 2007

Abstract

This document describes the public distribution of the STARLIGHT spectral synthesis code, version 04, available from www.starlight.ufsc.br.

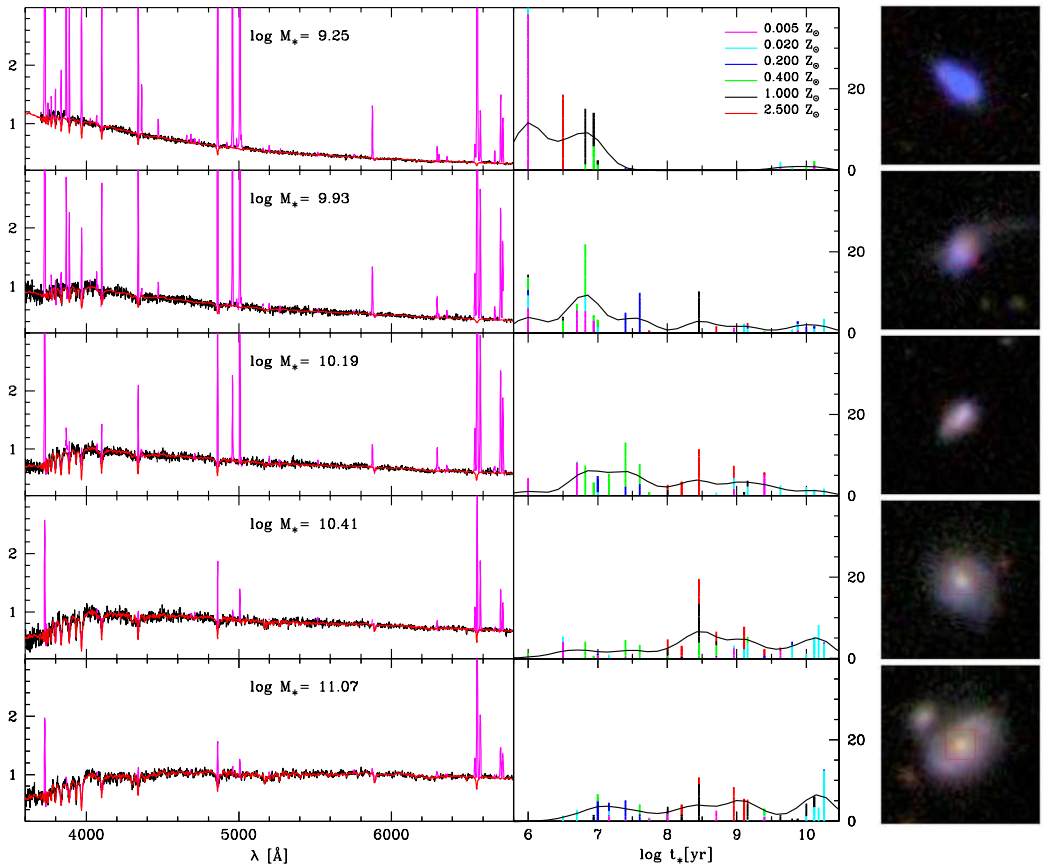

Figure 1: Five examples of STARLIGHT fits. *Left:* Observed (black) and fitted (red) spectra, with masked regions plotted in magenta. *Middle:* Fraction of light at $\lambda_0 = 4020$ Å associated to each of the 25 SSP ages used in the fits, color-coded by the SSP metallicity. Curves represent a 0.5 dex smoothed version of the Star Formation History. *Right:* SDSS images. From Asari et al. (2007).

Contents

1 Introduction

STARLIGHT is a Fortran 77 program to fit an observed spectrum O_λ with a model M_λ which adds up $N_\star$ spectral components from a pre-defined set of base spectra. The spectral base can be made up of observed template spectra (eg, Cid Fernandes et al. 2004a), evolutionary synthesis models (eg, Cid Fernandes et al. 2004b, 2005, 2007; Mateus et al. 2006; Martins 2005; Corbin et al. 2006; Asari et al. 2007), individual stars (say, for velocity dispersion estimates, as in Garcia-Rissman et al. 2005; Vega 2004; Barbosa et al. 2006) or whatever else might be relevant for your specific application. So far, all applications were restricted to the $\sim$ 3500–9000 Å range, though in principle other ranges can be tried.

STARLIGHT is by no means the smarter, faster, better or more elegant way of fitting a spectrum. In fact, its a rather slow, brute-force method with not few limitations, but with a nice virtue: it works! Excellent results have been achieved for objects as diverse as Low Luminosity AGN, Seyfert nuclei, normal galaxies (from ellipticals to spirals and extreme cases such as ultra compact blue dwarf galaxies). You may use it to derive properties of the stellar population mixtures, or just to produce a stellar-template to aid emission line measurements, or even to estimate velocity dispersions. STARLIGHT is designed to be as general as possible, which implies a certain degree of complexity. Even users with experience in population synthesis should take some time to adapt their data and get it running, so take a deep breath and/or a fresh cup of coffee and read on.

This document tells you how to use STARLIGHT. It is meant as a "not-so-quick"-start guide. Beware that **the text is written "upside-down"**: We first explain how you should prepare/format your data and auxiliary files (§4). Then we explain how to run STARLIGHT (§5), and how to digest its output file (§6). Only in §7 we explain what it actually does and how. Example fits (using data accompanying this distribution) are shown in §9. Many details are just not covered here yet, and probably will never be, as by the time we find the time to explain them the code will be altogether different... In any case, judging by the many requests for a version of STARLIGHT, this distribution should hopefully be useful.

2 Download things & test

Unfortunately we cannot distribute the source code at present, though we plan to do it in the future. Meanwhile, look up **www.starlight.ufsc.br** and find a compiled version which will (hopefully) work for your system. Besides the executable file, download Starlight_v04.Distrib.tar.bz2 and unpack it with

```
tar xvfj StarlightChains_v04.Distrib.tar.bz2
```

This will create the directory STARLIGHTv04/, where you will work. Notice that it includes a sub-directory called BasesDir/, which contains some base spectra (§4.3)[1] Put the executable file there too and **rename** it to **StarlightChains_v04.exe**. ls -l on this dir should give

[1] More complete versions of this dir can be found in our site.

```
-rw-r--r-- 1 cid cid 196098 2006-02-23 19:56 0414.51901.393.cxt
-rw-r--r-- 1 cid cid 236437 2007-04-01 23:01 0414.51901.393.cxt.sc4.C11.im.CAL.BN
-rw-r--r-- 1 cid cid 278017 2007-04-01 23:14 0414.51901.393.cxt.sc4.C11.im.CAL.BS
-rw-r--r-- 1 cid cid 236437 2007-04-01 22:58 0414.51901.393.cxt.sc4.C11.im.CCM.BN
-rw-r--r-- 1 cid cid 278017 2007-04-01 23:38 0414.51901.393.cxt.sc4.C11.im.CCM.BS
-rw-r--r-- 1 cid cid 187909 2006-01-19 15:57 0784.52327.478.cxt
-rw-r--r-- 1 cid cid 236437 2007-04-01 23:02 0784.52327.478.cxt.sc4.C11.im.CAL.BN
-rw-r--r-- 1 cid cid 278017 2007-04-01 23:23 0784.52327.478.cxt.sc4.C11.im.CAL.BS
-rw-r--r-- 1 cid cid 236437 2007-04-01 22:59 0784.52327.478.cxt.sc4.C11.im.CCM.BN
-rw-r--r-- 1 cid cid 278017 2007-04-01 23:58 0784.52327.478.cxt.sc4.C11.im.CCM.BS
-rw-r--r-- 1 cid cid   5190 2007-02-10 11:38 Base.BC03.N
-rw-r--r-- 1 cid cid  16530 2007-02-10 11:38 Base.BC03.S
drwxr-xr-x 2 cid cid   8192 2007-02-11 21:53 BasesDir
-rw-r--r-- 1 cid cid   1550 2007-03-30 01:38 grid_example1.in
-rw-r--r-- 1 cid cid   9386 2007-04-01 22:59 grid_example1.log
-rw-r--r-- 1 cid cid   2512 2007-04-01 13:15 grid_example2.in
-rw-r--r-- 1 cid cid  33862 2007-04-01 23:58 grid_example2.log
-rw-r--r-- 1 cid cid   2377 2007-03-30 23:36 grid_example3.in
-rw-r--r-- 1 cid cid 204996 2007-04-02 00:30 grid_example3.log
-rw-r--r-- 1 cid cid   6413 2006-03-03 12:06 Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN
-r--r--r-- 1 cid cid   5861 2006-01-20 16:24 Mask.0784.52327.478.cxt.sc2.CRAP.gm.BN
-rw-r----- 1 cid cid   1254 2004-06-25 15:33 Masks.EmLines.SDSS.gm
-rw-r----- 1 cid cid  45122 2003-12-21 00:18 n5377x.nuc.txt.DR
-rw-r--r-- 1 cid cid 101887 2007-04-02 00:06 n5377x.nuc.txt.DR.sc4.C11.gm.CAL.BN
-rw-r--r-- 1 cid cid 143467 2007-04-02 00:12 n5377x.nuc.txt.DR.sc4.C11.gm.CAL.BS
-rw-r--r-- 1 cid cid 101887 2007-04-01 23:32 n5377x.nuc.txt.DR.sc4.C11.gm.CCM.BN
-rw-r--r-- 1 cid cid 143467 2007-04-01 23:45 n5377x.nuc.txt.DR.sc4.C11.gm.CCM.BS
-rw-r--r-- 1 cid cid 101887 2007-04-02 00:08 n5377x.nuc.txt.DR.sc4.C99.gm.CAL.BN
-rw-r--r-- 1 cid cid 143467 2007-04-02 00:30 n5377x.nuc.txt.DR.sc4.C99.gm.CAL.BS
-rw-r--r-- 1 cid cid 101887 2007-04-01 23:36 n5377x.nuc.txt.DR.sc4.C99.gm.CCM.BN
-rw-r--r-- 1 cid cid 143467 2007-04-02 00:05 n5377x.nuc.txt.DR.sc4.C99.gm.CCM.BS
-rwxr-xr-x 1 cid cid 157138 2007-04-01 22:57 StarlightChains_v04.exe
-rw-r--r-- 1 cid cid   6894 2007-03-29 02:06 StCv04.C11.config
-rw-r--r-- 1 cid cid   6785 2007-02-18 14:43 StCv04.C99.config
```

To check if things work, edit grid_example1.in and adjust the second line to the full path of BasesDir, eg "/home/arthurdent/STARLIGHTv04/BasesDir/". On the third to fifth lines, write either ./ or the full path to the STARLIGHT dir, eg, "/home/arthurdent/STARLIGHTv04/". Having done that, type

./StarlightChains_v04.exe < grid_example1.in

and running messages should start appearing on your screen. (If they don't it may be because the executable is not right for your OS, so try another one, or because the full path strings are too long, like > 70 characters.) Leave it running and read the rest to find out what you have just done.

3 Conditions of use

STARLIGHT is completely free. We do not require co-authorship in papers nor anything like that, except, of course, for a citation to at least paper describing the public distribution

Cid Fernandes, et al 2007; RevMexAA, in prep.

and the original paper

Cid Fernandes, R.; Mateus, A.; Sodré, L.; Stasińska, G. & Gomes, J. M. 2005, MNRAS, 358, 363

though, strictly speaking, the version of STARLIGHT you have just downloaded is closer to that described in

Mateus, A.; Sodré, L.; Cid Fernandes, R.; Stasińska, G.; Schoenell, W. & Gomes, J. M. 2006, MNRAS, 370, 721

and

Asari, N. V.; Cid Fernandes, R.; Stasińska, G.; Torres-Papaqui, J. P.; Mateus, A.; Sodré, L. & Schoenell, W. 2007, MNRAS, submitted.

so you might want to cite that too. Have a look at the *Publications* link on www.starlight.ufsc.br for more refs. The BAA49 and IAU241 proceedings papers are reasonably didactic, and may be helpful for beginners.

An acknowledgment to our sponsors would also be appreciated:

The STARLIGHT project is supported by the Brazilian agencies CNPq, CAPES and FAPESP and by the France-Brazil CAPES/Cofecub program.

STARLIGHT is completely free, but ... we invite users who have access to decent Linux-machines or clusters to consider the possibility of offering us access to run extensive grids, like the whole SDSS with several different setups (which entails $\sim 10^7$ fits, or ~ 100 yr in a single CPU!). We always run things with the lowest priority (nice -n19), minimizing interference with other people's work, and in a highly automated fashion, grabbing data from our server, running it, sending it back and cleaning the space. The results of such runs will be made public in a VO-environment, so that people can play with, say, the Star Formation History of hundreds of thousands of galaxies modeled in different ways. As of the time of writing, at www.starlight.ufsc.br you can already access results for over **half a million** SDSS galaxies fitted with the Bruzual & Charlot (2003) models, 5 of which are illustrated in Fig. 1. This database, to be fully described in **Schoennel et al. 2007, RevMexAA, in prep**, is undergoing revision, reorganization and redocumentation, but you are welcome to play with it an send us opinions. We emphasize that granting access to your CPUs is by no means a necessary condition for you to use STARLIGHT.

Unfortunately, we do not have the (wo)man-power to offer efficient trouble-shooting help, so do not expect quick replies to emails, but send them anyway. We'll do our best to help.

4 Input data & auxiliary files

To use STARLIGHT you must provide:

1. an **input spectrum**, "arq_obs" (§4.1);
2. a **masks file**, "arq_masks" (§4.2);

3. a **base master-file** ("arq_base") and $N_\star$ associated base spectra (§4.3);
4. a **configuration file**, "arq_config" (§4.4);
5. a **grid file** (§4.5).

This section describes each of these. At the end of each subsection we describe the example files provided, whose names are in **red** throughout this document. Have a look at the example files while reading what they contain. As explained in §4.5, the input spectrum, mask and output files may be stored in different directories, specified in the grid file.

OBS: Here is a little incentive to double your attention while reading the next pages: Experience shows that badly formatted files or inconsistent input information is the most common problem faced by STARLIGHT users. In some cases STARLIGHT will detect what the problem is and complain, but in most cases you are on your own.

4.1 Input spectrum

Your spectra must be a plain ASCII file formatted as follows: The 1st line *may* be a header, in which case it will be skipped. The following lines must contain 4 columns:

$\lambda \quad O_\lambda \quad e_\lambda \quad \mathrm{flag}_\lambda$

where λ is the wavelength in Å, O_λ is the flux in your favorite F_λ or L_λ units (e.g., 10^{-17} erg/s/cm^2/Å or $L_\odot$/Å), e_λ is the error in O_λ (in the same units), and flag_λ is an **integer** which signals if that pixel is good or bad: Pixels with $\mathrm{flag}_\lambda \geq$ **2** will be ignored in the synthesis, i.e, they will be assigned weight $w_\lambda = 0$ in the fits (see eq. 15). Good pixels should be signaled with $\mathrm{flag}_\lambda = 0$ or 1. (Another way to neglect certain pixels in the fit is through the masks file described in §4.2.)

If your 1^{st} line is a header then you should set **Is1stLineHeader** to 1 in arq_config (§4.4), otherwise set it to 0.

4.1.1 What to do when you do not have e_λ & flag_λ

Very often one does not have e_λ and/or flag_λ at hand. If you do not have the error spectrum e_λ, you are encouraged to use your creativity and invent one and plug it in the 3^{rd} column of your file. Similarly, if you do not have a flag spectrum at hand, you may just fill in 0's or 1's in the 4^{th} column, or create a proper flag_λ signalling bad pixels with values ≥ 2.

To skip these error and flag-filling steps, STARLIGHT also accepts spectra in the following formats:

$\lambda \quad O_\lambda \quad e_\lambda$

when you don't have flag_λ, and

when neither e_λ nor flag$_\lambda$ are available. The grid file (§4.5) has a couple of On/Off switches (IsFlagSpecAvailable and IsErrSpecAvailable) to inform which of these three formats is to be used. When you do *not* provide e_λ, STARLIGHT defines one from the rms of O_λ in a "S/N λ-range" also defined in the grid file (§4.5). Notice that in this case a flat e_λ spectrum is assumed. Of course, if e_λ is not a proper error, the χ^2 of the fit will not be a proper χ^2. This does not prevent you from achieving a good fit, but try not to err too much. An absurdly small/large e_λ, (by, say, a factor of 1000) may cause numerical problems in the fit. This feature may be handy, but you are still strongly encouraged to invent a less stupid e_λ yourself and plug it in your 3rd column.

OBS 1: STARLIGHT does not accept spectra with flag$_\lambda$ but without e_λ. If this is your case, invent your own e_λ plug it in the 3rd column of arq_obs.

OBS 2: Make sure the IsFlagSpecAvailable and IsErrSpecAvailable (§4.5) are set correctly. If you have a λ O_λ spectrum but say that IsFlagSpecAvailable = 1 and/or IsErrSpecAvailable = 1 then STARLIGHT may suspect something is wrong and issue a warning, but it will not stop!

4.1.2 Pre-processing steps

Before running STARLIGHT, you should:

1. Flux-calibrate your spectrum properly! This version of STARLIGHT **cares about the continuum**, so detector response shapes or other artifacts must all be removed from the data. A future version will deal with rectified spectra[2], but until then make sure you feed the code with calibrated spectra. A decent wavelength calibration is also needed, of course. Absolute flux calibration is not required unless you want to derive things like the stellar mass ($M_\star$) from the spectra alone.

2. Correct O_λ for Galactic extinction. Do not forget to apply this correction in the observed frame and to propagate it to e_λ.

3. Shift the spectrum to its rest-frame (say, using IRAF's dopcor).

4. Sample O_λ, e_λ and flag$_\lambda$ **uniformly** (say, using IRAF's dispcor), say, from 3500 to 8000 Å in steps of $\Delta\lambda = 1$ Å. We recommend working with integer lambdas spaced by $\Delta\lambda = 1$ Å (for instance, 3500, 3501, 3052 ... 7998, 7999, 8000).

5. Write down your spectral resolution ($\sigma_{\rm inst}$, in km/s). You'll need this to correct the output value of the velocity dispersion $\sigma_\star$. STARLIGHT implicitly assumes $\sigma_{\rm inst}$ is the same throughout the spectrum. If this is a very bad approximation in your case (eg, $\sigma_{\rm inst}$ changes by $> \sigma_\star$ across your spectrum) then homogenize $\sigma_{\rm inst}(\lambda)$ directly in your data somehow.

[2]There are ways of using this version to deal with rectified spectra, but they are non-trivial and definitely not very elegant...

4.1.3 Note about sampling

STARLIGHT will re-sample the spectrum from λ_{ini}^{syn} to λ_{fin}^{syn} in steps of $\Delta\lambda^{syn}$. These values are specified in the grid file (see §4.5). Hence, in principle, the resampling pre-processing step described above can be skipped. However, the resampling routine is very simple. The error spectrum, in particular, is resampled as if it were a true spectrum, which is not correct! (For instance, the error in a $\Delta\lambda = 2$ Å bin is not the average of the two corresponding e_λ's in a $\Delta\lambda = 1$ Å spectrum.)

For this and other reasons, it is **strongly recommended that you do your resampling in the pre-processing stage**. STARLIGHT will still call its resampling routine, but by using $\Delta\lambda^{syn}$ equal to the $\Delta\lambda$ of the data you assure that nothing silly is done to e_λ. In fact, in this case the resampling routine will return the exact same data it is fed with (except perhaps for the λ-range), which is the ideal case. It is also recommended working with integer λ's spaced by $\Delta\lambda = 1$ Å. This is not strictly necessary, but that's how STARLIGHT has been used most of the time.

The bottom line is that we need a better resampling routine! Until then, follow these recommendations or else cross your fingers and try your luck... but don't complain!

4.1.4 Example input spectra

Example input spectra with all four columns ($\lambda, O_\lambda, e_\lambda, \text{flag}_\lambda$) are given in files **0414.51901.393.cxt** and **0784.52327.478.cxt**. Both are sampled from 3200 to 9200 Å, in steps of 1 Å. Notice that some of the O_λ values may be ≤ 0, in which case $\text{flag}_\lambda \geq 2$ to inform that the data is corrupted. Here's a piece of **0784.52327.478.cxt**:

```
3200.  0.  0. 99
3201.  0.  0. 99
3202.  0.  0. 99
...
3538.  21.2845993  0. 14
3539.  21.2874699  0. 14
3540.  21.2907429  0. 12
3541.  21.3282471  2.98489189 0
3542.  18.2889023  2.94670773 0
3543.  17.6351337  2.91834164 0
3544.  16.8361225  2.86408353 0
...
8519.  44.0476036  2.41160035 0
8520.  43.8350258  2.41420412 0
8521.  42.4367447  2.40460205 0
8522.  41.4942436  2.54906344 13
8523.  40.9034348  2.81240392 13
8524.  0.  0. 99
8525.  0.  0. 99
8526.  0.  0. 99
...
9198.  0.  0. 99
9199.  0.  0. 99
9200.  0.  0. 99
```

Though the spectrum runs from 3200 to 9200 Å, the first useful ($\text{flag}_\lambda \leq 1$) pixel is $\lambda = 3541$ and the last is 8521 Å.

A spectrum with just λ and O_λ is given in file **n5377x.nuc.txt.DR**. In this case the spectrum runs from 3400 to 5450 Å, also with $\Delta\lambda = 1$ Å. It is not the case in this file, but in principle some of the O_λ values may be ≤ 0. In the absence of a flag_λ to tell you these are bad pixels, STARLIGHT may automatically discard such values with the **f_cut** parameter in arq_config (see §7.2.5), or with the clipping options (§7.2.2).

4.2 Mask file

You should examine your spectrum and identify regions which you do **not** want to model with STARLIGHT, like emission lines, artifacts and holes in O_λ. This can be done with the flag_λ spectrum, if you have one. Another way to mask out some spectral regions is to write them down in the mask file arq_masks.

The 1st line in arq_mask should contain the number of regions to be masked, N_{masks}. Then, in each of the following N_{masks} lines, write

$$\lambda_{ini}^{mask} \qquad \lambda_{fin}^{mask} \qquad w_\lambda^{mask}$$

where $\lambda_{ini}^{mask} \leq \lambda \leq \lambda_{fin}^{mask}$ is the region to be masked and w_λ is the weight to be given to this region. Use $w_\lambda = 0$ to ignore the region. Alternatively, you may use this feature to increase/decrease the weight given to the $\lambda_{ini}^{mask} \rightarrow \lambda_{fin}^{mask}$ range. Normally, $w_\lambda = 1/e_\lambda$, but if you set $w_\lambda^{mask} = 2$ in arq_mask then points between λ_{ini}^{mask} and λ_{fin}^{mask} will be given twice as much weight as normal (i.e, $w_\lambda = 2/e_\lambda$, and so forth). I've seen cases where the CaII K line is not well fitted unless you give it a large weight... We try to avoid playing this game of "give more weight to things you care more about", but sometimes you just can't resist![3]

OBS: You may not need to mask anything, but STARLIGHT still needs a mask file! In this case just write a one-line mask file with $N_{masks} = 0$.

4.2.1 Example mask files

The file **Masks.EmLines.SDSS.gm** provides a general mask containing the most notorious optical emission lines. This is the file we used in our study of 50362 SDSS galaxies in Cid Fernandes et al. (2005). (In that work we further used the SDSS flag_λ spectrum to mask bad pixels.) Here is a piece of it:

```
17
3710.0  3744.0  0.0  [OII]          3726+3729 emission line
```

[3]For this trick to be effective you should gauge the extra weight given to a window n pixels wide by something of order n/n_{tot}, such that your special window will have $\sim$ the same weight as the rest of the spectrum. Also, the parameter **wei_nsig_threshold** (discussed in §7.2.2) should be set to 0 in arq_config, otherwise STARLIGHT will think the weights are suspiciously large and undo your extra weighting.

```
3858.0  3880.0  0.0  [NeIII]        3869 emission line
3960.0  3980.0  0.0  Hepsilon       3970 emission line (over CaII H)
4092.0  4112.0  0.0  Hdelta         4102 emission line
...
```

You may mask the whole $\lambda \leq 3700$ Å blue end of your spectrum adding a line like this:

```
0.0     3700.0   0.0   Bad data!
```

and similarly for the red end or eventual gaps/problematic parts of O_λ. This is useful when flag_λ is not available.

In more recent papers (eg, Mateus et al. 2006; Stasinska et al. 2006; Cid Fernandes et al. 2007; Asari et al. 2007) we have constructed individual masks tailored to each object. Files **Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN** and **Mask.0784.52327.478.cxt.sc2.CRAP.gm.BN** are examples of individual masks for the input spectra **0414.51901.393.cxt** and **0784.52327.478.cxt**, respectively.

OBS: Only N_{masks} and the first three columns of lines 2 to $N_{masks}+1$ are read from these files. All the rest are comments which you may erase/ignore. Most STARLIGHT files come with such unnecessary but harmless extras.

4.3 Base files

The base is the essence of your model. It tells STARLIGHT what are the spectra to be linearly combined when trying to fit your data, so choose them well! Here is how you should organize your base files.

4.3.1 Base Master file

The base master-file arq_base must start with the number of base elements ($N_\star$) in its 1st line. ($N_\star$ must be $\leq$ **300**.) The next $j = 1 \ldots N_\star$ lines must contain:

1. Col 1: arq_j = the file which contains the spectrum $l_{\lambda j}$ of component j.
2. Col 2: The age t_j (in yr) of component j.
3. Col 3: The metallicity Z_j of component j. We usually work with the mass-fraction notation for Z, where $Z_\odot = 0.02$, but you may choose other system.
4. Col 4: A ≤ 15 characters-long nick-name for component j.
5. Col 5: The fraction $f_{\star,j}$ of the initial stellar mass which is still in stars at age t_j for component j (ie, discounting the mass returned to the ISM by SNe, winds, ...). $f_{\star,j}$ should be > 0 and ≤ 1.

6. Col 6: The "YAV_flag". Components with YAV_flag = 0 will be extincted by A_V, whereas those with YAV_flag = 1 will be allowed to have V-band extinctions of $A_V + A_V^Y$. If you want to do something not completely unlike Charlot & Fall (2000), set YAV_flag = 1 for components of $t_j \leq 10^7$ yr and = 0 for all the others. If all components have YAV_flag = 0 then STARLIGHT will fit a single value of the extinction (A_V). I suggest always starting with YAV_flag = 0 for all components.

   Both A_V and A_V^Y are limited between lower & upper values specified in arq_config (§4.4). Beware that A_V^Y will not be constrained at all if the YAV_flag = 1 components have negligible flux. For instance, $A_V^Y = 2$ mag means nothing if the components which are allowed to have this extra extinction are not present or have very low flux ($x_j \sim 0$)! The opposite may also happen, ie, your spectrum is heavily dominated by the YAV_flag = 1 components, in which case you may not be able to distinguish between A_V and A_V^Y, and all you can trust is the total extinction $A_V + A_V^Y$. The bottom line, again, is that you should be very careful when interpreting the results! In fact, we suggest you **do not use this option at all**, or else, **use it with much care!** Dealing with multiple extinctions is a major problem for all synthesis codes, and STARLIGHT is no exception.

7. Col 7: The α/Fe ratio of component j.

OBS: Of all these info, STARLIGHT really **only needs columns 1 and 6**. The ages, metallicities, nick-names and α/Fe columns are only used to produce an organized, informative output (arq_out, presented in §6), while $f_{\star,j}$ is only used in the conversion from light to mass-fractions. **None of these things is actually used in the fit**. Sometimes, these informations are not available, and in other cases they do not even exist! For instance, when you fit a spectrum with a set of template spectra of individual stars, power-laws, nebular-continuum or observed galaxy spectra, things like age or $f_{\star,j}$ are just not definable. Worry not, but make sure you fill in the fields.

4.3.2 Base spectra files

The individual base spectra (arq_j) must have λ in its 1st column and $l_{\lambda j}$ in the 2nd column. Header lines, if present, must start with a "#" in the 1^{st} column.

All that is required from these spectra is that they come in an λ-increasing order, and that **all base spectra be sampled in exactly the same way**. (Do not mix an arq_j with 6000 λ's with an arq_k with 2000 λ's ...) STARLIGHT does the necessary resampling and trimming. Yet, beware that funny things may happen if, for instance, your base is sampled at $\Delta\lambda = 20$ Å from 4000 to 5000 Å and at $\Delta\lambda = 1$ Å from 5000 to 7000 Å, and you try to fit the whole 4000–7000 Å range.[4] The kinematical parameters may go bananas in this case. (Such a transition in sampling occurs, for instance, in the high-resolution BC03/STELIB models at 3322 Å.)

The individual base files should be stored in a directory "base_dir" specified in the grid-file. The "**bc2003_hr_*.spec**" files under "STARLIGHTv04/BasesDir" are examples of base spectra. They

[4] It is possible to circumvent these problems fixing the kinematical parameters beforehand using the grid-file option "FXK" explained in §4.5.

were extracted from www.cida.ve/∼bruzual/bc2003.[5]

OBS: base_dir + arq_j should not exceed 100 characters. (If this is a problem create a symbolic link to base_dir.)

• **Units:** For a BC03 base of SSPs, the $l_{\lambda j}$ spectra come in units of $L_\odot/$Å per **initial** $M_\odot$. The spectrum is thus a light-to-mass-ratio spectrum, but to convert it to a light-to-**current**-stellar-mass-ratio we must divide it by $f_{\star,j}$. Again, this is only relevant in the conversion from light to mass fractions, done at the very end of a STARLIGHT run. Users using a base from a different evolutionary synthesis code (eg, PÉGASE, SED, Starburst99, ...) should adopt these same units, or else do their own light-to-mass conversions. Obviously, none of this makes any sense whatsoever for bases made up of observed spectra, like template galaxies, stars or power-laws.

4.3.3 Example Base-files

The file **Base.BC03.N** shows an example base master file which uses BC03 spectra of 3 metallicities and 15 different ages (hence $N_\star = 3 \times 15 = 45$). Values for the "stellar-mass fraction" (column 5) were taken from the bc2003_hr_m*_chab_ssp.4color files in the BC03 distribution. Notice that this example has YAV_flag = 0 for **all** components (column 6). Here is a piece of it:

```
45             [N_base]
bc2003_hr_m42_chab_ssp_020.spec          0.00100e9      0.00400      age020_m42      1.0000     0      0.0000
bc2003_hr_m42_chab_ssp_045.spec          0.00316e9      0.00400      age045_m42      0.9999     0      0.0000
bc2003_hr_m42_chab_ssp_055.spec          0.00501e9      0.00400      age055_m42      0.9488     0      0.0000
bc2003_hr_m42_chab_ssp_070.spec          0.01000e9      0.00400      age070_m42      0.8862     0      0.0000
...
bc2003_hr_m42_chab_ssp_150.spec          2.50000e9      0.00400      age150_m42      0.5463     0      0.0000
bc2003_hr_m42_chab_ssp_161.spec          5.00000e9      0.00400      age161_m42      0.5228     0      0.0000
bc2003_hr_m42_chab_ssp_185.spec         11.00000e9      0.00400      age185_m42      0.4980     0      0.0000
bc2003_hr_m42_chab_ssp_193.spec         13.00000e9      0.00400      age193_m42      0.4927     0      0.0000
...
bc2003_hr_m72_chab_ssp_020.spec          0.00100e9      0.05000      age020_m72      0.9998     0      0.0000
bc2003_hr_m72_chab_ssp_045.spec          0.00316e9      0.05000      age045_m72      0.9577     0      0.0000
bc2003_hr_m72_chab_ssp_055.spec          0.00501e9      0.05000      age055_m72      0.9147     0      0.0000
bc2003_hr_m72_chab_ssp_070.spec          0.01000e9      0.05000      age070_m72      0.8524     0      0.0000
...
bc2003_hr_m72_chab_ssp_150.spec          2.50000e9      0.05000      age150_m72      0.5583     0      0.0000
bc2003_hr_m72_chab_ssp_161.spec          5.00000e9      0.05000      age161_m72      0.5320     0      0.0000
bc2003_hr_m72_chab_ssp_185.spec         11.00000e9      0.05000      age185_m72      0.5070     0      0.0000
bc2003_hr_m72_chab_ssp_193.spec         13.00000e9      0.05000      age193_m72      0.5018     0      0.0000
# spec-file                               age [yr]            Z            code       Mstar    YAV?     a/Fe
...
```

Here is a sample of the spectrum of the 25^{th} element in this base (**bc2003_hr_m62_chab_ssp_135.spec**), extracted from BC03:

```
# Output file name = bc2003_hr_m62_chab_ssp_135.spec
# Input file name  = bc2003_hr_m62_chab_ssp.ised
# Column           2
# Record         135
# Age (yr)      9.048E+08
# Lambda(A)       Flux
  9.100000E+01  1.276E-07
  9.400000E+01  1.442E-07
  9.600000E+01  1.495E-07
...
  3.270000E+03  9.475E-05
  3.290000E+03  9.373E-05
  3.310000E+03  9.430E-05
```

[5]We thank Gustavo Bruzual and Stephane Charlot for allowing us to include their models in this distribution.

```
  3.322000E+03  8.968E-05
  3.323000E+03  8.702E-05
  3.324000E+03  8.956E-05
  3.325000E+03  9.520E-05
  3.326000E+03  9.930E-05
...
  9.297000E+03  9.801E-05
  9.298000E+03  9.810E-05
  9.299000E+03  9.725E-05
  9.300000E+03  9.649E-05
  9.310000E+03  1.011E-04
  9.330000E+03  1.028E-04
...
  1.000000E+06  4.534E-12
  1.200000E+06  2.180E-12
  1.400000E+06  1.173E-12
  1.600000E+06  6.859E-13
```

(Notice the changes in sampling at $\lambda = 3322$ and 9300 Å.)

File **Base.BC03.S** is an example with $N_\star = 150$ BC03 spectra of 6 metallicities and 25 different ages (used in Mateus et al. 2006; Cid Fernandes et al. 2007; Asari et al. 2007).

OBS: This distribution of STARLIGHT includes only the 159 files from BC03 used in **Base.BC03.N** and **Base.BC03.S**. Our site contains many other base spectra and base master-files from BC03 and other sources. Quite a few more should appear during 2007–2008, when several new sets of evolutionary synthesis models are expected to become available.

4.4 The configuration file

This is the harder to explain, as it contains all of the technical parameters which control STARLIGHT. It also specifies things like where to normalize, what are the limits for extinction and kinematical parameters, if deviant points should be clipped and how, ... Each entry is followed by a comment which should hopefully give you some clue as to what that parameter does. The most important of these are described below when we explain what the code does in the next sections, so we'll come back to arq_config later. Meanwhile, have a quick look at any of the *.config files to have a feeling of how scary they look. Parameters in arq_config are signaled with **this color** throughout this manual. (In fact, some have already appeared in this manual.)

4.4.1 Example configuration files

StCv04.C11.config and **StCv04.C99.config** are examples of configuration files. The first is designed to run faster. The latter does many more iterations and loops, and should in principle produce a more detailed fit. **StCv04.C99.config** takes $\sim$ 3 times longer to run and computes $\sim$ 3 times more models, but note these are approximate numbers which vary from case to case, depending how fast convergence is achieved. In practice, this makes little difference in the results or fit quality! In fact, given that STARLIGHT uses random numbers to evolve its Markov Chains, it is possible that a long fit may yield a slightly worse χ^2 than a fast one! The essential features of the fit, however, should be equivalent.

4.5 The grid file

Don't panic (Adams 1979)! You're nearly there. You already know how to prepare your input spectra (§4.1), write a masks file (§4.2) and organize the base files (§4.3). You also learned that there is an ugly and mysterious configuration file (§4.4), to be explained later. The last thing you need to run STARLIGHT is a grid file like grid_example1.in, which starts with a **15-line "header"** informing:

1. The number of spectra to be fitted (N_{fits}).
2. The base directory string base_dir, which says where the base spectra are located.
3. The obs directory string obs_dir, which says where the arq_obs input spectra to be fitted are located.
4. The mask directory string mask_dir, which says where the arq_masks mask files are located.
5. The out directory string out_dir, which says where the arq_out output STARLIGHT files are to be stored.
6. An integer seed for the random number generator. Your phone number will do it. (If you use 0, the **same** seed will be used for every one of the N_{fits}, which may be useful if you are testing the effects of, say, different configuration files.)
7. llow_SN, and
8. lupp_SN. Fluxes in the $\lambda =$ llow_SN $\rightarrow$ lupp_SN range will be used to define a "S/N" ratio (= rms/mean flux). If you do *not* provide an error spectrum e_λ, the rms of O_λ in this window will be taken as a measure of the noise and used as e_λ (see §4.1). If you do provide e_λ, then a S/N will still be computed in the llow_SN $\rightarrow$ lupp_SN range and reported in arq_out, but it will *not* be actually used for anything but output purposes. Masked and flagged pixels are ignored in the computation of S/N.
9. λ^{syn}_{ini},
10. λ^{syn}_{fin}, and
11. $\Delta\lambda^{syn}$ define the lower and upper wavelengths to be synthesized and the sampling, all in Å. This range can exceed that of your data or vice-versa, but make sure data and model overlap somewhere! If you tell STARLIGHT to model λ's beyond the range spanned by your base, these λ's will be flagged as bad and ignored. As already said, we recommend $\Delta\lambda^{syn} = 1$ Å.
12. A cooking factor which should be set equal to 1 unless everything else fails. (It is a multiplicative factor applied to χ^2, only useful when your errors are severely wrong, and even then it should not be really used! Fix your e_λ instead.)
13. A fitting option string which should be set to 'FIT' or 'FXK'. The latter option will freeze in attempts to fit kinematical parameters ($v_\star$ & $\sigma_\star$). Hence, if you chose 'FXK', $v_\star$ and $\sigma_\star$ will be kept fixed at their starting values (see below) throughout the fits. Otherwise, with 'FIT' STARLIGHT will try to find the values of $v_\star$ and $\sigma_\star$ which produce a best match.

14. A 1/0 = Yes/No switch to inform whether your data file has an error spectrum or not.

15. A 1/0 = Yes/No switch to inform whether your data file has a flag spectrum or not.

After this general header, each of the following N_{fits} lines should contain:

arq_obs arq_config arq_base arq_masks red_law_option v0_start vd_start arq_out

• The arq_*'s have been explained above, except for arq_out, which is the corresponding output file name. It is convenient to construct output filenames by appending to the input data filename suffixes informing the reddening law, base, configuration and mask files used, phase of the moon, etc.

• red_law_option (column 5) is a 3-characters string (like 'CCM' or 'CAL') which specifies which reddening law you want to use. See §4.6 for the options.

• v0_start and vd_start are first guesses for the velocity shift (v0 = $v_\star$) and velocity dispersion (vd = $\sigma_\star$), both in km/s. We normally use v0_start = 0 and vd_start = 150 when we know nothing about kinematics beforehand and want to fit it. In this case, the fit-option string in line 13 of the header should be 'FIT'. STARLIGHT usually does a very good job in measuring v0 and vd, but sometimes (eg, HII galaxies) a spectrum has few absorption lines where to anchor v0 and vd estimates, which may then go wild, since all there is to fit in this case is the continuum shape. If you want to keep the kinematical parameters fixed during the fit, plug your favorite values in columns 6 & 7 and set the fit-option string to 'FXK'. If you want v0 and/or vd to be constrained to within certain limits, use 'FIT' and adjust the **v0_low**, **v0_upp**, **vd_low** and **vd_upp** limits in arq_config.

In case you are wondering why on earth (or elsewhere, for that matter) we should allow for a v0 if the spectrum is supposed to be in the rest frame (§4.1), think of v0 as a fine-tuning for slight errors in your de-redshifting process. Beware that STARLIGHT is not prepared to handle very badly de-redshifted data (error > 500 km/s). For well de-reshifted spectra, v0 will usually be a small number, like ±5–20 km/s. For long-slit or IFU galaxy spectra, you may have de-redshifted the spectra using the nuclear value. In this case, the output values of v0 can help you derive a rotation curve, as well as vd maps (eg., Barbosa et al. 2006). A way of effectively turning off v0-fitting is to limit its low & upp limits in arq_config to a small range. Conversely, if you are in interested in v0 but do not have a base with enough resolution to measure vd, set **vd_low** = **vd_upp** = 0 in arq_config.

IMPORTANT NOTE ABOUT KINEMATICS: The v0 and vd values are the shift and broadening parameters applied to the model which best fits the data. Hence, v0 and vd are only really equal to $v_\star$ and $\sigma_\star$ **if**: (1) your base spectra are in the rest-frame, and (2) your base and data have the same spectral resolution. If your spectral resolution is $\sigma_{\rm inst}$ and the base has $\sigma_{\rm base}$, do

$$\sigma_\star^2 = {\rm vd}^2 - \sigma_{\rm inst}^2 + \sigma_{\rm base}^2 \qquad (1)$$

to correct vd to a proper velocity dispersion. Don't expect reliable estimates when $\sigma_\star < \sigma_{\rm base}$ or $\sigma_\star < \sigma_{\rm inst}$, ie, when the true velocity dispersion is smaller than the resolution of the base or your

data. For instance, a very high resolution spectrum ($\sigma_{\rm inst} \sim 0$) of a dwarf galaxy or a star cluster modeled with a base with $\sigma_{\rm base} \sim 100$ km/s, you should get vd ~ 0. This means that STARLIGHT did not find it necessary to broaden the model spectrum to improve the fit. If this is a serious problem for you (can't see why), then smooth your data to $\sim \sigma_{\rm base}$ before fitting it. See §4.7 for other subtleties on v0 and vd estimates.

4.5.1 Example grid files

Files **grid_example1.in**, **grid_example2.in** and **grid_example3.in** are examples of grid files. Here is **grid_example1.in**, which fits 2 SDSS spectra (with both e_λ and flag$_\lambda$) from 3400 to 8900 in steps of 1 Å, using config parameters from StCv04.C11.config, a Base.BC03.N base, individual mask files, a CCM reddening law, and guessing v0 = 0, vd = 150 km/s to start:

```
2                                       [Number of fits to run]
/home/cid/STARLIGHTv04/BasesDir/        [base_dir]
/home/cid/STARLIGHTv04/                 [obs_dir]
/home/cid/STARLIGHTv04/                 [mask_dir]
/home/cid/STARLIGHTv04/                 [out_dir]
-2007200                                [your phone number]
4730.0                                  [llow_SN]   lower-lambda of S/N window
4780.0                                  [lupp_SN]   upper-lambda of S/N window
3400.0                                  [Olsyn_ini] lower-lambda for fit
8900.0                                  [Olsyn_fin] upper-lambda for fit
1.0                                     [Odlsyn]    delta-lambda for fit
1.0                                     [fscale_chi2] fudge-factor for chi2
FIT                                     [FIT/FXK] Fit or Fix kinematics
1                                       [IsErrSpecAvailable]  1/0 = Yes/No
1                                       [IsFlagSpecAvailable] 1/0 = Yes/No
0414.51901.393.cxt   StCv04.C11.config  Base.BC03.N   Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN   CCM   0.0   150.0   0414.51901.393.cxt.sc4.C11.im.CCM.BN
0784.52327.478.cxt   StCv04.C11.config  Base.BC03.N   Mask.0784.52327.478.cxt.sc2.CRAP.gm.BN   CCM   0.0   150.0   0784.52327.478.cxt.sc4.C11.im.CCM.BN

Example of 2 SDSS galaxies fitted with Base.BC03.N (45 components), CCM law, etc.
Cid@Lagoa - 29/March/2007
```

grid_example2.in is similar to **grid_example1.in** (in fact, they could be merged in a single grid file). It changes the base file and/or reddening law option, but fits the same 2 galaxies.

grid_example3.in is different. It fits a spectrum with no e_λ nor flag$_\lambda$ (thus IsErrSpecAvailable = 0 and IsFlagSpecAvailable = 0), uses a general mask, a smaller λ-range and tries 2 different config files, including a "slow" one (StCv04.C99.config).

OBS: In all grid file examples given, the arq_obs, arq_mask and arq_out files are in the same directory (STARLIGHTv04/). Change obs_dir, mask_dir and out_dir as you will, but **make sure that the full path + file name never exceeds 100 characters**. (Again, use symbolic links to bypass this limit.)

4.6 Reddening-law options

STARLIGHT has a suite of reddening-laws options, which the user chooses through the 3-byte-long string red_law_option informed in the grid file. Some options were shamelessly "borrowed" from the HyperZ code (webast.ast.obs-mip.fr/hyperz/). The options are:

CCM - Cardelly, Clayton & Mathis (1989, with $R_V = 3.1$). Beware of this curve for large extinctions! (We've seen its 7^{th} order polynomial do weird things with ULIRG spectra...)

CAL - The "Calzetti law"

GD1 - Gordon et al. (2003) SMC Bar (interpolated from their Table 4)

GD2 - Gordon et al. (2003) LMC2 Super-Shell (interpolated from their Table 4)

GD3 - Gordon et al. (2003) LMC Average (interpolated from their Table 4)

HZ1 - HYPERZ = Allen (1976), from HyperZ

HZ2 - HYPERZ = Seaton (1979), from HyperZ

HZ3 - HYPERZ = Fitzpatrick (1986) LMC, from HyperZ

HZ4 - HYPERZ = for Prevot et al. (1984) and Bouchet et al. (1985) SMC, from HyperZ

HZ5 - HYPERZ = Calzetti (astro-ph/9911459), from HyperZ

*** **Your own law:** If you'd rather define your own extinction law, prepare a file with two columns, λ and A_λ/A_V (just that, no headers, footers, etc), and call it **ExtinctionLaw.*****, where "***" is a 3-byte-long extension of your liking. Choose a name for *** and use it as the 3-character-long red_law_option string to be given in the grid-file. Place the file in the same directory as your executable. Use this option with care, and make sure your λ's cover all the range of the data you want to model (preferably more). STARLIGHT interpolates linearly in λ^{-1} from the table. Don't worry about having too many points in this file, as the reddening array (sampled as your model) is only computed once.

4.7 Subtleties about λ ranges and the kinematical filter

STARLIGHT **needs** a safety cushion on the edges of your data to deal with kinematical parameters (v0 & vd). The size of this cushion in Å is set by **dl_cushion** in arq_config (we usually use 50 Å). This means that your base spectra should be defined at least from λ_{ini}^{syn} − **dl_cushion** to λ_{fin}^{syn}+ **dl_cushion**. If you do not observe this, then the kinematical kernel may go bananas with edge effects.

Example: If your base spectra go from 4000 to 7000 Å and you set **dl_cushion** = 50, the maximum range to be fitted is $\lambda_{ini}^{syn} = 4050$ to $\lambda_{fin}^{syn} = 6950$ Å. The points between 4000–4050 and 6950–7000 will be used to compute a model and in the convolution, but will be automatically masked from the fits (ie, they will not be compared to the data because at least some of them will be badly affected by edge effects.) If you forget this detail, STARLIGHT will issue a warning and mask out points outside the 4050–6950 anyway.

Dealing with kinematics slows things down quite a bit. By the time STARLIGHT gets to the last stage of the fitting process (the EX0-phase, explained later), kinematical parameters have probably converged. In this case, you may choose to fix them setting **IsNoKin4LargeBaseInEX0sFits** = 1 and **frac_NoKin4LargeBaseInEX0sFits** = 0 in arq_config. If you are more interested on the kinematics than on the populations, do the opposite, such that kinematical parameters will be re-fitted all the way.

5 Running STARLIGHT

In the unlikely case you have already understood everything and prepared your own input and auxiliary files, go ahead and try your luck. Otherwise, run the grid examples provided. Do this with

./StarlightChains_v04.exe < grid_example1.in

or

nice -n19 ./StarlightChains_v04.exe < grid_example1.in > grid_example1.log &

if you want to save the stuff which goes to the screen and be nice to whoever else is using your CPU. Here is what the screen output for the 1st run in **grid_example1.in** looks like:

```
**> Welcome to STARLIGHT (v04) & good luck with this run!

**> Read config file: StCv04.C11.config
**> Read data file: 0414.51901.393.cxt
    from dir = /home/cid/STARLIGHTv04/
    Data goes from:   3200.0  -->   9200.0 Angs with N =  6001 pixels
    STARLIGHT has read an ERROR (column 3) from 0414.51901.393.cxt
    STARLIGHT has read an FLAG  (column 4) from 0414.51901.393.cxt
**> Read base files: Base.BC03.N                                N_base  =  45
    from dir = /home/cid/STARLIGHTv04/BasesDir/
**> Read masks file: Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN      N_masks =  39
    from dir = /home/cid/STARLIGHTv04/
**> Modeling 0414.51901.393.cxt            ==> 0414.51901.393.cxt.sc4.C11.im.CCM.BN
    Sampling: 3400.0 ==> 8900.0  with dl =   1.0 Angs  & fscale_chi2 = 1.000E+00
    dl_cushion (A) =   50.0
    Extinction Law = CCM
    Initial kinematical parameters: v0 =    0.0 & vd =  150.0 km/s
    We are going to FIT the spectrum of this galaxy
    Number of components with different extinction =   0  ( 0.00 < YAV <  0.00)
    S/N     =   24.912    32.739     in S/N & norm. windows
    S/N_err =   16.546    21.744     in S/N & norm. windows
********************************************************************************
* STARTING FIRST FITS ...
* N_loops  =   3
* N_chains =   7
* Temp     = 1.00E+02 ==> 1.00E+00
* GR-R     = 1.30E+00 ==> 1.30E+00  Method =  0  (0/1=Soft/Hard)
********************************************************************************
********************************************************************************
* STARTING RE-FIT AFTER CLIPPING ...
* Clipped     3 points with > 3.0 sigma residuals with method = NSIGMA
* N_loops =   1
* Temp     = 1.00E+00 ==> 1.00E+00
* GR-R     = 1.30E+00 ==> 1.30E+00  Method =  0  (0/1=Soft/Hard)
********************************************************************************
********************************************************************************
* STARTING FINAL BURN-IN LOOP...
* N_loops =   1
* Temp     = 1.00E+00 ==> 1.00E+00
* GR-R     = 1.20E+00 ==> 1.20E+00  Method =  0  (0/1=Soft/Hard)
********************************************************************************
********************************************************************************
* STARTING EX0s FITS ...
* EX0_method     =   CUMUL
* EX0s_Threshold =   0.020
* EX0s_PopVector =     MIN
* N_loops =   5
* Temp     = 1.00E+00 ==> 1.00E-03
* GR-R     = 1.20E+00 ==> 1.00E+00  Method =  1  (0/1=Soft/Hard)
********************************************************************************

**> RESULTS for 0414.51901.393.cxt.sc4.C11.im.CCM.BN
   N_tot chi2/N1_eff    adev |     v0    vd | sum       A_V  Y_AV |     x_j... [%], j = 1 ... 45
 1225140 6.5669E-01   4.7752 |  -22.9  42.5 | 101.606  0.26  0.00 |   11.84   1.60   1.54   4.27   0.00  18.13   9.83   6.98   6.35   9.13
                                                                     0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00  10.10   0.00
                                                                     0.00   0.00   0.00  11.95   0.00   0.00   0.00   0.00   0.00   0.00
```

```
 0.00   0.00   1.07   1.41   0.00   4.39   0.00   0.00   3.00   0.00
 0.00   0.00   0.00   0.00   0.00
Total chi2 =   2.7292E+03  for Nl_eff =  4156 lambdas!
```

When you do something really silly or provide inconsistent input, the screen output may tell you what you've done wrong. Otherwise, screen output is not useful for you, so you might as well redirect it to > /dev/null. There are verbose-level options in arq_config.

When you're done, repeat the commands with **grid_example2.in** and **grid_example3.in**.

Be patient! Each run may take several minutes. Nobody ever said STARLIGHT is fast! grid_examples 1, 2 and 3 take $\sim$ 2, 30 and 30 minutes in a 2–3 GHz CPU, respectively.

6 STARLIGHT output

All synthesis results are stored in arq_out in a $\sim$ self-explanatory way. The length of the output file depends on the size of the base ($N_\star$) and the length of the model spectrum M_λ. arq_out has an extensive "header", which, besides output, repeats most of the input info for completeness. Here is a brief tour through an example.

• The first block should look like this (example taken from my run of **0414.51901.393.cxt.sc4.C11.im.CAL.BN** in **grid_example2.in**):

```
## Some input info
0414.51901.393.cxt                          [arq_obs]
Base.BC03.N                                 [arq_base]
Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN      [arq_masks]
StCv04.C11.config                           [arq_config]
   45                                       [N_base]
    0                                       [N_YAV_components = # of components with extra extinction!]
    0                                       [i_FitPowerLaw (1/0 = Yes/No)]
-9.99                                       [alpha_PowerLaw]
CAL                                         [red_law_option]
 1.355739                                   [q_norm = A(l_norm)/A(V)]
```

As you can see, it tells you nothing you did not know already, with the possible exception of q_norm, which is the value of q_norm $= q_{\lambda_0} = A_{\lambda_0}/A_V$ for your choice of extinction curve.

• The block

```
## (Re)Sampling Parameters
 3400.00                                    [l_ini (A)]
 8900.00                                    [l_fin (A)]
    1.00                                    [dl    (A)]     (OBS: dl_cushion =    50.00 A)
```

repeats the λ_{ini}^{syn}, λ_{fin}^{syn} and $\Delta\lambda^{syn}$ info given in the grid file. The only reason I mention it here is because of the value of **dl_cushion**, an important parameter set in arq_config which we explained in §4.7. (Hopefully it will not be relevant/critical for you.)

• The fragment below repeats the normalization λ's given in arq_config and, more importantly, gives you the flux by which your spectrum was divided, **fobs_norm**. You will need this value to denormalize the fit.

```
## Normalization info
 4020.00                                  [l_norm (A) - for base]
 4010.00                                  [llow_norm (A) - window for f_obs]
 4060.00                                  [lupp_norm (A) - window for f_obs]
1.655215E+01                              [fobs_norm (in input units)]
```

• The next block informs the computed S/N (= mean/rms flux) in the S/N and normalization windows. (The S/N window is informed in the grid file while the normalization window is specified in arq_config.)

```
## S/N
 4730.00                                  [llow_SN (A) - window for S/N]
 4780.00                                  [lupp_SN (A) - window for S/N]
  24.912                                  [S/N in S/N window]
  32.739                                  [S/N in norm. window]
```

These lines are followed by two corresponding "S/N_err" values, which are S/N estimates computed with the error spectrum (if you have one). Recall that if you have provided an error spectrum, **none** of these numbers were really used in the fit.

• In the etc-block which follows, you may be interested in

```
  4159                                    [NOl_eff]
  4156                                    [Nl_eff]
     3    NSIGMA                          [Ntot_cliped & clip_method]
  1387680                                 [Nglobal_steps]
   7                                      [N_chains]
  21                                      [NEXOs_base = N_base in EXOs-fits]
```

which tell you what is the number of good pixels (i.e., unmasked & unflagged) in your original data, the number which survived clipping and the number of clipped pixels (along with your chosen **clip_method_option**, specified in arq_config). Nglobal_steps tells you the total number of models tried throughout the fit. **N_chains** repeats the number of Markov Chains you chose in arq_config, and NEX0s_base tells how many of your original $N_\star$ base components were actually used in the fit. The line

```
    78     67.020     66.848      0.172  [idt_all, wdt_TotTime, wdt_UsrTime & wdt_SysTime (sec)]
```

tells you $\sim$ how long the fit took to run (wdt_TotTime is the better CPU time-counter.)

• The **"Synthesis Results - Best model"** block **is the most important one.** It starts with

```
## Synthesis Results - Best model ##

 6.55432E-01                                [chi2/Nl_eff]
     4.76778                                [adev (%)]

   101.77190                                [sum-of-x (%)]
 2.35170E+01                                [Flux_tot (units of input spectrum!)]
 3.65062E+04                                [Mini_tot (???)]
 2.21600E+04                                [Mcor_tot (???)]
```

where chi2/Nl_eff is the fit χ^2 divided by the number of λ's used in the fit, and adev gives you the percentage mean $|O_\lambda - M_\lambda|/O_\lambda$ deviation over all fitted pixels. Both measure the quality of the fit.

Next comes sum-of-x $= \sum x_j$. Do not be surprised to find that the x_j's do not add up to 100% (more on this later). Flux_tot gives the total dereddened synthetic flux at l_norm (prior to application of the kinematical filter, and in units of fobs_norm), in case you need it. Mini_tot and Mcor_tot are more important. If your spectrum has a reliable absolute flux calibration and your base spectra were SSPs given in proper $L_\odot\,\text{Å}^{-1}M_\odot^{-1}$ units, these values allow you to compute galaxy masses. If your observed spectrum O_λ comes in units of, say, 10^{-17} erg/s/cm^2/Å, do

$$M_\star = \text{Mcor\_tot} \times 10^{-17} \times 4\pi d^2 \times (3.826 \times 10^{33})^{-1} \tag{2}$$

to obtain the present mass in stars (in $M_\odot$). Similarly

$$M_\star^{ini} = \text{Mini\_tot} \times 10^{-17} \times 4\pi d^2 \times (3.826 \times 10^{33})^{-1} \tag{3}$$

will tell you how many $M_\odot$'s have been processed into stars throughout the galaxies life. (This is where the $f_{\star,j}$ values in the base master-file, discussed in §4.3, are used. For a not-too-crazy galaxy and a Chabrier (2003) IMF, $M_\star^{ini} \sim 2M_\star$, due to the mass returned to the ISM by SNe and winds).

Next,

```
 -19.90                                     [v0_min  (km/s)]
  44.92                                     [vd_min  (km/s)]
 0.2672                                     [AV_min  (mag)]
 0.0000                                     [YAV_min (mag)]
```

give you the best fit values of v0, vd, A_V and A_V^Y. Obviously, YAV_min will only have meaning if you ignored doctor's recommendations and dared to use YAV_flag = 1 in your arq_base. Recall that this feature has its own severe existential problems.

Then comes the part which tells you what was the best **population mixture** found by STARLIGHT. For many applications this is the most relevant result. You should be able to derive mean ages, mean metallicities and star-formation histories from it. It looks like this:

# j	x_j(%)	Mini_j(%)	Mcor_j(%)	age_j(yr)	Z_j	(L/M)_j	YAV?	Mstars	component_j	a/Fe...	SSP_chi2r	SSP_adev(%)	SSP_AV	SSP_x(%)
1	15.0005	8.5233E-01	1.4041E+00	1.000000E+06	0.00400	1.114E-02	0	1.0000	age020_m42	0.0000	1.0183E+00	6.1188	2.2985	92.6204
2	7.8143	1.2144E-01	2.0004E-01	3.160000E+06	0.00400	4.073E-02	0	0.9999	age045_m42	0.0000	9.0516E-01	5.6222	1.5142	96.3750
3	0.0000	0.0000E+00	0.0000E+00	5.010000E+06	0.00400	2.534E-02	0	0.9488	age055_m42	0.0000	8.9463E-01	5.5813	1.7901	95.8431
4	3.9879	1.9860E-01	2.8994E-01	1.000000E+07	0.00400	1.271E-02	0	0.8862	age070_m42	0.0000	8.1546E-01	5.3180	0.7523	100.7819
5	0.0032	4.0396E-04	5.4396E-04	2.512000E+07	0.00400	5.070E-03	0	0.8174	age090_m42	0.0000	7.7199E-01	5.1622	0.6343	101.2523
6	3.8530	6.9583E-01	8.9997E-01	4.000000E+07	0.00400	3.505E-03	0	0.7851	age104_m42	0.0000	7.3431E-01	5.0223	0.3965	100.0800
7	9.5516	2.6658E+00	3.1650E+00	1.015200E+08	0.00400	2.268E-03	0	0.7207	age116_m42	0.0000	7.4348E-01	5.0608	0.7737	98.6686
8	8.1674	4.7913E+00	5.2197E+00	2.861200E+08	0.00400	1.079E-03	0	0.6613	age125_m42	0.0000	8.0436E-01	5.2328	0.2201	98.7746
9	2.2559	2.8126E+00	2.8714E+00	6.405400E+08	0.00400	5.077E-04	0	0.6197	age132_m42	0.0000	9.5596E-01	5.7433	-0.1108	99.5684
10	2.3293	4.1708E+00	4.1390E+00	9.047900E+08	0.00400	3.535E-04	0	0.6024	age135_m42	0.0000	1.1538E+00	6.4879	-0.3476	97.3618
...														
36	9.5865	1.8975E+00	2.3813E+00	4.000000E+07	0.05000	3.198E-03	0	0.7618	age104_m72	0.0000	1.5719E+00	7.8272	0.7568	107.1004
37	0.0178	6.5608E-03	7.6652E-03	1.015200E+08	0.05000	1.716E-03	0	0.7092	age116_m72	0.0000	1.1137E+00	6.4566	0.6120	106.1525
38	0.0000	0.0000E+00	0.0000E+00	2.861200E+08	0.05000	7.047E-04	0	0.6618	age125_m72	0.0000	9.4658E-01	5.6301	0.0863	100.7158
39	6.2124	1.4935E+01	1.5385E+01	6.405400E+08	0.05000	2.633E-04	0	0.6253	age132_m72	0.0000	1.6647E+00	7.6342	-0.6874	95.1361
40	0.0000	0.0000E+00	0.0000E+00	9.047900E+08	0.05000	1.484E-04	0	0.6088	age135_m72	0.0000	2.7284E+00	9.2713	-1.0000	82.4165
41	0.0000	0.0000E+00	0.0000E+00	1.434000E+09	0.05000	7.827E-05	0	0.5847	age139_m72	0.0000	5.4087E+00	11.7629	-1.0000	65.6914
42	0.0000	0.0000E+00	0.0000E+00	2.500000E+09	0.05000	3.759E-05	0	0.5583	age150_m72	0.0000	9.1184E+00	15.9948	-1.0000	54.8760
43	0.0000	0.0000E+00	0.0000E+00	5.000000E+09	0.05000	1.548E-05	0	0.5320	age161_m72	0.0000	1.5789E+01	22.6938	-1.0000	43.0039
44	0.0000	0.0000E+00	0.0000E+00	1.100000E+10	0.05000	6.532E-06	0	0.5070	age185_m72	0.0000	2.0586E+01	26.2549	-1.0000	35.3899
45	0.0000	0.0000E+00	0.0000E+00	1.300000E+10	0.05000	5.316E-06	0	0.5018	age193_m72	0.0000	2.1968E+01	27.1923	-1.0000	33.4145

These $N_\star$ lines give in column 2 the "light-fraction" population vector x_j. Columns 3 and 4 transform $\vec{x}$ to initial and current mass fractions. All are given in percentages. The next 7 columns (up to "a/Fe") just repeat info in arq_base and the base spectra. (L/M)_j in column 7 gives you what we call $l_{\lambda_0,j}$ in eq. (6), ie., the value of your j^{th} base spectrum at the normalization wavelength λ_0 (= l_norm in the "Normalization info" block). There are numerous ways of using the light-weighted and mass-weighted population vectors. The reader is referred to our papers for inspiration on such *post-processing* steps, as well as warnings about potential caveats.

The last 4 columns

```
... SSP_chi2r  SSP_adev(%)  SSP_AV  SSP_x(%)
```

are new to version 4 and give you chi2/Nl_eff, adev, AV_min and the normalization factor (sum-of-x) for fits using **exclusively** the component number j in the base. The only 2 parameters in these fits are A_V (SSP_AV) and the normalization factor (SSP_x). This feature may be useful for those looking for the best **single population fit**, like people using STARLIGHT for star-clusters or ellipticals galaxies. These fits (which are 50% analytical and 50% numerical), are done after STARLIGHT fits a mixture of populations. Thus, if these single-pop fits are what you want more and your are in a hurry, fiddle with the config parameters to force STARLIGHT to run fast for the composite-pops fit, such that it gets to this final stage quicker. Notice that kinematical parameters are **not** fitted in this stage, so either fix them or make sure the values obtained in the composite-pops fits are suitable.

• The following 3 blocks **should be ignored altogether!** (They are relics from debugging days, as many other things in arq_out and the code itself...)

• Finally, you'll find the last block: **"Synthetic spectrum"**. This is where the spectral fit is stored.

```
## Synthetic spectrum (Best Model) ##l_obs f_obs f_syn wei
 5501    [Nl_obs]
3400.00    0.00000   0.82017   -2.000
3401.00    0.00000   0.81350   -2.000
3402.00    0.00000   0.80737   -2.000
...
3709.00    0.00000   0.71285   -2.000
3710.00    0.00000   0.69655   -2.000
3711.00    1.03425   0.68478   -2.000
3712.00    1.07127   0.68849   -2.000
3713.00    1.07245   0.70993   -2.000
3714.00    1.07361   0.74032   -2.000
3715.00    0.85735   0.76776    7.879
3716.00    0.69766   0.78332    7.889
3717.00    0.80657   0.78283    7.697
3718.00    0.63605   0.76177    0.000
3719.00    0.54425   0.72062    0.000
...
8897.00    0.32947   0.31252   21.539
8898.00    0.31375   0.31263   21.821
8899.00    0.28400   0.31314   21.953
8900.00    0.25205   0.31389   22.061
```

The **first thing** you'll want to do after completing a spectral fit is to have a look at how good/bad it is! The info for your model versus data plot is given here in 4 columns: λ, O_λ and M_λ and w_λ. Both O_λ and M_λ are given in units of fobs_norm. The weight-spectrum is equal to $1/e_\lambda$ (with the error also in units of fobs_norm), except for pixels which were masked (signalled with $w_\lambda = 0$), flagged ($w_\lambda = -2$) or clipped ($w_\lambda = -1$) away. In the example above, the first useful pixel in at $\lambda = 3715$ Å. Just 3 pixels after it, the [OII]λ3727 emission line mask starts (as defined in **Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN**), so $w_\lambda = 0$. The plots at the end of this doc (§9) show examples of how to visualize this and other fits.

6.1 Example STARLIGHT output files

This distribution includes 8 STARLIGHT fits to each of the 3 input spectra listed in §4.1.4 (24 fits in total). The 8 fits correspond to 2 different arq_config's, 2 extinction laws and 2 bases. They are named appending suffixes to arq_obs. For example: **0414.51901.393.cxt.sc4.C11.im.CAL.BN** is the fit of **0414.51901.393.cxt** with StarlightChains_v04 (hence 'sc4'), configuration **StCv04.C11.config** (hence 'C11'), individual mask (from **Mask.0414.51901.393.cxt.sc1.CRAP.gm.BN**, hence 'im'), CALzetti law (hence 'CAL') and base **Base.BC03.N** (hence the 'BN' suffix).

OBS: You may have overwritten them by now running the 3 grid_example*.in files, but that's Ok! In case you didn't, do **not** expect your own fits to these same spectra to yield results identical to those reported in these files! Because STARLIGHT uses random Markov-Chains, if, due to, say, machine-precision, one step of one of the chains changes then the whole thing may change! The essential characteristics of the fits, however, should not change.

6.2 Which fit should I use?

The multiple fits of each galaxy spectrum in this distribution differ in the base, extinction law and arq_config. Many more versions could be produced varying astrophysical ingredients and/or technical parameters. This brings up a nagging question: *Which fit should I adopt as the "official" one?*

If this question haunts your dreams/nightmares, welcome to the wonderful and hopelessly degenerate world of population synthesis! There is no simple answer to this question. Eventually, some of the fits will really be obviously worse in terms of figures of merit like χ^2 or adev. Most often, you'll find $\sim$ equally good fits changing important things like the base or the extinction law.

You just have to learn to live with this. *Priors* are often the only way to discard some fits. For instance, you may think the nuclear spectrum of an early type galaxy should have stellar metallicities $\geq Z_{\odot}$, and thus restrict your fits to metal rich bases. Similarly, very young components could be discarded if you are applying STARLIGHT to an elliptical galaxy. Last but not least, learn not to over-interpret the fits! When working with large bases, you should somehow compress the output population vector to something reasonable. This is not the place to discuss this issue, so have a look at Cid Fernandes et al. (2004b and 2005) and Cid Fernandes (2007a and 2007b), where compression strategies are reviewed. (Just yesterday there appeared a nice paper by Tojeiro et al. on arXiv:0704.0941 discussing this issue.)

The plots at the end of this document show that the general look of the age distribution is essentially the same for all fits of a same galaxy, which should at least alleviate the which-fit-to-use crisis.

Figure 2: Welcome to the middle of this document!

7 What does STARLIGHT do and how?

Now that you know how to run STARLIGHT, you may be wondering exactly what you have done... Here's a not-so-brief description (which, despite its length, does not tell you all details!). As you may already have realized, the code is far more complex than it needed be (and many times less elegant than desirable!), and offers many more options than an average user will ever want to play with. This happened because the code grew up gradually from applications to very different data sets, with different ingredients and studied for different purposes. Besides this diversity in what goes in and comes out of the code, we have experimented with many methods to explore the parameter space, and settled for an efficient but frankensteinish combination of numerical techniques with an inevitably large number of technical parameters. The code input, output and structure reflect this history and that of the numerous tests carried out on the way to the present version. Indeed, several of its features are obsolete skeletons from previous versions, kept only for backwards compatibility. As a result, STARLIGHT is a rather messy code, but prepared to handle a variety of applications, so its complexity payed off.

In this second half of this manual we tell you some of STARLIGHT secrets. You need to have at least a rough understanding of these details to have an idea of what the many entries in arq_config mean and be able to play with them to best suit your needs. You also need to read to fully understand the physical results stored in the output file. Take another deep breath and another cup of whatever it is you are drinking...

7.1 The model spectrum $M_\lambda(\vec{x}, A_V, A_V^Y, v_\star, \sigma_\star)$

STARLIGHT deals with $N_\star + 4$ parameters: The **population vector** $x_1 \ldots x_{N_\star}$, the global extinction A_V, the "selective" extinction A_V^Y, plus two kinematical parameters: a velocity shift, $v_\star$, and velocity dispersion $\sigma_\star$. These parameters enter in the equation for a model spectrum M_λ, deduced below.

Start decomposing M_λ onto $N_\star$ components:

$$M_\lambda \;=\; \sum_{j=1}^{N_\star} L_{\lambda,j} = \sum_{j=1}^{N_\star} L^0_{\lambda,j} \otimes G(v_\star, \sigma_\star)\; 10^{-0.4 A_{\lambda,j}} \tag{4}$$

$$= \sum_{j=1}^{N_\star} L^0_{\lambda_0,j} b_{\lambda,j} \otimes G(v_\star, \sigma_\star)\ 10^{-0.4 A_{\lambda,j}} \tag{5}$$

where G is a Gaussian filter[6] centered at velocity $v_\star$ and with dispersion $\sigma_\star$ and $A_{\lambda,j}$ is the extinction at λ of population j. In these equations $L^0_{\lambda,j}$ is the spectrum of population j *without* extinction nor kinematics, and

$$b_{\lambda,j} \equiv \left(\frac{L^0_{\lambda,j}}{L^0_{\lambda_0,j}} \right) = \left(\frac{B_{\lambda,j}}{B_{\lambda_0,j}} \right) \tag{6}$$

where $B_{\lambda,j}$ is the spectrum of base component j (as read from arq_j specified in arq_base). The value of this spectrum at the normalization wavelength, $B_{\lambda_0,j}$, is listed in the column (L/M)_j in STARLIGHT's output files, in its original units (eg, $L_\odot \text{Å}^{-1} M_\odot^{-1}$ if the base comes from properly calibrated evolutionary synthesis models).

Let's further define the j^{th} normalized base spectrum convolved with the kinematical filter

$$\gamma_{\lambda,j} \equiv b_{\lambda,j} \otimes G(v_\star, \sigma_\star) \tag{7}$$

and write

$$A_\lambda = A_V q_\lambda \tag{8}$$

Then

$$M_\lambda = \sum_{j=1}^{N_\star} \left[L^0_{\lambda_0,j} 10^{-0.4 A_{V,j} q_{\lambda_0}} \right] \gamma_{j,\lambda} 10^{-0.4 A_{V,j} (q_\lambda - q_{\lambda_0})} \tag{9}$$

The term in square brackets is the synthetic flux at λ_0 due to component j, extincted by $A_{V,j}$, and *prior* to convolution with G. This is what we call x_j:

$$x_j \equiv L^0_{\lambda_0,j} 10^{-0.4 A_{V,j} q_{\lambda_0}} \tag{10}$$

so our final equation for the model spectrum is

$$M_\lambda = \sum_{j=1}^{N_\star} x_j \gamma_{j,\lambda} 10^{-0.4 A_{V,j} (q_\lambda - q_{\lambda_0})} \tag{11}$$

[6]This "convolution" transforms a spectrum $F(\lambda)$ to $I(\lambda)$, given by

$$I(\lambda) = \int_{-\infty}^{+\infty} F\left(\lambda' = \frac{\lambda}{1 + v/c} \right) G(v; v_\star, \sigma_\star) dv$$

Repeating, written in this way, x_j is the monochromatic flux at λ_0 of component j, extincted by its respective $A_{V,j}$ and prior to application of the kinematical kernel.[7] Since all spectra will be normalized at or around λ_0 (see below), in practice, x_j is $\sim$ the fraction of light due to component j at λ_0. It is not, however, exactly a light fraction! Indeed, in this formalism, the x_j's will in general not add up to 100%. Unlike in some earlier population synthesis work (eg, Bica 1988; Cid Fernandes et al. 2001), STARLIGHT does not impose that model and data match exactly at the normalization wavelength.

As written above, each component can have a different A_V, but in practice we set $A_{V,j}$ according to the "YAV_flag" read from the base master-file arq_base:

$$A_{V,j} = A_V \qquad \text{if} \quad \text{YAV\_flag(j)} = 0 \tag{12}$$

$$A_{V,j} = A_V + A_V^Y \qquad \text{if} \quad \text{YAV\_flag(j)} = 1 \tag{13}$$

Hence, instead of $N_\star$ extinctions, we deal with just two (one if YAV_flag = 0 for all j's), one global (A_V) and another "selective" (A_V^Y). As said before, using YAV_flag = 1 is **not** recommended unless you know exactly what you are doing and are prepared to deal with all problems related to more-than-one extinction fits.[8] A version of STARLIGHT allowing for multiple extinctions is under development.

7.1.1 Normalization

As said above, the base spectra are all normalized at λ_0. The choice of λ_0 is made in arq_config, where we call it **l_norm**.

In principle we could do the same for the observed spectrum (O_λ), but in real life the λ_0 pixel in O_λ may be affected by garbage (a noise spike, a cosmic ray, etc.). This is why STARLIGHT normalizes O_λ by O_N, defined as the **median** flux between **llow_norm** and **lupp_norm**, also specified in arq_config. Choose a relatively wide, problem- and line-free window, such that fobs_norm does not deviate much from the continuum in the range.[9] We usually set **l_norm** = 4020, **llow_norm** = 4010 and **lupp_norm** = 4060 Å. You may choose other values, but make sure your choice satisfies

$$\textbf{llow\_norm} < \textbf{l\_norm} < \textbf{lupp\_norm} \tag{14}$$

for simplicity.

[7] Comparing (11) with eq. (1) of Cid Fernandes et al. (2005), you will realize that in eq. (11) the M_{λ_0} term present in that paper has been **absorbed** in the definition of x_j (this is how it is defined inside STARLIGHT). A more subtle difference is that in eq. (1) of Cid Fernandes et al. (2005) you may get the impression that we apply the convolution with the kernel G *after* reddening the spectrum, whereas in practice it is the other way around. This is really irrelevant, since the extinction curve is smooth over scales of $\sigma_\star$.

[8] Another way to emulate different extinctions is to forget YAV_flag altogether (setting it to 0) but **include reddened spectra** in your base! This has been tried and we got sensible qualitative results, though hard to manipulate quantitatively and not-easy to interpret...

[9] *All* O_λ fluxes in the **llow_norm** → **lupp_norm** range, including masked and flagged pixels, are used in the computation of the median.

The value of O_N is denoted by **fobs_norm** in arq_out. Its units are the same as the input O_λ, whatever they are. You'll need this value to denormalize the observed & synthetic spectrum at the end of arq_out.

With this normalization x_j will be the flux (or luminosity) of the j^{th} component in units of O_N. As said above, in general, it will **not** add up to 100%, but you may trivially renormalize x_j by $\sum x_j$ if you would rather work with light fractions. While STARLIGHT does not force $\sum x_j = 1$, it limits this normalization factor to within **fn_low** and **fn_upp**, given in arq_config.

7.2 The Numerical scheme

This long section describes numerical aspects, focusing on issues related to technical parameters in arq_config, so that you can have at least a rough idea of what all those numbers mean. Less curious users may want to skip this section altogether and go to §8 where we suggest how to set-up arq_config for different types of fits.

The fit is carried out with a mixture of simulated annealing plus Metropolis plus Markov Chain Monte Carlo techniques which explores the parameter space and searches for the minimum

$$\chi^2 = \chi^2(\vec{x}, A_V, A_V^Y, v_\star, \sigma_\star) = \sum_\lambda \left[(O_\lambda - M_\lambda)\, w_\lambda\right]^2 \tag{15}$$

where the weight $w_\lambda = e_\lambda^{-1}$ except for masked/flagged points (where it may be 0 or something else, depending on what you did in the masks file).

The minimization consists of **4 stages**, whose general aims are:

1. **First Fits (FF):** Make a broad sweep of the parameter space;
2. **Clip & Refit:** Exclude "unfitable" pixels;
3. **Burn-In:** Detailed fit with the full base;
4. **EX0s fit (EX0):** Discard irrelevant components and fine tune the fit.

In each of these stages, **N_chains** sets of parameters are evolved through the parameter space in a pseudo-random fashion, accepting or rejecting moves according to the Metropolis algorithm (ie, based on the likelihood ratio of the before and after positions). Every now and then the chains talk to each other to try to meet in a common region of parameter space.

In what follows we describe each stage. The FF stage is explained in greater detail, as all others follow a similar logic. In fact, in all 4 stages STARLIGHT calls the same set of routines.

7.2.1 First Fits: Broad exploration of the parameter space

• **The cooling schedule:** The FF stage of the algorithm goes through **N_loops_FF** loops where we decrease the "Temperature" from an initially hot to a final cold value according to a cooling schedule:

$$T_k = \textbf{Temp\_ini\_FF} \times \left(\frac{\textbf{Temp\_fin\_FF}}{\textbf{Temp\_ini\_FF}}\right)^{(k-1)/(\textbf{N}_{\textbf{loops}}-1)} \tag{16}$$

where $k = 1 \ldots$ **N_loops_FF**. For example, with **N_loops_FF** $= 3$, **Temp_ini_FF** $= 100$ and **Temp_fin_FF** $= 1$ we would have $T = 100, 10$ and 1. "Temperature" here means the square of the factor by which we multiply the errors e_λ in the data. Recall that, under the assumption of Gaussian errors, the likelihood function looks like a Boltzmann factor,

$$\mathcal{L} \propto e^{-\chi^2/2} \equiv e^{-E}$$

with $\chi^2/2$ playing the role of an adimentional energy. The temperature at loop k modifies this to

$$\mathcal{L}_k \propto e^{-E/T_k}$$

so that with $T_k > 1$ we *broaden* the likelihood distribution, whereas with $T_k < 1$ we make it *narrower*. With very large temperatures, the likelihood hyper-surface is $\sim$ flat, such that any model will do it, and most moves in parameter space are acceptable, whereas at low temperature the landscape is more complex, with peaks and valleys, such that moves in some directions will produce unacceptably large (in the Metropolis-criterion sense) upward changes in χ^2.

For the First Fits, you should start from high T, say, $T =$ **Temp_ini_FF** ~ 100 or more, and end up with $T =$ **Temp_fin_FF** ~ 1, which is equivalent to starting from errors overestimated by a factor of $\sqrt{100} = 10$ and end up with errors as they are ($T = 1$). High T allows large moves in the parameter space, which is good to take chains from their random initial positions to more likely regions in not too many steps. Cooling the system gradually then focus on these regions and produces a more detailed mapping of the likelihood hyper-surface.

With **i_RestartChains_FF** $= 1$ the parameters of each of the chains are reset to their respective best values so far before the next loop (or stage), while with **i_RestartChains_FF** $= 0$ the chains parameters are left wherever they are in parameter space and re-start from there in the next loop (or stage). Yet another option is to reset only the highest energy chain (worst χ^2) to the best global model so far, which you do with **i_RestartChains_FF** $= 2$.

• **Adaptive step sizes:** At high T, most small movements in parameter space are accepted and the system tends to a pure random-walk, whereas at low T large movements have a very large probability of being rejected, so the chains get stuck. Clearly, fixed step sizes would render the whole scheme miserably inefficient. In its first version (Cid Fernandes et al. 2004b), STARLIGHT synchronized the cooling schedule with a step-size reduction schedule calibrated with simulations. The current version does something much better: It adjusts step sizes dynamically, aiming to reach a certain efficiency = **eff_IDEAL_FF** = number of accepted moves / total number of steps.

To evaluate efficiencies, STARLIGHT must evolve the chains a certain number of iterations (N_{iter}). This number is calibrated in terms of the number of parameters ($N_{par} = N_\star + 2$)[10]: $N_{iter} =$ **fN_sim_FF** $\times N_{par}$, which corresponds, statistically speaking, to **fN_sim_FF** moves per parameter.

[10] $N_{par} = N_\star + 2$, ie, $\vec{x}$ plus the 2 extinctions, since $v_\star$ and $\sigma_\star$ are dealt with separately.

Every N_{iter} iterations the code pauses to have a look at how the efficiency (ϵ) of each chain is doing. If ϵ is smaller than desired, step sizes are reduced by a factor

$$\alpha = \textbf{Falpha}^{(\epsilon - \textbf{eff\_IDEAL\_FF})/\textbf{eff\_IDEAL\_FF}} \tag{17}$$

whereas if $\epsilon >$ **eff_IDEAL** then $\alpha > 1$ and steps get larger.

The **fini_eps_FF** and **ffin_eps_FF** entries in arq_config are used to define initial step sizes, computed dividing the allowed parameter range by **fini_eps_FF** in the first loop of the cooling schedule and by **ffin_eps_FF** in the last one. As you have just seen, step-sizes are adjusted dynamically inside the code, so these two technical parameters are essentially irrelevant.

With **i_UpdateAlpha** $= 0$ you turn off adaptive step sizes, a very bad idea! With **i_UpdateAlpha** $= 1$ all chains will have the same α, determined from the mean efficiency among all **N_chains** chains, while **i_UpdateAlpha** $= 2$ sets up one α for each chain.

• **Convergence criteria:** Now comes the question of how to decide when to stop iterating. Every $N_{par}\times$ **fN_sim_FF** iterations (per chain) we apply a test similar to that proposed by Gelman & Rubin (1992), which compares the typical variance of a parameter within each of the chains to the variance of this same parameter between all chains. Doran & Müller (2003) describe this test (as well as an adaptive step size scheme similar to the one implemented in STARLIGHT). Essentially, it all boils down to monitoring an adimensional number R, which is large when chains are far apart in the parameter space and tends to one as their separations converge to distances comparable to their internal variances (though $R = 1$ is hard to achieve in practice). The **R_ini_FF** entry in arq_config defines when acceptable convergence has been achieved in the first FF loop. This value may be decreased for subsequent (cooler) loops in the annealing sequence, by setting **R_fin_FF** $<$ **R_ini_FF**. Values of 1.1, 1.2 or 1.3 are Ok. The smaller R, the more stringent the criterion and the longer it takes to satisfy it.

Another decision to be made is if you want all parameters to satisfy the Gelman & Rubin test individually or if you are happy with an on-average convergence, ie, whether the criterion is applied to every R_j or just to $\overline{R}$. Setting **IsGRTestHard_FF** $= 1$ you require a strict one-by-one parameter convergence, whereas **IsGRTestHard_FF** $= 0$ only requires convergence on average. The latter option is obviously faster. **IsGRTestHard_FF** $= 0$ is Ok for the First Fits.

Sometimes chains will take forever to converge, specially if you are very rigorous, doing things like **IsGRTestHard_FF** $= 1$ and **R_fin_FF** $= 1.0$. (They will literally take forever if you chose **R_fin_FF** $\ll 1$!) To avoid convergence to take too long, the total number of steps per chain in each cooling loop cannot exceed $N_{par}\times$ **fNmax_steps_FF**. Playing with these 3 numbers you can force the number of steps ($\propto$ computing time) to be controlled by the maximum number of steps or convergence. Setting **R_fin_FF** $= 0.0$, for instance, you will always be limited by your choice of **fNmax_steps_FF**. Conversely, use a very large value for **fNmax_steps_FF** and a reasonable **R_fin_FF** (say, 1.2) if you want to insure convergence.

• **Kinematical Parameters:** All the above applies to the x_j's and A_V (and A_V^Y too if you are using it). The kinematical parameters $v_\star$ and $\sigma_\star$ are **not** changed during the Metropolis runs, since

the convolution with the LOSVD kernel at each step would slow things down tremendously. $v_\star$ and $\sigma_\star$ are only refitted after each of the annealing loops. The code thus goes through a sequence of

1. fix $v_\star$ and $\sigma_\star$;
2. find best $\vec{x}$ and A_V;
3. fix $\vec{x}$ and A_V;
4. find better $v_\star$ and $\sigma_\star$; [11]
5. cool T and goto (1)

Hence, if you are more interested in the kinematics $(v_\star, \sigma_\star)$ than in the populations $(\vec{x}, A_V)$, use many loops, say **N_loops_FF** = 10.

7.2.2 Clip & Refit

In most circumstances (in fact in all I have seen so far), the best parameters found during the First Fits stage already produce a very good spectral fit. Hence, if a pixel could not be fitted in the FF stage it will very probably not be well fitted at all regardless of how many more models you try, so we might as well give up fitting it!

This is the role of this second stage. We check which points deviate by "too much" (with respect to the best M_λ found in the FF stage) and clip them, setting $w_\lambda = 0$. There are **4 possible clipping methods: clip_method_option** = NSIGMA, RELRES, ABSRES and NOCLIP, the latter of which needs not be explained.

• In the "ABSRES" method we clip points with $|O_\lambda - M_\lambda| >$ **sig_clip_threshold**$\times$ the rms of $(O_\lambda - M_\lambda)$ over all (non-masked) pixels. This is **not** a good choice if your noise changes appreciably with λ, but it is a reasonable one when e_λ is not available.

• "RELRES" clipping consists of excluding points whose $|O_\lambda - M_\lambda|/e_\lambda$ "relative residual" deviates by more than **sig_clip_threshold**$\times$ the rms of this same quantity over all non masked points. If e_λ is constant this is equivalent to ABSRES.

• Finally, "NSIGMA" clipping consists of excluding points whose $|O_\lambda - M_\lambda|$ residual deviates by more than **sig_clip_threshold** $\times$ the **local** error in O_λ (better known as e_λ). This scheme should clip less points than ABSRES, since it may allow points with large errors to be kept in the fit, but with their correspondingly low $w_\lambda = 1/e_\lambda$ weights. This is our favorite method when e_λ is reliable.

In all cases, clipped points are signaled with -1 in the w_λ column at the bottom of arq_out. **You should always check whether STARLIGHT clipped too many points.** You can do this looking at Ntot_cliped in the output file or, even better, plotting the $w_\lambda = -1$ pixels with a different color in your O_λ versus M_λ comparison. The red-crosses in the residual spectrum panels in Figs 3–5 mark clipped points, all obtained with **clip_method_option** = NSIGMA and **sig_clip_threshold** = 3 (as set in the respective arq_config's). If too many points have been

[11] Before this step STARLIGHT actually does a quick brute-force fine-tuning of A_V (and A_V^Y).

clipped, increase **sig_clip_threshold** or go for the NOCLIP method. In fact, if you believe your masks and/or flags already clean all that is wrong or unfitable in your spectrum, NOCLIP should be your primary choice.

If $N \geq 1$ pixels were deemed to be clipped, STARLIGHT refits the spectrum with the same technical parameters (Temperature, R-threshold, etc) used in the last FF loop before proceeding.

OBS: If you want to compare the χ^2's of two fits of a same spectrum (say, with different bases), make sure they both clipped the same points, ie, that both runs fitted the exact same data, or else take into account these differences. (It is not fair to compare a low χ^2 fit which clipped many points with one which is apparently worse but clipped fewer points.)

• **Weight control filter:** Sometimes an error spectrum contain pixels whose e_λ is much smaller than all other pixels, and will thus be given much larger weights ($w_\lambda = 1/e_\lambda$) in the fit. If you are happy with this (i.e., you trust all your e_λ's), set **wei_nsig_threshold** $= 0$ in arq_config and read no further.

It is possible that such large w_λ's result from artifacts in the reduction process, in which case we should change w_λ to more reasonable values. To correct these suspicious errors we first compute the mean and rms of w_λ over all non flagged nor masked pixels. Then we identify all pixels with $w_\lambda > w_{limit} = \overline{w_\lambda}$ + **wei_nsig_threshold** $\times \sigma(w_\lambda)$ and reset them to $w_\lambda = w_{limit}$. Setting **wei_nsig_threshold** $= 2$, for instance, would bring w_λ's larger than 2 sigma over the mean to $w_\lambda = 2$ sigma over the mean, still large, but not so much.

The n_censored_weights entry in the etc block of arq_out tells you how many points had their weights changed by this filter. The value of w_{limit} is also listed there (wei_limit).

OBS: This feature should be **turned off** (with **wei_nsig_threshold** $= 0$) if you are playing with weights other than 0 in your masks file, otherwise STARLIGHT will undo your extra-weighting!

7.2.3 Burn-In

By this stage all Markov Chains should be relatively close to each other in parameter space. We could thus start a final iteration with $T = 1$ to sample the full probability distribution function (PDF) of the parameters. This would work like this: Run the chains until they satisfy $R <$ **R_Burn_in** (with **IsGRTestHard_BurnIn** $= 0$ or 1), throw away the results, freeze-in the adaptive-size scheme, and iterate another many steps to sample the PDF. STARLIGHT, however, stops when $R <$ **R_Burn_in** is achieved, since it does **not** attempt to map the PDF nor estimate errors in its parameters (admittedly a **major drawback**, but an inevitable one given that we try to be as general as possible in terms of astrophysical applications of the code). Instead, STARLIGHT seeks a single best solution, so when the desired convergence is achieved, it goes to the last stage. The Burn-In stage is the last chance to try a detailed fit with **all** $N_\star$ components.

The Burn-In phase fit is carried out at a Temperature = **Temp_fin_FF**, which should be = 1 unless you (think you) know what you are doing. (The only reason to fiddle with **Temp_fin_FF** is if your errors are very wrong, in which case it is much better to fix them instead!)

7.2.4 EX0s: Fits with a condensed base

In the EX0 stage, the whole fit is fine tuned repeating an annealing loop similar to that in the FF stage, but with **NEX0s_loops** loops, lowering the temperature from **Temp_ini_EX0** to **Temp_fin_EX0** and etc. All technical parameters present in the FF stage have a counterpart in this stage: **Temp_ini_EX0**, **Temp_fin_EX0**, **fini_eps_EX0s**, **ffin_eps_EX0s**, **R_ini_EX0s,** **R_fin_EX0s**, **IsGRTestHard_EX0s**, **N_loops_EX0s**, **fN_sim_EX0s**, **fNmax_steps_EX0s**, and **eff_IDEAL_EX0s**. If you did not understand them in the FF section, it is because we were not clear enough, so there is no point in trying again here.:(

• **Reduction of the base:** The real difference in this last stage is that it offers you the possibility of throwing away all "irrelevant" components and going for a more refined fitting. When your work with large bases (say, $N_\star = 100$ or more), many of the components are just completely irrelevant, with very little or no flux at all. The smaller $N_\star$, the more efficient and the faster the STARLIGHT numerical scheme is (CPU time scales roughly with $N_\star^2$). Ok, but how are you going to decide which components are relevant?

First, you need a fiducial population vector $\vec{x}$ upon which to base your decision. Choosing **EXOs_PopVector_option** = MIN in arq_config you will pick the best $\vec{x}$ found so far. Alternatively, you may chose **EXOs_PopVector_option** = AVE, in which case the current mean $\vec{x}$ over all chains will be used.

Now let's decide what "irrelevant" means in practice. With **EXOs_method_option** = SMALL you will throw away all components with $x_j <$ **EXOs_Threshold**. For instance, with **EXOs_Threshold** = 0.02 components with $x_j < 2\%$ will be discarded.

EXOs_method_option = CUMUL provides a better way of defining irrelevance. It sorts the x_j and excludes the smaller ones whose light-fractions add up to $\leq$ **EXOs_Threshold**. With **EXOs_Threshold** = 0.02, for instance, this would ensure that we lose only $< 2\%$ of the flux at λ_0 in this base-reduction process.

Why is this better than **EXOs_method_option** = SMALL? Suppose you have a base with $N_\star = 100$ where component $x_1 = 80\%$ and the remaining 20% of the light is $\sim$ equally spread over the remaining 99 components, with $x_j = 0.20/99 \sim 0.002 = 0.2\%$ for $j = 2 \ldots 100$. Using the SMALL method and **EXOs_Threshold** = 0.02 = 2% would then throw away 99 weak components, but leave a big 20% deficit in the flux at λ_0, which you would need to compensate forcing the surviving component x_1 to be increased to $\sim 100\%$. You most definitely do not want to do this. In fact, STARLIGHT will not accept reduced bases with less than $\max(3, \textbf{fEX0\_MinBaseSize} \times N_\star)$ components. In any case, STARLIGHT will realize if the new fits with a condensed base are worse than the previous ones, so the final best model will be the one found prior to this ill-fated EX0s stage.

In both cases arq_out reports the size of the condensed base as NEX0s_base, and populations out of this list are given $x_j = 0$. In our fits of 573141 SDSS galaxies with $N_\star = 150$ (**Base.bc03.S**), available at www.starlight.ufsc.br, NEX0s_base is typically 30. This scheme thus provides a crude but effective compression method (by a factor of 5 in this case).

• **Fine tuning:** Since you now have a smaller base, you can afford much more detailed fits. For instance, you may set up a cooling schedule which takes you to $T \ll 1$, which is a way to narrow on your best solution. **StCv04.C11.config**, for instance, does this setting **Temp_ini_EX0s** = 1 and **Temp_fin_EX0s** = 0.001. Markov Chain Monte Carlo methods are good to sample a likelihood distribution, but do not care much about finding the best solution (the chains will keep wandering about within a region around the best model, doing no special effort to get there). This trick is a way to fool the algorithm, such that it will sample smaller and smaller regions, bracketing the best χ^2. Requiring a large **eff_IDEAL_EX0s** (say, 0.5) works in the same direction.

This is one way of using the EX0s stage. That's how we like to use it, specially when dealing with large bases. Cooling below $T = 1$ is also a way of guaranteeing you will fit your data even with overestimated errors. Indeed, the whole cooling schedule from the FF to the EX0 phases could be scaled according to your beliefs about how over/under-estimated your errors are. We normally do things such that the Burn-In phase (the mid-point of the code) runs with $T =$ **Temp_fin_FF** $= 1$, such that the likelihood distribution is sampled at its correct shape in this stage. You may want to play with the temperatures, but as long as you start from $T \gg 1$ and end up $T \ll 1$ things should work smoothly.

7.2.5 Other entries in arq_config

Some quick comments on other things in arq_config.

• **AV_low** and **AV_upp** are exactly what their names say: The allowed limits for A_V. Physically, **AV_low** = 0, but you may allow for **negative extinction**! There are reasons (related to inconsistencies in the base) to think this unphysical choice should be allowed for (e.g., Gallazzi et al. 2005; Mateus et al. 2006), even though it is unclear how to interpret the results then. (The example fits of 0784.52327.478.cxt all have $A_V < 0$).

• **YAV_low** and **YAV_upp** are the limits for A_V^Y. Use zero or close to it unless you know what you're doing.

• **fn_low** and **fn_upp** are the allowed range for the normalization factor, i.e., the synthetic flux at λ_0 ($\sum x_j$) in units of fobs_norm. Choose a range centered in 1. For SDSS data and our favorite choice of normalization ($\lambda_0 = 4020$ Å), **fn_low** = 0.7 and **fn_upp** = 1.3 suits all galaxies.

• **f_cut** is the minimum allowed flux (in units of fobs_norm). Pixels with fluxes smaller than **f_cut** ×fobs_norm will be masked. This is a safety feature mainly to help cases where flag$_\lambda$ is not available and some flux values are too low or even negative. These points would most likely be clipped away anyway. Using **f_cut** < 0 you effectively turn this off.

• **N_int_Gauss** is the number of points within ±6 sigma in the Gaussian LOSVD convolution integral. **N_int_Gauss** = 31 or 51 is fine. If you care very much about $\sigma_\star$, use more points.

• If you are running 1000's of fits, you may have several grid files running in parallel (say, different nodes of a cluster). With **i_SkipExistingOutFiles** = 1 you make sure that if arq_out already exists then STARLIGHT will skip it and go for the next fit. This allows you to use the *same grid file* for several parallel runs. Eventually, if a run is aborted (say, due to a power cut), some of the files will be incomplete. Check if some of the arq_out's are abnormally small (ls -lS | more will

show this) and rerun them if needed. Setting **i_SkipExistingOutFiles** = 0 will redo the fit and overwrite arq_out.

• **i_FitPowerLaw** and **alpha_PowerLaw** is an obsolete feature to allow you to include a $F_\lambda \propto \lambda^{\textbf{alpha\_PowerLaw}}$ power-law in the base. If you want a power-law, say, for AGN fits, it is better to build one and include it in the base-file yourself. This feature will be removed soon.

• **i_FastBC03_FLAG** = 1 speeds up the reading and resampling of BC03/STELIB base-files. Not applicable to bases with different samplings.

• The example arq_config files offer half-a-line clues on what **xinit_max**, **i_UpdateAVYAVStepSeparately**, **i_HelpParWithMaxR,** **prob_jRmax**, **i_HelpPopVectorMove2Average,** **prob_HelpPopVectorMove2Average**, **i_HelpAVYAVMove2Average,** **prob_HelpAVYAVMove2Average** are. Most of these parameters control the communication between chains, which emulates a smart "proposal distribution function" (roughly in the directions of largest covariances in parameter space). Several different combinations were tried before settling for the values given. They are not very critical anyway.

• **NRC_AV_Default** has to do with the rapid-χ^2 tricks, explained below. Not critical either, specially because STARLIGHT will change it if it finds necessary.

7.3 Rapid-χ^2 tricks

An important difference (from the computational point of view) with respect to initial versions of STARLIGHT is that we now perform a series expansion of the

$$R_\lambda \equiv -0.4 \left[\frac{A_\lambda}{A_V} - \frac{A_{\lambda_0}}{A_V} \right] \tag{18}$$

extinction factor in eq. (11), which allows a much faster computation of χ^2! Here is a very-very brief description of this nice trick. Look up Jean Michel Gomes thesis in the Publications section in www.starlight.ufsc.br, where all the math is presented.

The idea is that when you have a spectrum with many points, it will take long to sum over all λ's to compute χ^2 (see eq. 15). The trick is to open up the squares in χ^2, which produces 3 terms:

$$\chi^2 = \sum_\lambda w_\lambda^2 O_\lambda^2 - 2 \sum_\lambda w_\lambda^2 O_\lambda M_\lambda + \sum_\lambda w_\lambda^2 M_\lambda^2 \tag{19}$$

where you already see that the first term needs be computed only once. Now do two things: plug in our equation for M_λ, which consists of a sum over $j = 1 \ldots N_\star$ components (eq. 11), and rewrite the extinction term as follows:

$$10^{R_\lambda A_{V,j}} \equiv e^{\rho_\lambda A_{V,j}} = \sum_{n=0}^{\infty} \frac{(\rho_\lambda A_{V,j})^n}{n!}$$

Finally, rearrange the terms and you'll see that the $\sum_\lambda$ terms can be grouped into terms which depend on n and j but do **not** depend on either $\vec{x}$ or $\vec{A_V}$! Thus, they can be computed **once** and used every time these parameters change (in the Metropolis routine). Of course, you'll have to truncate the exp-series at N_e instead of ∞, which requires some care. STARLIGHT starts with $N_e =$ **NRC_AV_Default** terms in the series, but every now and then it checks that the rapid and "slow" χ^2's are not too different, and adjusts N_e if needed.) The gain in speed should be of order $N_\lambda/N_\star N_e$. It can be even better (by a further factor of $N_e \sim 10$) if you split the changes in $\vec{x}$ from those in $\vec{A_V}$ and explore some of the symmetries. For $N_\lambda \sim 5000$, $N_\star \sim 150$, this trick speeds things up by a factor of *over 300!* Notice that, because of this trick, STARLIGHT is not twice as fast if you lower your spectral sampling by a factor of two.

8 The configuration file again

We've gone through most of the parameters in arq_config. Here is what StCv04.C11.config looks like.

```
# Configuration parameters for StarlightChains_v04.for - Cid@Lagoa - 10/Feb/2007 #
#
#
# Normalization lambdas
#
4020.0        [l_norm    (A)]                = for base spectra only
4010.0        [llow_norm (A)]                = for observed spectrum
4060.0        [lupp_norm (A)]                =  "   "        "
#
#
# Parameter Limits
#
  -1.0        [AV_low (mag)]                 = lower allowed AV
   4.0        [AV_upp (mag)]                 = upper allowed AV
  -0.0001     [YAV_low (mag)]                = lower allowed YAV
   0.0001     [YAV_upp (mag)]                = upper allowed YAV
   0.7        [fn_low]                       = lower allowed Norm. factor = sum x_j
   1.3        [fn_upp]                       = upper allowed Norm. factor = sum x_j
-500.0        [v0_low (km/s)]                = lower allowed v0
 500.0        [v0_upp (km/s)]                = upper allowed v0
   0.0        [vd_low (km/s)]                = lower allowed vd
 500.0        [vd_upp (km/s)]                = upper allowed vd
#
#
# Clipping options & Weight-Control-Filter
#
NSIGMA        [clip_method_option]           = NOCLIP/NSIGMA/RELRES/ABSRES = possible clipping methods
3.0           [sig_clip_threshold]           = clip points which deviate > than this # of sigmas
2.0           [wei_nsig_threshold]           = weight-control-filter. Use <= 0 to turn this off! (see manual)
#
#
# Miscellaneous
#
    50.0      [dl_cushion (A)]               = safety margin for kinematical filter!
     0.001    [f_cut (units of f_norm)]      = Mask/ignore very low fluxes: f_obs <= f_cut
    31        [N_int_Gauss]                  = # of points for integration of kinematical filter
     1        [i_verbose]                    = 0/1      = Quiet/Talkative
     0        [i_verbose_anneal]             = 0/1/2/3  = Quiet/.../Verborragic
     0        [Is1stLineHeader]              = 1/0 = Y/N
     1        [i_FastBC03_FLAG]              = 1 for Fast-rebin of BC03 spectra!
     0        [i_FitPowerLaw]                = 1/0 = Y/N - include a Power Law in base
    -0.5      [alpha_PowerLaw]               = PL index, only used if iFitPowerLaw = 1
     0        [i_SkipExistingOutFiles]       = 1/0 = Y/N - skip or overwrite fits with already existent arq_out
#
#
# Markov Chains technical parameters
#
     7        [N_chains]                     = # of Markov Chains
     0.50     [xinit_max]                    = max(x_j) for initial random chain pop-vecs
     0        [i_UpdateEps]                  = 1/0 = Y/N. Not well tested: use 0!
     2        [i_UpdateAlpha]                = 0/1/2. 1 & 2 update step-sizes dynamically. 0 turns this off.
     2.0      [Falpha]                       = step-updating-factor.
     1        [i_MoveOneParOnly]             = 1/0 = Y/N. Not tested/debugged! Use 1!
     1        [i_UpdateAVYAVStepSeparately]  = 1/0 = Y/N.
     1        [i_HelpParWithMaxR]            = 1/0 = Y/N. Help convergence of ParWithMaxR
```

```
     0.2      [prob_jRmax]                    = prob to pick ParWithMaxR
     1        [i_HelpPopVectorMove2Average]   = 1/0 = Y/N. Help x convergence
     0.4      [prob_HelpPopVectorMove2Average] = prob of inverting sign of x-move to go towards mean
     1        [i_HelpAVYAVMove2Average]       = 1/0 = Y/N. Help AV/YAV convergence
     0.4      [prob_HelpAVYAVMove2Average]    = prob of inverting sign of AV/YAV-move to go towards mean
    10        [NRC_AV_Default]                = initial # of terms in the RC AV-series
#
#
# First Fits (FF) technical parameters
#
1.0e2         [Temp_ini_FF]                   = initial Temperature
1.0           [Temp_fin_FF]                   = final   Temperature
1.0e1         [fini_eps_FF]                   = initial step-size (fraction of low->upp range)
1.0e2         [ffin_eps_FF]                   = final   step-size (fraction of low->upp range)
1.3           [R_ini_FF]                      = initial GR-R convergence threshold
1.3           [R_fin_FF]                      = final   GR-R convergence threshold
0             [IsGRTestHard_FF]               = 0/1 = Hard/Soft GR-R convergence criterion
3             [N_loops_FF]                    = # of annealing loops
1             [i_RestartChains_FF]            = 1/0 = Y/N = Reset chains to best pars after each loop
1.0e1         [fN_sim_FF]                     = x N_par = length of each chain loop...
1.0e4         [fNmax_steps_FF]                = max length of each chain loop...
0.23          [eff_IDEAL_FF]                  = ideal efficiency, used to optimize step-sizes
#
#
# GR R-threshold & Method for Burn-In loop
#
1.2           [R_Burn_in]                     = GR-R convergence threshold for burn-in
0             [IsGRTestHard_BurnIn]           = 0/1 = Hard/Soft GR-R convergence criterion
#
#
# EXOs technical parameters
#
MIN           [EXOs_PopVector_option]         = MIN/AVE = Use x_min or x_ave to define condensed base
CUMUL         [EXOs_method_option]            = CUMUL/SMALL = method to define irrelevant components
0.02          [EXOs_Threshold]                = Irrelevance threshold
1.0           [Temp_ini_EXOs]                 = Analogous to FF entries above!
1.0e-3        [Temp_fin_EXOs]                 =    "
1.0e2         [fini_eps_EXOs]                 =    "
1.0e3         [ffin_eps_EXOs]                 =    "
1.2           [R_ini_EXOs]                    =    "
1.0           [R_fin_EXOs]                    =    "
1             [IsGRTestHard_EXOs]             =    "
5             [N_loops_EXOs]                  =    "
1             [i_RestartChains_EXOs]          =    "
1.0e2         [fN_sim_EXOs]                   =    "
1.0e3         [fNmax_steps_EXOs]              =    "
0.50          [eff_IDEAL_EXOs]                =    "
1             [IsScaleNstepsInEXOsFits]       = 1/0 = Y/N = Scale numbers of steps by # of EXO components.
1             [IsNoKin4LargeBaseInEXOsFits]   = 1/0 = Y/N = Stop v0 & vd fits in EXOs for large bases
0.0           [frac_NoKin4LargeBaseInEXOsFits] = Will stop v0 & vd fits if EXO-bases > frac_... * N_base
0.1           [fEXO_MinBaseSize]              = EXO-fits will use at least fEXO_MinBaseSize * N_base components

Cid@Lagoa - 10/February/2007

OBS: This config is a reformatted (for v04) & renamed version of the
     StCv03.t02.config used in the config tests of Jan/2007.


Technical parameters you may want to play with to obtain FAST, MEDIUM
& SLOW fits:

---------------------------------
|  FAST  |    MEDIUM  |  LONG  |
---------------------------------
|  5     |    7       |  12    | [N_chains]
|  3     |    5       |  10    | [N_loops_FF & *_EXOs]
|  1.3   |    1.2     |  1.1   | [R_ini_FF & R_fin_FF & *_EXOs]
|  0     |   0 or 1   |  1     | [IsGRTestHard_FF & *_BurnIn & *_EXOs]
---------------------------------
```

The comments at the bottom of the file give you some guidance as to what you may want to play with to produce fast, medium or slow fits.

The fits of nearly 600k SDSS galaxies discussed in Asari et al. (2007) were carried out with a medium -speed configuration, but we have tested both faster and slower fits and results do not change in any significant way. We repeat that given the pseudo-random-search nature of the fitting scheme, it may well happen that a fast fit produce a worse figure of merit (like χ^2 or adev) than a slow one.

Fits with different config parameters surely differ in the details. You can verify this comparing the *C11* with the *C99* example fits. Notice, however, that for the same reasons explained above

two fits of a same galaxy with identical configs will still differ if they use different seed for the random number generator!! (You may try this yourself repeating one galaxy in the grid file). Such differences can be quite substantial in individual components of $\vec{x}$, but the overall picture should not change appreciably. Again, read our papers to learn how to compress STARLIGHT results to achieve a more robust description of the fits.

9 Some examples

This distribution includes multiple STARLIGHT fits to each of the 3 input spectra listed in §4.1.4. Figs 3–5 show the results.

These examples illustrate some of the things one must be aware of when fitting galaxy spectra. First, the configuration does not make much difference, provided you sweep the parameter space minimally well. Second, as if needed be said, the base is important. Fits of **0414.51901.393.cxt** with bases N and S, for instance, differ. While the overall population vector is similar in both cases, with base S a small amount of light is allowed to go into old populations of low $Z_\star$ which are not present in base N. This has a minor effect upon $\vec{x}$ but a large one in the mass-fractions vector μ, given the much larger M/L of old populations. Hence, a small amount of light in old pops may translate into most of the mass. The non-linear transformation from what you see to what you get (light to mass) is of course one of those nasty aspects of stellar population synthesis you just have to get used to. There are also implications for the resulting A_V values (at the 0.1 mag level in this case).

In the case of **0784.52327.478.cxt**, one sees that including more base elements (ie, going from base N to base S) has the effect of spreading the light concentrated in a single pop to adjacent bins, which are spectrally similar.

Life may not be so degenerate if you reduce your parameter space (= use a small base), but if you do go for large $N_\star$, these are some of the issues you'll have to face.

Notwithstanding these and other caveats, STARLIGHT has been a very useful tool for our group, so we hope you'll find it useful too.

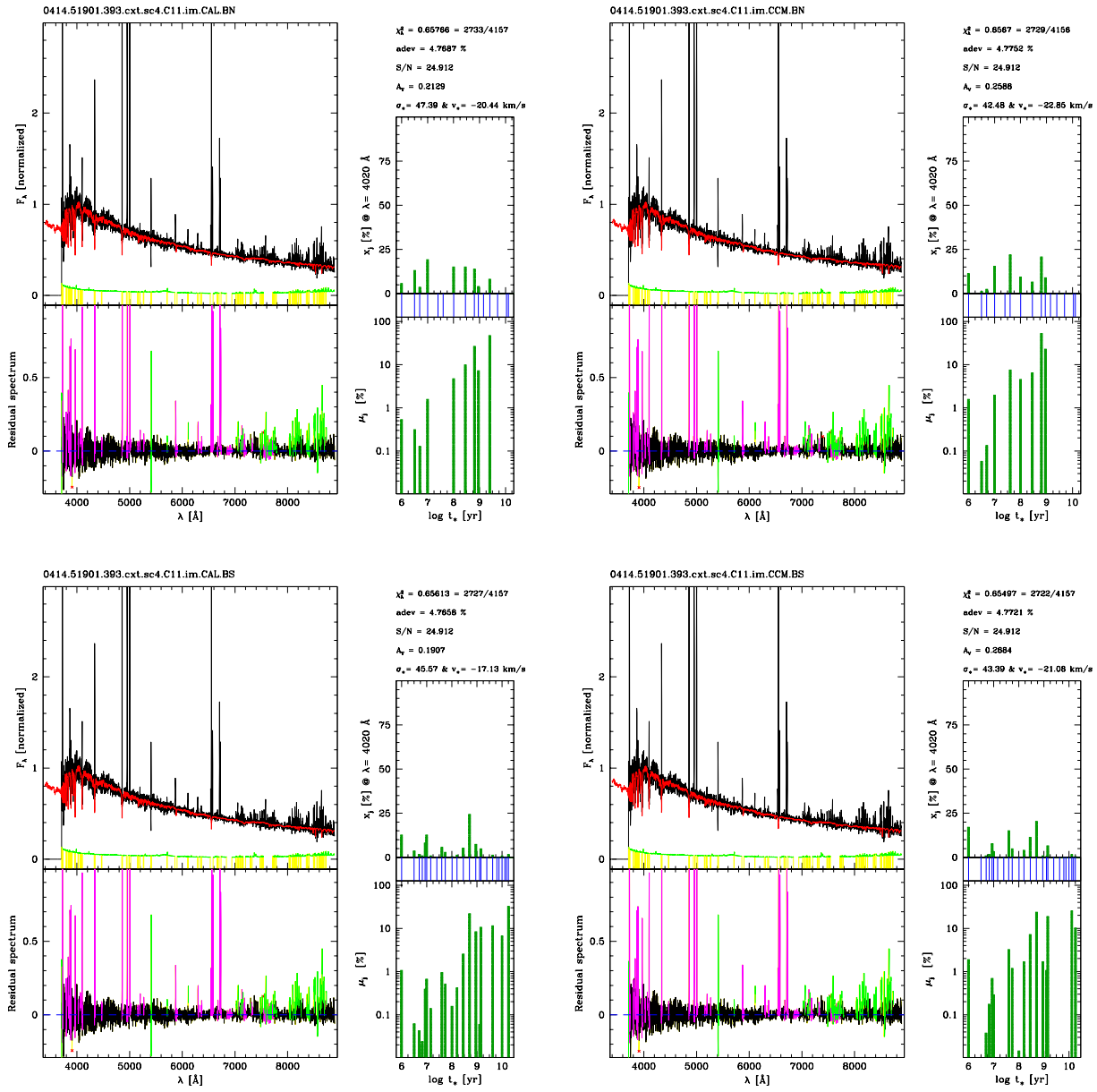

Figure 3: 4 fits of a star-forming SDSS galaxy **0414.51901.393.cxt**, varying the base and extinction law. *Left:* The top panel shows the data (O_λ, black) and STARLIGHT fit (M_λ, red). The green line is the $e_\lambda = w_\lambda^{-1}$ error spectrum, with gaps and yellow lines marking bad ($\text{flag}_\lambda \geq 2$) pixels. The bottom panel show the $O_\lambda - M_\lambda$ residual spectrum. Magenta corresponds to masked windows (emission lines), green marks flagged pixels and red crosses mark clipped points. *Right:* Age distribution in the population vector, expressed as a light (top) or mass (bottom) fraction. The "bar-code" panel marks the ages included in the base.

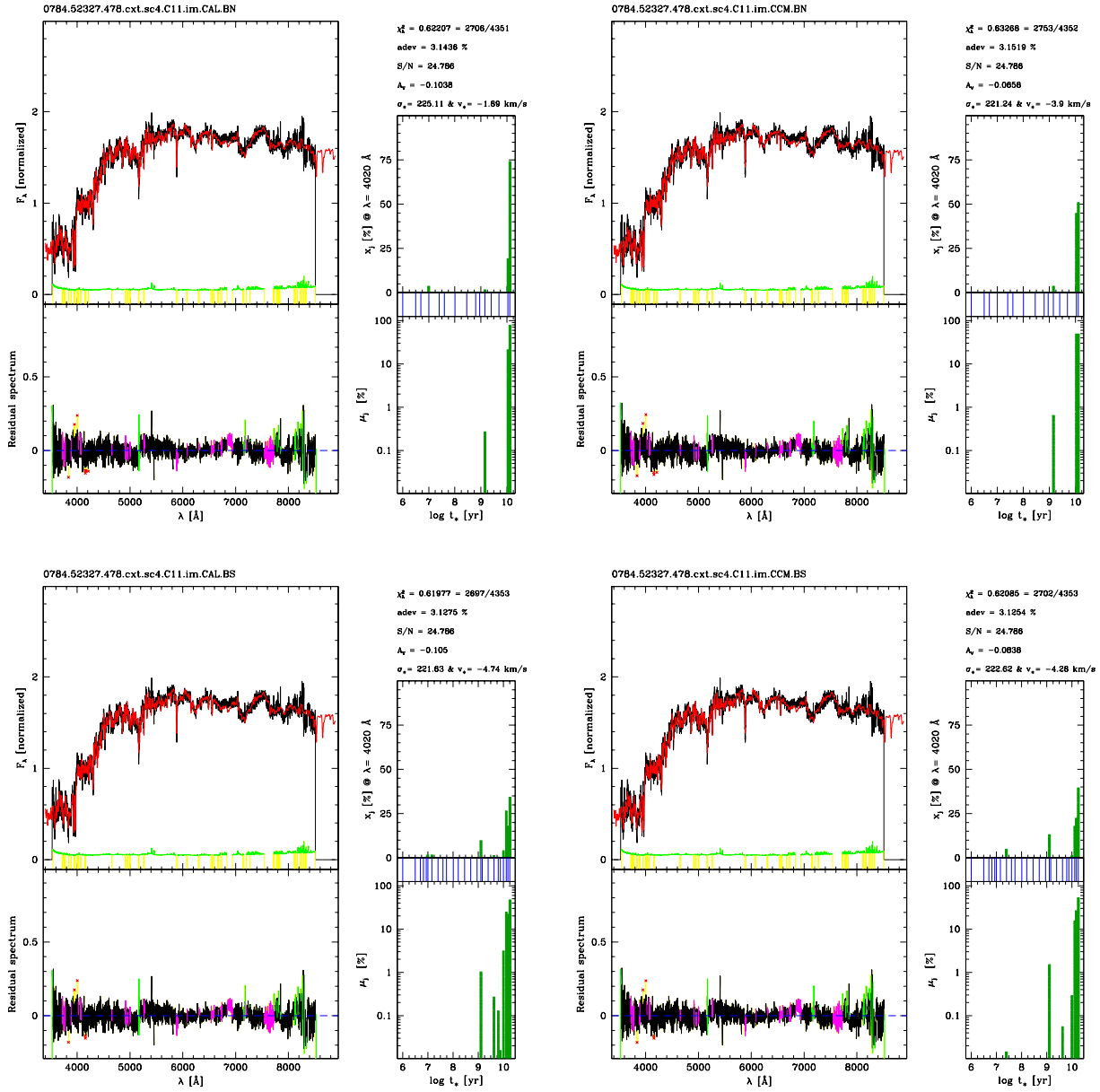

Figure 4: 4 fits of the early type SDSS galaxy **0784.52327.478.cxt**, varying the base and extinction law.

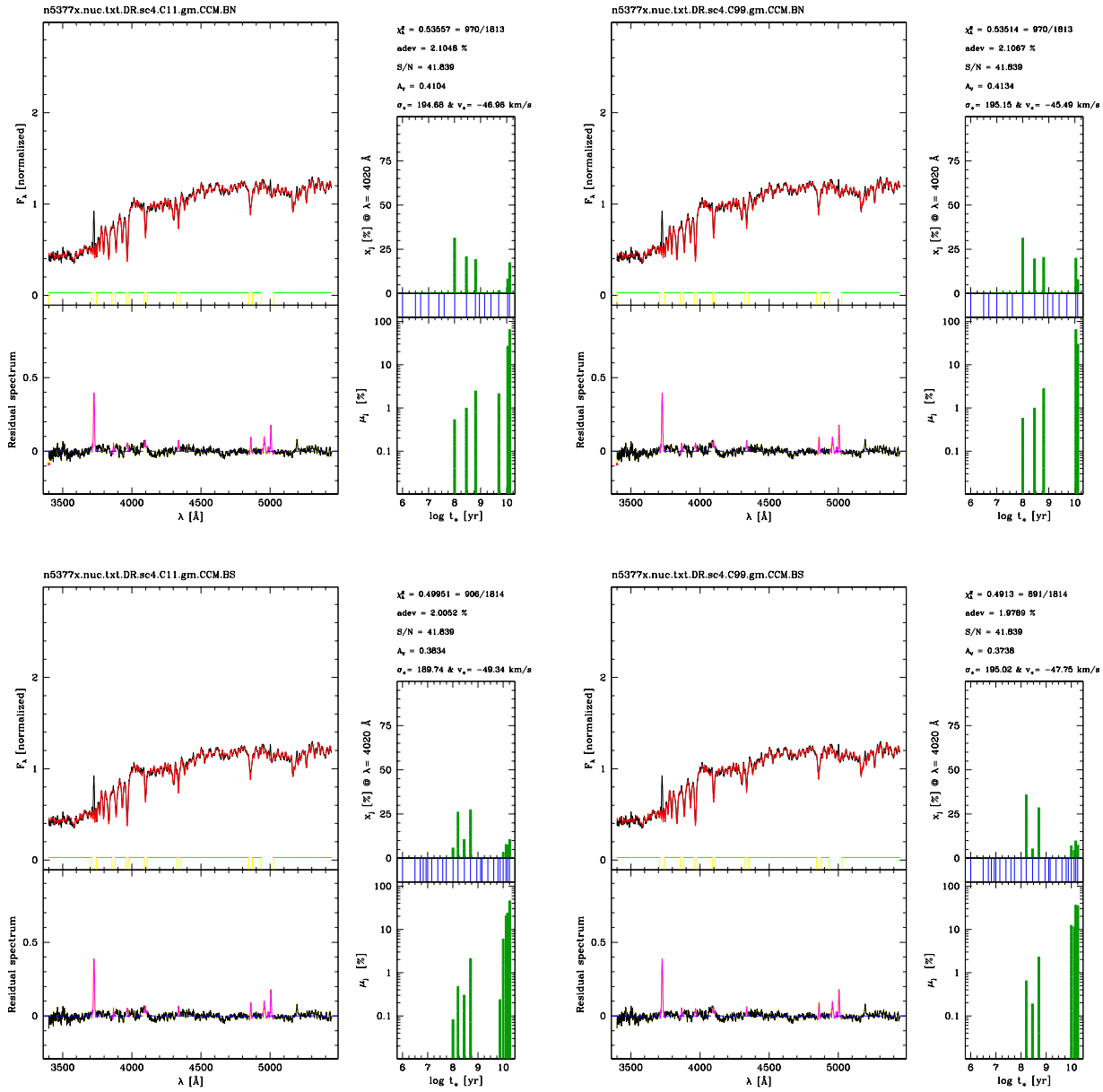

Figure 5: 4 fits of **n5377x.nuc.txt.DR**, varying the base and config. Note the flat e_λ spectrum, because neither error nor flag spectra are available in this case. These fits used the general mask **Masks.EmLines.SDSS.gm**, which does not include [NII]λ5200, present in this LINER/HII Transition Object. (Data from Cid Fernandes et al. 2004a and González Delgado et al. 2004).

10 Wish list

There are several points where STARLIGHT still needs improvements.

• **Multiple Extinctions:** As it is widely known, a single foreground dust screen is a naive model for galaxies. As it is also widely known, going beyond this simplest scenario brings about a number of computational problems and astrophysical degeneracies... In some cases, like ULIRGS and other extremely dusty systems, however, there is no escape but to allow for multiple A_V's if one wants to be remotely realistic.

STARLIGHT internally handles an $N_\star$-dimensional extinction vector $\vec{A_V}$, but on input we allow $A_{V,j}$ to assume only 1 or 2 values (§4.3). As we speak, STARLIGHT is being adapted to allow for up to $N_\star$ extinctions by changing the meaning of the YAV flag in arq_base. We envisage applications with relatively few base elements (small $N_\star$), so that degeneracies between populations and extinctions are held under control.

Until this version is fully debugged and becomes public, a way to fool STARLIGHT is to include reddened spectra *in the base itself*. This has been tried and found to work to a certain extent. The challenge here is to interpret the results quantitatively.

• **Rectified spectra:** Very often the observed spectrum is not flux calibrated, because of changing weather, variations in detector response, difficulties in joining independently taken blue and red spectra, and other calibration issues. In these cases one has to forget the shape of the continuum and analyze only the absorption line information. Rectifying the spectrum is the normal procedure in this case.

Paradoxically, it is harder to deal with this simpler problem. The reason is that rectification invalidates the rapid-χ^2 tricks employed in the code, at least in the way they were coded in. While there are ways of using STARLIGHT to deal with rectified spectra, they are rather cumbersome, so we will not explain them here. A future version will be adapted to deal with these cases.

• **Sampling:** As explained in §4.1.3, our sampling routine needs to be improved.

• **Errors:** The most serious drawback of STARLIGHT, as it is, is that it does not provide error estimates on its parameters. Most likely, we will *never* provide this, because (1) it is an ugly issue, and (2) it is hard (impossible?) to think of an error estimation scheme applicable to all sorts of problems STARLIGHT is expected to cope with. Our suggestion to estimate errors is to do it with Monte Carlo simulations: Perturb your data and fit it many times, and evaluate uncertainties from the ensemble of solutions. A general idea on what sort of errors you should expect can be found in our papers, as well as in those of the "competing" codes, some of which do tackle this issue in a comprehensive manner.

• **SED-fitting:** What if you wanna fit a high resolution optical spectrum plus the H and K band fluxes at the same time? STARLIGHT is not prepared to deal with this. There are ways to model spectra of variable resolution playing with the config parameters and fixing the kinematics, but they

have not been thoroughly tested.

11 Acknowledgements

STARLIGHT has grown over the years from a simple-minded to a general and flexible spectral fitting code. Many people participated either directly or indirectly in this process, contributing with ideas, reporting bugs or inconsistencies, complaints and suggestions. Some of these are F. Barbosa; C. Brandt; P. Coelho; M. Corbin; M. Cerviño; R. Gonzalez Delgado; Q. Gu; D. Kunth; P. Lira; J. Leão; C. Mendes de Oliveira; L. Martins; J. Melnick; E. Perez; I. de la Rosa; H. Schmitt; T. Storchi Bergmann; E. & R. Terlevich. My thanks to all of them.

Thanks also to the people in Semi-Empirical Analysis of Galaxies collaboration, A. Mateus; L. Sodré; G. Stasińska and J. P. Torres-Papaqui; especially my students N. Asari; J. Gomes; M. Schlickmann, W. Schoenell and L. Vieira da Silva.

It is also a pleasure to thank all those who dedicate their time to developing evolutionary synthesis methods and their ingredients, as well as those behind the magnificent public databases of galaxy spectra available nowadays.

The STARLIGHT project is supported by the Brazilian agencies CNPq, CAPES and FAPESP, and by the France-Brazil CAPES/Cofecub program.

12 References

Adams, D. 1979, "The Hitchhiker's Guide to the Galaxy", Pan Books

Allen, C. W. 1976, in "Astrophysical Quantities", University of London eds., The Athlone Press, p. 264

Asari, N. V.; Cid Fernandes, R.; Stasińska, G.; Torres-Papaqui, J. P.; Mateus, A.; Sodré, L. & Schoenell, W. 2007, MNRAS, submitted.

Barbosa, F. K. B., Storchi-Bergmann, T., Cid Fernandes, R., Winge, C., & Schmitt, H. 2006, MNRAS, 371, 170

Bica E., 1988, A&A, 195, 76

Bouchet, P., Lequeux, J., Maurice, E., Prevot, L., & Prevot-Burnichon, M. L. 1985, A&A, 149, 330

Bruzual G., Charlot S., 2003, MNRAS, 344, 1000

Calzetti (astro-ph/9911459), from HyperZ

Cardelli, J. A., Clayton, G. C., & Mathis, J. S. 1989, ApJ, 345, 245

Chabrier G., 2003, PASP, 115, 763

Charlot, S., & Fall, S. M. 2000, ApJ, 539, 718

Cid Fernandes R. 2007a, Proceedings of IAU symposium 241, "Stellar Populations as Building Blocks of Galaxies", in press (astro-ph/0701899)

Cid Fernandes R. 2007b, Boletin de la Asociacion Argentina de Astronomia, vol.49, in press

(astro-ph/0701902)

Cid Fernandes R., Sodré L., Schmitt H. R., Leão J. R. S., 2001, MNRAS, 325, 60

Cid Fernandes, R., et al. 2004a, ApJ, 605, 105

Cid Fernandes, R., Gu, Q., Melnick, J., Terlevich, E., Terlevich, R., Kunth, D., Rodrigues Lacerda, R., & Joguet, B. 2004b, MNRAS, 355, 273

Cid Fernandes R., Mateus A., Sodré L., Stasińska G., Gomes J. M., 2005, MNRAS, 358, 363 (SEAGal I)

Cid Fernandes, R., Asari, N. V., Sodré, L., Stasińska, G., Mateus, A., Torres-Papaqui, J. P., & Schoenell, W. 2007, MNRAS, 375, L16

Corbin, M. R., Vacca, W. D., Cid Fernandes, R., Hibbard, J. E., Somerville, R. S., & Windhorst, R. A. 2006, ApJ, 651, 861

Doran, M. & Müller 2003, astro-ph/0311311

Fitzpatrick, E. L. 1986, AJ, 92, 1068

Gallazzi A., Charlot S., Brinchmann J., White S. D. M., Tremonti C. A., 2005, MNRAS, 362, 41

Garcia-Rissmann, A., Vega, L. R., Asari, N. V., Cid Fernandes, R., Schmitt, H., González Delgado, R. M., & Storchi-Bergmann, T. 2005, MNRAS 359, 765

Gelman, A. & Rubin, D. B. 1992, Statist. Sci., 7, 457.

González Delgado, R. M., Cid Fernandes, R., Pérez, E., Martins, L. P., Storchi-Bergmann, T., Schmitt, H., Heckman, T., & Leitherer, C. 2004, ApJ, 605, 127

Gordon, K. D., Clayton, G. C., Misselt, K. A., Landolt, A. U., & Wolff, M. J. 2003, ApJ, 594, 279

Martins, L. 2005, PhD Thesis, Universidade de São Paulo.

Mateus, A.; Sodré, L.; Cid Fernandes, R.; Stasińska, G.; Schoenell, W. & Gomes, J. M. 2006, MNRAS, 370, 721 (SEAGal II)

Prevot, M. L., Lequeux, J., Prevot, L., Maurice, E., & Rocca-Volmerange, B. 1984, A&A, 132, 389

Seaton, M. J. 1979, MNRAS, 187, 73

Stasińkska G., Cid Fernandes R., Mateus A., Sodré L., Asari N. V., 2006, MNRAS, 371, 972 (SEAGal III)

Vega, L. R. 2004, MSc Thesis, Universidade Federal de Santa Catarina