

# SNANA User's Manual: Simulation, Lightcurve Fitter & Cosmology Fitter

R. Kessler (U.Chicago Dept. of Astronomy)

January 24, 2013

## Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>SNANA Basics</b>	<b>5</b>
2.1	Dr. Evil-ABORT-Face . . . . .	6
2.2	Citations . . . . .	6
<b>3</b>	<b>The Calibration + K-Correction file</b>	<b>7</b>
3.1	HBOOK and FITS Format . . . . .	7
<b>4</b>	<b>The SNANA Simulation: <code>snlc_sim.exe</code></b>	<b>8</b>
4.1	Overview of Model Magnitudes and Noise Calculation . . . . .	8
4.2	Getting Started Quickly . . . . .	9
4.3	Synchronizing Random Numbers . . . . .	10
4.4	Simulating Type Ia Supernova . . . . .	10
4.5	Simulating Non-Ia Supernova (II, Ib, Ic) . . . . .	12
4.5.1	K-correction Tables for Rest-Frame “nonIa” Option . . . . .	14
4.6	The ‘SIMLIB’ Observing Conditions File . . . . .	15
4.6.1	SIMLIB Options in the Sim-Input File . . . . .	16
4.7	Simulating Overlapping Fields . . . . .	18
4.8	Simulating Multiple Instruments . . . . .	18
4.9	Simulating Multiple Seasons . . . . .	20
4.10	Simulating a Filter as a Sum of Components . . . . .	20
4.11	Global Noise Corrections (Optional) . . . . .	21
4.12	Noise Calculation . . . . .	22
4.13	K-corrections . . . . .	24
4.14	Intrinsic Brightness Variations . . . . .	25
4.15	Search Efficiency . . . . .	27
4.15.1	Determining $\epsilon_{spec}$ . . . . .	31

4.15.2	Legacy Spectroscopic Efficiency Options . . . . .	32
4.16	Selection Cuts . . . . .	33
4.17	Varying the Exposure Time/Aperture/Efficiency . . . . .	36
4.18	Simulating the Host Galaxy . . . . .	37
4.19	Simulating the SN Rate: Volumetric and per Season . . . . .	41
4.20	Simulating Galactic Extinction . . . . .	43
4.21	“Perfect” Simulations . . . . .	43
4.22	Redshift-Dependent Parameters . . . . .	44
4.23	Generating Efficiency Maps . . . . .	44
4.24	Light Curve Output Formats . . . . .	46
4.24.1	Verbose Light Curve Output . . . . .	46
4.24.2	Terse Light Curve Output (Default) . . . . .	47
4.24.3	Model-Mag Light Curve Output . . . . .	47
4.24.4	Suppress SIM_XXX Info . . . . .	47
4.24.5	Random CID . . . . .	48
4.24.6	FITS Format . . . . .	48
4.25	Simulation Dump Options . . . . .	49
4.25.1	SIMLIB_DUMP Utility . . . . .	49
4.25.2	Cadence Figure of Merit Utility . . . . .	50
4.25.3	SIMGEN_DUMP File . . . . .	50
4.25.4	Rest-Frame Model Dump . . . . .	50
4.26	Including a Second Sim-Input File . . . . .	51
4.27	Multi-dimensional GRID Option . . . . .	51
<b>5</b>	<b>The SNANA Fitter: snlc_fit.exe</b> . . . . .	<b>54</b>
5.1	Getting Started Quickly . . . . .	54
5.2	Discussion of Lightcurve Fits . . . . .	54
5.3	Methods of Fit-Parameter Estimation . . . . .	55
5.4	Initial Fit-Parameter Estimation . . . . .	56
5.5	Galactic Extinction Uncertainties (as of v9_96) . . . . .	57
5.6	Fitting Priors . . . . .	57
5.7	Selecting an Efficiency Map for a Fitting Prior . . . . .	58
5.8	Viewing Lightcurve Fits: mkfitplots.pl . . . . .	58
5.9	Tracking SN versus Cuts . . . . .	59
5.10	PhotoZ Fits . . . . .	60
5.10.1	Redshift-Dependent Selection in PhotoZ Fits . . . . .	63
5.10.2	Initial Parameter Estimate . . . . .	65
5.10.3	Smooth Model Error Transition Across Filter Boundaries . . . . .	65
5.10.4	Don’t Fool Yourself when PhotoZ-Fitting Simulations . . . . .	65
5.11	Including the $\log(\sigma)$ Term in the $\chi^2$ . . . . .	66
5.12	Optional Redshift Sources . . . . .	67
5.13	Excluding/Downweighting Filters and Epoch Ranges . . . . .	68

5.14	Rest-Frame Wavelength Range . . . . .	69
5.15	Extracting Light Curve Shape from the Fit . . . . .	70
5.16	Landolt ↔ Bessell Color Transformations . . . . .	71
5.17	Interpolating Fluxes and Magnitudes . . . . .	72
5.18	Fitting Rest-Frame Peak-Magnitudes and Colors . . . . .	73
5.19	Selecting Telescope, Field and SNe . . . . .	74
5.19.1	Quickly Analyzing a few SNe from a Large Sample . . . . .	74
5.20	Mag-Shifts in Zero Points and Primary Reference Star . . . . .	75
5.21	Fudging the FLUXCAL Offsets and Uncertainties . . . . .	76
5.22	Updating the Filter Transmission for each SN . . . . .	76
5.23	Analysis Ntuples . . . . .	77
5.24	Analysis Alternative for those with HBOOK-phobia . . . . .	77
5.25	Monitoring Fit-Jobs with “grep” . . . . .	78
5.26	User SN Tags . . . . .	79
<b>6</b>	<b>Private Options</b>	<b>80</b>
6.1	Private \$NON1A_ROOT . . . . .	80
6.2	Creating Your Private Fitter: “snlc_fit_private.exe” . . . . .	80
6.3	Private Data Path: PRIVATE_DATA_PATH . . . . .	81
6.4	Private Model-Path: \$SNANA_MODELSPATH . . . . .	82
6.5	Private Variables in Data Files . . . . .	82
<b>7</b>	<b>Adding a New Survey</b>	<b>83</b>
7.1	Filter Names and Rules for K-corrections . . . . .	84
<b>8</b>	<b>Photometric Classification: psnid.exe</b>	<b>85</b>
<b>9</b>	<b>Light Curve Models</b>	<b>86</b>
9.1	MLCS2k2 . . . . .	87
9.2	SALT-II . . . . .	88
9.3	SNOOPy . . . . .	90
9.4	SIMSED . . . . .	91
<b>10</b>	<b>SALT-II Programs</b>	<b>93</b>
10.1	Computing Distance Moduli from SALT-II fits: SALT2mu . . . . .	93
10.2	SALT-II Training Scripts . . . . .	93
<b>11</b>	<b>Cosmology Fitters</b>	<b>94</b>
11.1	Peculiar Velocity Covariances . . . . .	95
<b>12</b>	<b>Miscellaneous Tools and Features</b>	<b>96</b>
12.1	Analysis Variables, Fitres files and Ntuples . . . . .	96
12.1.1	Combining “Fitres” Files: combine_fitres.exe . . . . .	96

12.1.2	Ntuple ↔ Fitres Format: ntprint.pl and ntdump.pl . . . . .	97
12.1.3	Appending Variables to the Standard Fitres Output: ntappend2fitres.pl . .	98
12.1.4	Extract Value from Fitres File: get_fitresValue.pl . . . . .	99
12.2	General Misc. Tools . . . . .	99
12.2.1	Command-line Overrides . . . . .	99
12.2.2	Synchronizing/Updating Survey Files: survey_update.pl . . . . .	99
12.2.3	Bug-Catcher: the SNANA_tester Script . . . . .	99
12.2.4	Data Backup/Archival: backup_SNDATA_version.cmd . . . . .	100
12.2.5	K-correction Dump Utility: kcordump.exe . . . . .	100
12.3	Misc. Simulation Tools . . . . .	101
12.3.1	Simulate Ia/non-Ia mix: sim_SNmix.pl . . . . .	101
12.3.2	Co-Adding SIMLIB Observations on Same Night: simlib_coadd.exe . . . .	101
12.3.3	Preparing Non-Ia Templates for the Simulation . . . . .	101
12.3.4	Fudging Simulated Errors and Signal-to-Noise Ratio (S/N) . . . . .	101
12.4	Misc. Fitting Tools . . . . .	103
12.4.1	Fit Multiple Samples with Multiple Fit-Options: split_and_fit.pl . . . .	103
12.4.2	Analyzing Residuals from Lightcurve Fits . . . . .	103
12.4.3	Extracting Light Curves into ASCII Formatted Files . . . . .	103
12.4.4	Translating SNDATA files into SALT-II Format . . . . .	104
12.4.5	Translating TEXT data-files into FITS Format . . . . .	104
12.4.6	Fudging Fitting Errors . . . . .	104
12.5	Misc. SIMSED Utilities . . . . .	105
12.5.1	SIMSED Spectrum Extraction: SIMSED_extractSpec.exe . . . . .	105
12.5.2	SIMSED Fudge Afterburner: SIMSED_fudge.exe . . . . .	105
12.5.3	SIMSED Preparation for SNANA: SIMSED_prep.exe . . . . .	105
<b>13</b>	<b>SNANA Updates</b>	<b>106</b>
<b>14</b>	<b>Reporting Problems</b>	<b>106</b>
	<b>References</b>	<b>107</b>

# 1 Introduction

This manual describes how to use the lightcurve simulation and fitter in the SNANA product. This code was originally developed for the SDSS-II Supernova Survey, and then it was modified to simulate and fit SN Ia lightcurves for non-SDSS surveys. Current SN models include MLCS2k2 [1], SALT-II [2], SNooPy [3], stretch [4], and two-stretch[5].

The simulation is designed to be fast, generating a few dozen lightcurves per second, and still provide an accurate and realistic description of supernova lightcurves. In particular, the simulation accounts for variations in noise, atmospheric transmission, and cadence. The reliability of the simulation is based on the accuracy of the “seeing conditions” that describes the seeing, sky-noise, zeropoints, and cadence. The conditions file is easy to prepare post-survey; predicting the conditions file before the survey is crucial to making reliable predictions for the lightcurve quality. This simulation does not use pixels or images directly, although it makes use of information generated from the images.

## 2 SNANA Basics

From any DES or SDSS server at Fermilab, invoke the SNANA product with the command:

```
> setup snana
```

This command defines a few useful global environment variables:

```
> echo $SNANA_DIR/  
/sdss/ups/prd/snana/v8_04/Linux
```

```
> echo $SNDATA_ROOT/  
/data/dp62.b/data/sn/data/
```

At non-FNAL sites, you will have to define \$SNANA\_DIR and \$SNDATA\_ROOT as part of the installation. The environment variable \$SNANA\_DIR points to the software that is accessible to everyone, but write-protected. The SNANA developers use a cvs repository to share code, and an updated version is “cut” (i.e, released) on occasion to provide a stable software version for collaborators. SNANA is re-released as often as necessary (§ 13), as bugs are fixed and improvements are made. The current software version is v8\_04 as indicated by \$SNANA\_DIR. You will likely have a future software version when you read this manual. If there is a problem with the current software version, you can fall back to an earlier version with

```
> setup snana v3_03
```

The main source codes for the simulation and fitter are

```
> $SNANA_DIR/src/snlc_sim.c  
> $SNANA_DIR/src/snlc_fit.car
```

The environment variable \$SNDATA\_ROOT points to the user area. Everyone has write privilege here, so please be careful. The contents are explained in \$SNDATA\_ROOT/AAA\_README.

## 2.1 Dr. Evil-ABORT-Face

The SNANA simulation and fitter programs have intensive error checking throughout the execution. If anything looks fishy, the program will abort with a message looking like

```
FATAL[get_user_input]: Cannot open input file :  
                        'sim_BLABLA.input'
```

```
\ | \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ | \  
< | o \ / o | >  
  | ' ; ' |  
  | _____ |  
  | ' ' |  
  | \ - - - / |  
  | _____ |  
  \ | _____ /
```

ABORT program on Fatal Error.

While some of these aborts may at first seem frustrating, they are crucial for catching bugs as early as possible so that you don't waste months (years) doing something silly.

## 2.2 Citations

While an SNANA citation is always appreciated ([6]), please make sure to reference any underlying work that is used by SNANA. Referencing the appropriate light curve model (§1) is the most obvious example. However, there may be other features to reference such as the source of spectra for the SIMSED model, survey efficiencies, host-galaxy correlations, etc ... If you read a published article and suspect SNANA usage with a missing reference, please contact the lead author and one of the SNANA authors.

### 3 The Calibration + K-Correction file

Before running the simulator or light curve fitting program, a “K-correction” file must be created. This file contains

- filter transmissions.
- native mag for each filter.
- SED of the primary reference (i.e., AB, BD17, ...) and each SN epoch.
- zeropoint offsets (native – synthetic mags)
- Optional AB offsets (to apply to data)
- for rest-frame models requiring K-correction,
  - K-correction tables vs. redshift, epoch,  $A_V$ -warp.
  - rest-frame magnitudes.
  - Galactic extinction corrections.

The purpose of this file is two-fold: (1) collect the relevant information in one file that can be used for any model such as SALT2 or mlcs2k2, and (2) for K-corrections, pre-compute quantities that vastly speeds up the simulation and fitting programs. The  $A_V$ -warp parameter warps the SN SED to match a grid of colors, and is used to quickly find the warped SN SED that matches the observed colors. Example kcor-input files are here: `$SNDATA_ROOT/analysis/sample_input_files/kcor` and the command to create a K-corr file is

```
kcor.exe myKcor.Input
```

#### 3.1 HBOOK and FITS Format

The output file is specified by the kcor-input file key “OUTFILE: \$outfile,” although the legacy key “OUTFILE\_GRID\_HIS:” is still supported. The original format is a collection of HBOOK histograms using CERNLIB; this format is somewhat clumsy for mass storage, but convenient for making quick plots. Since CERNLIB support ceased in 2006, a newer FITS format is available (since v10\_21b) that does not depend on CERNLIB. The formats are specified by the extension of the outfile; “.his” or “.HIS” for hbook and “.fits” or “.FITS” for the fits format. One or both outputs can be specified,

```
OUTFILE: mySurvey.his
OUTFILE: mySurvey.fits
```

Since the fits-format option is new (Jan 2013), we recommend generating both outputs and testing that using either file gives the same result in the simulation and fitting programs.

Lastly, the `HFILE_KCOR` namelist variable for the `snlc_fit.exe` program is still allowed, but we recommend using the newer `KCOR_FILE` variable (it also has the same name as the corresponding simulation key).

## 4 The SNANA Simulation: `snlc_sim.exe`

### 4.1 Overview of Model Magnitudes and Noise Calculation

For a rest-frame model of supernova, such as `MLCS2k2` or `SNooPy`, here is a brief overview of how the simulation generates observed fluxes in CCD counts,

1. pick random luminosity parameter (e.g.,  $\Delta$ ,  $\Delta M_{15}$ ) and random extinction ( $A_V$ ) according to measured distributions.
2. generate rest-frame light curve:  $U, B, V, R, I$  mag vs. time.
3. apply host-galaxy extinction to  $UBVRI$  mags.
4. add K-correction to transform  $UBVRI$  to observer-frame filters.
5. apply Galactic (MilkyWay) extinction.
6. apply zero-points to translate generated magnitude into CCD counts; this step account for atmospheric transmission and telescope efficiency.

For an observer-frame mode such as `SALT-II`, steps 2-4 are replaced by a function that generates observer-frame magnitudes.

The noise in the simulation is computed as follows,

$$\sigma_{\text{SIM}}^2 = [F + (A \cdot b) + (F \cdot \hat{\sigma}_{\text{ZPT}})^2 + (\hat{\sigma}_0 \cdot 10^{0.4 \cdot \text{ZPT}_{\text{pe}}})^2 + \sigma_{\text{host}}^2] \hat{S}_{\text{SNR}}^2 \quad (1)$$

where

- $F$  is the simulated flux in photoelectrons (p.e.).
- $A$  is the noise-equivalent area given by  $[2\pi \int PSF^2(r, \theta) r dr]^{-1}$ .
- $b$  is the background per unit area (includes sky + CCD readout + dark current).
- $\hat{\sigma}_{\text{ZPT}}$  is the zeropoint uncertainty.
- $\hat{\sigma}_0$  is a constant `FLUXCAL` uncertainty, and  $\text{ZPT}_{\text{pe}}$  transforms  $\hat{\sigma}_0$  into an uncertainty in p.e.
- $\hat{S}_{\text{SNR}}$  is an empirically determined scale that depends on the signal-to-noise ratio (SNR)
- $\sigma_{\text{host}}$  is from the underlying host galaxy.

The terms with hats ( $\hat{\sigma}_0$ ,  $\hat{\sigma}_{\text{ZPT}}$ ,  $\hat{S}_{\text{SNR}}$ ) may be difficult to compute from first principles, but these terms can be determined empirically from fits that match simulated uncertainties to those from the data. The  $A, b, \hat{\sigma}_{\text{ZPT}}$  terms are discussed in §4.6. The  $\hat{\sigma}_0$ ,  $\hat{S}_{\text{SNR}}$  terms are discussed in §4.11. The host-galaxy noise ( $\sigma_{\text{host}}$ ) is discussed in §4.18.

## 4.2 Getting Started Quickly

In this section, you should be able to start simulating lightcurves in a few minutes. There are many options that may take some practice to use properly. The first step is to copy a sample input file to your private area,

```
> cp $SNDDATA_ROOT/analysis/sample_input_files/mlcs2k2/sim_[SURVEY].input.
```

where [SURVEY] is one of the surveys for which a sample sim-input file is available. Edit the file and change the GENVERSION name:

```
GENVERSION: CHANGE_ME
```

We recommend just adding your initials and/or project name so that you do not over-write somebody else's files. Now you can run the simulation, for example, with

```
> snlc_sim.exe sim_SDSS.input
```

which should generate ten lightcurves using the MLCS2k2 model. The next step is to modify the input file to suit your needs. The input parameters are internally commented within the source code; for example, to get more information about the GENMAG\_SMEAR keyword,

```
> grep GENMAG_SMEAR $SNANA_DIR/src/snlc_sim.h
> grep GENMAG_SMEAR $SNANA_DIR/src/snlc_sim.c
```

will help you trace the meaning of this variable. All of the input options are defined in a structure called INPUTS in snlc\_sim.h. Please report variables that are not commented, or that have confusing comments. Don't hesitate contacting other people familiar with snlc\_sim.exe. Some of the light curve parameters are obscure (i.e, describing the distribution of extinction &  $\Delta$ ), and our best estimates of these parameters can change. To reduce the burden of tracking all of these parameters, defaults for these obscure parameters are stored in the file

```
$SNDDATA_ROOT/models/snlc_sim.defaults .
```

If you simply ignore these obscure parameters in your input file, the "defaults" defined above will be used in the simulation. You can modify any parameter by defining the parameter explicitly in your input file.

The simulated lightcurves are located in \$SNDDATA\_ROOT/SIM. Do NOT try 'ls' !!! Instead, try 'ls -d \*/' to see all versions. To avoid sifting through hundreds of versions from multiple users, I always use 'RK' as a prefix for my versions, and therefore I can see my versions with 'ls -d RK\*/.' Each simulated version is located in a sub-directory named by your version. Recall that you have full write-privilege, so use caution. Each version has an auto-generated README file. If your version is called 'MYFIRSTSIM', then do

```
> more $SNDDATA_ROOT/SIM/MYFIRSTSIM/MYFIRSTSIM.README
```

which contains a list of all your simulation options. The default format is FITS: one FITS file for the header info with one row per SN, and a second FITS file with all of the light curves. Instead of FITS format, you can generate a text file per SN with the option “FORMAT\_MASK: 2” (§4.24). The SNANA fitting programs read both the FITS and TEXT formats. You can get a list of files with the commands

```
cd $SNDATA_ROOT/SIM/MYFIRSTSIM/  
ls MYFIRSTSIM_SN*  
or  
more MYFIRSTSIM.LIST
```

### 4.3 Synchronizing Random Numbers

The simulation is designed to preserve the random number sequence when input parameters and options are changed. This feature allows generating the exact same SNe (redshift, sky-coords, SN properties) when making changes such as the SIMLIB, exposure time, mag-offsets, intrinsic smearing, and model parameters. To ensure that different simulations are synchronized to the same random numbers, use the same random seed (RANSEED key) and verify that the following output to the README file is identical:

```
Random Number Sync:  
RANDOM SEED: 123459  
FIRST/LAST Random Number: 0.181881 0.649706
```

Selection cuts may result in a different number of generated SNe, and hence a different last random number; in this case use the SIMGEN\_DUMP option (§4.25.3) to verify the sync.

### 4.4 Simulating Type Ia Supernova

The simulation of Type Ia supernova involves the selection of one of the following models:

```
GENMODEL: mlcs2k2 # JRK 2007  
GENMODEL: SALT2 # Guy 2007  
GENMODEL: snoopy # Burns 2010  
GENMODEL: stretch # stretch a template  
GENMODEL: stretch2 # stretch pre- and post-max  
KCOR_FILE: <kcor filename>
```

The lower-case models indicate that rest-frame magnitudes are generated and K-corrections are used to transform into the observer-frame. Upper case (SALT2) indicates that observer-frame magnitudes are generated directly from the model without the need for K-corrections. The KCOR\_FILE must reside in \$SNDATA\_ROOT/kcor, and must be specified for all models, even observer-frame models such as SALT2. Although SALT2 does not use K-corrections, it uses the filter transmission curves and primary reference spectrum stored in the KCOR\_FILE, and thus ensures that the same information is used for all models that point to the same KCOR\_FILE.

For each Ia model, there are parameters to control the distributions of the SN brightness and color. Default parameters are given by `GENMEAN_XXX`, `GENRANGE_XXX` and `GENSIGMA_XXX` in the default sim-input file:

```
$SNDATA_ROOT/models/snlc_sim.defaults ,
```

where `xxx` refers to the Ia model that is chosen. Since the extinction ( $A_V$ ) profile is exponential instead of Gaussian, the  $A_V$  profile is controlled by the parameters `GENRANGE_AV` and `GENTAU_AV`, where the latter is the exponential slope. You can change any default parameter by adding it in your private sim-input file or using the command-line override. Note that there are two `GENSIGMA_XXX` values to specify a bifurcated Gaussian with peak at `GENMEAN_XXX`. The `GENRANGE_XXX` specify parameter ranges to avoid extreme or unphysical values.

Aside from the profile-parameters above, the SN model parameters from the training reside in

```
$SNDATA_ROOT/models/[mlcs2k2,stretch,SALT2_templates]
```

For `mlcs2k2`, you can specify the default with “`GENMODEL: mlcs2k2`”, or specify any option in the `/mlcs2k2` subdirectory. The `SALT2` templates are fixed as of this version, but options may be available later. For the `stretch` and `stretch2` models you must specify a template in the sim-input file with

```
STRETCH_TEMPLATE_FILE: <mytemplate.dat>
```

The simulation will first check if this file exists in `$SNDATA_ROOT/models/stretch`; if not, then the simulation will check your current working directory. If the template file cannot be found, the simulation will abort.

You can adjust the rise-time for any model using the following sim-input parameters:

```
GENMEAN_RISETIME_SHIFT: 2.3      # shift at -18 days
GENSIGMA_RISETIME_SHIFT: 0.6 0.6
GENRANGE_RISETIME_SHIFT: -4. 4.
```

In the above example, the rest-frame rise-time at each epoch is increased by  $2.3 \times T_{\text{rest}}/18$  days with a Gaussian sigma of 0.6 days, and shifts past  $\pm 4$  days are excluded. The rise-time adjustments can be used, for example, to generate a double-stretch model by simulating two separate samples, each with a different rise-time shift.

Finally, select the Ia integer-type-code with input key “`SNTYPE_Ia`” (default Ia type is 1). For example, the SDSS–II code for type Ia is

```
SNTYPE_Ia: 120
```

and this code appears in the header of each output data file after the “`SNTYPE:`” key.

## 4.5 Simulating Non-Ia Supernova (II, Ib, Ic)

While the Type Ia light curve models are based on a parametric equation or photometric templates, the non-Ia models are based on smoothed light curves and spectral templates. A spectral template can be a smoothed average from an ensemble of observations, or a template can correspond to a particular (well-observed) supernova where a composite spectrum is warped to match the observed photometric colors. The `snlc_sim.exe` simulation is designed to take full advantage of both types of templates. In addition to non-Ia SNe, this feature can be used to simulate peculiar Ia, theoretical models, AGN, or any transient object.

There are two methods for generating non-Ia light curves

```
GENMODEL: nonIa # (or nonIa) rest-frame mag + K-correction
GENMODEL: NONIa # (or NONIa) compute observer-mag from SED (like SALT2)
```

The rest-frame “nonIa” model is useful if the SED has not been warped to match the photometry of the corresponding light curve, or to accurately simulate host-galaxy extinction using CCM89 (i.e., specifying  $R_V$  and an  $A_V$  distribution). A potential disadvantage of this method is the need to generate many K-correction tables. The observer-frame “NONIA” model is useful if each SED has already been warped to match the photometry of the underlying light curve. The observer-frame magnitudes are computed directly from the SED the same way as for SALT2, and in fact using the same software tools. Since no K-corrections are needed, the user can point to the SN Ia K-corr table to read in the filters and primary reference. Starting at `v9_84` host-galaxy extinction (from  $R_V$  and  $A_V$ ) is computed at  $\bar{\lambda}_{\text{obs}}/(1+z)$ , where  $\bar{\lambda}_{\text{obs}}$  is the central wavelength of each filter; this approximation is numerically accurate to about 0.01 mag. For the remainder of this section, the instructions apply to both the nonIa and NONIa options, except for brief mentions of nonIa K-correction tables.

A list of available non-Ia templates is given in

```
$SNDATA_ROOT/snsed/nonIa/NONIA.LIST .
```

As explained below, the user selects non-Ia templates using the integer indices in the `NONIA.LIST` file. For the rest-frame nonIa option, a K-correction table is needed for each SED template as described in §4.5.1. As more non-Ia templates become available, the `NONIA.LIST` file and K-corrections will be updated by the relevant experts. The simplest way to select non-Ia from the sim-input file is as follows,

```
GENMODEL: NONIA
NONIA:     0 1. # index and wgt
```

where specifying an INDEX of zero is an instruction to use all available non-Ia (in the `NONIA.LIST` file) with equal weight. Inside each generated light curve file, the non-Ia INDEX is specified by “SIM\_NONIA: INDEX”; the corresponding SNANA variable is `SIM_NONIA(isn)`, and the `fitres-ntuple` (ntid 7788) variable is “nonIa.” Although this “all” option is convenient to quickly study Ia-contamination using all available non-Ia templates, it is not very convenient for normalizing the individual types.

To specify normalizations and simulation parameters that depend on the nonIa type,

```
NGENTOT_LC: 1000
GENMODEL:   NONIA # or NONIa or nonIa
```

NON1A_KEYS: 5	INDEX	WGT	MAGOFF	MAGSMEAR	SNTYPE
NON1A:	2	0.2	0.0	1.2	2
NON1A:	3	0.2	-0.2	0.8	2
NON1A:	4	0.4	-0.6	0.9	3

Only the GENMODEL argument distinguishes upper and lower case; the other keys all use upper-case NON1A. Since the weights (WGT) are re-normalized to sum to unity, the first 1/4 of the generated SNe will use INDEX = 2, the second 1/4 will use INDEX = 3, and the latter 1/2 will use INDEX = 4. The keyword NGENTOT\_LC specifies the total number to generate, which is different from NGEN\_LC that specifies the number passing trigger & cuts and that are written out to SNDATA files. The relative WGT factors make physical sense only if NGENTOT\_LC is used to specify the statistics.

The NON1A\_KEYS are used to specify additional parameters that depend on nonIa type. MAGOFF is a global magnitude offset applied to all passbands (i.e, equivalent to the GENMAG\_OFF\_MODEL keyword), and is used to adjust the average brightness of each nonIa type. For SEDs that are normalized to have a peak mag of zero (i.e., the Nugent templates) rather than to physical units ( $\text{erg/s}/\text{\AA}/\text{cm}^2$ ), MAGOFF must be used for the NON1A option to get meaningful results. For the nonIa model the SEDs are used only for K-corrections, and therefore the SED normalization does not matter. MAGSMEAR is a Gaussian  $\sigma$  for global coherent magnitude fluctuations (i.e, equivalent to GENMAG\_SMEAR), and is used to simulate intrinsic brightness variations. SNTYPE is a spectroscopic typing-index that appears after the SNTYPE keyword in the SNDATA files.<sup>1</sup> In the above example, non-Ia indices 2 & 3 are both type II SNe, so they both get SNTYPE=2. Non-Ia index 3 is a different SN type, so it gets a different SNTYPE value. SNTYPE=1 is always reserved for type Ia.

Additional parameters can be added to the software as needed. With no keys specified, the default is two keys: INDEX & WGT. This default ensures that old sim-input files (i.e, from SNANA v8\_16 and earlier) will work. Also note that INDEX & WGT are required to be the first two keys in the list; if not, the simulation will abort with an error message.

In this example, the end of the README file will show the statistics for each non-Ia INDEX as follows:

NON1A-SUMMARY vs. INDEX:										
INDEX(TYPE)	NGEN written	NGEN total	SEARCH Effic	CID-range	Rest Peakmag+19	a	b	c	d	e
002 ( IIn)	132	333	0.396	1 - 40333	1.70	1.87	2.11	2.30	2.52	
004 ( IIL)	95	667	0.142	40334 - 41000	2.88	1.99	1.88	2.11	2.13	

Note that the column under “NGEN total” sums to the requested number of generated SNe (1000), and the column under “NGEN written” shows how many SNe pass trigger & selection cuts and were thus written to SNDATA files. The Search-Effic, CID-range and Rest-Peakmags are printed for convenience. SNe with the same INDEX are generated sequentially, and then the next INDEX is generated; i.e., *the non-Ia indices are NOT randomly mixed*. The sequential generation by INDEX is done for speed, and avoids re-initializing templates multiple times. It is therefore crucial to analyze the *entire* simulated sample in order to have the correct mixture of nonIa types.

<sup>1</sup>You can set the Ia type with input key “SNTYPE\_Ia” (default Ia type is 1)

#### 4.5.1 K-correction Tables for Rest-Frame “nonIa” Option

To simulate a rest-frame non-Ia model that includes host-galaxy extinction, K-correction tables are needed for each non-Ia SED. These K-cor tables are not included in the public `$SNANA_ROOT` because generating them for each survey would significantly increase the size of the tarball to download. There is an SNANA script to automatically generate all of K-cor tables (one table per nonIa SED) with the appropriate names assumed by the simulation:

```
$SNANA_DIR/util/kcor_gen_non1a.cmd,
```

where the usage instructions are at the top of the file. Note that the first step is to

```
cd $NON1A_ROOT/snsed/non1a/KCOR_GEN
```

where there are example kcor-input files for several surveys.

If you do NOT specify a `KCOR_FILE` in the sim-input file (recommended default), then the default K-correction file will be used automatically for each non-Ia INDEX. If you specify a `KCOR_FILE`, then this K-correction file will be used for all non-Ia indices.

## 4.6 The ‘SIMLIB’ Observing Conditions File

A user-generated ‘SIMLIB’ file<sup>2</sup> is needed to define the cadence, translate magnitudes into CCD counts, and to compute the uncertainty as described in Eq. 1. As an example, the start of the SIMLIB file for the SDSS-II 2005 survey is shown in Fig. 1. The public SIMLIB files are located in \$SNDATA\_ROOT/simlib, and a SIMLIB file is specified in the sim-input file with the keyword

```
SIMLIB_FILE: SDSS2005_ugriz.SIMLIB
```

The simulation will first check YOUR current working directory for this file; if it’s not there, then `snlc_sim.exe` will check the public directory \$SNDATA\_ROOT/simlib.

A SIMLIB file is created by someone with knowledge of the telescope and observation conditions. There is a convenient utility, `SNANA_DIR/src/simlib_tools.c`, that you can use to create the SIMLIB file in the correct format. This utility has a lot of error checking to prevent you from accidentally writing absurd values like a negative PSF. In principle, a SIMLIB file need be created only once per survey, although systematic studies may require multiple SIMLIBs. If there are several exposures per filter per night, the utility `simlib_coadd.exe` (§12.3.2) will re-make a SIMLIB with all exposures per filter combined into a single effective exposure per night.

For a non-overlapping field, the “FIELD:” header should appear only once per MJD-sequence. For overlapping fields, the FIELD key appears more than once as indicated in Fig. 1. You can repeatedly toggle between two fields, or simply list all the MJDs for one field (i.e, 82N) followed by all of the MJDs for the other field (i.e., 82S); the MJDs need not be chronological in the SIMLIB, as the simulation will sort them internally. You can ignore the FIELD key in the SIMLIB as well, in which case the SNDATA files and analysis lose track of the field.

The simlib header values for RA, DECL have units of degrees, the PIXSIZE value is in arcseconds, and MWEBV is the  $B - V$  color excess due to Galactic extinction. If MWEBV is zero, this is a flag for the simulation to use the default dust map from [7].

Below is a brief explanation for each column in the SIMLIB file, along with references to terms in Eq. 1,

1. **S: or T:** refers to Search (required) or Template (optional). Template information is used to add sky noise that results from an image subtraction. Instead of including template info, you could simply increase the search-image skynoise or use the  $\hat{\sigma}_0$  and  $\hat{S}_{\text{SNR}}$  terms described in §4.11.
2. **IDEXPT:** arbitrary identifier. You can set this to zero if you want. For SDSS, it glues together the run and field numbers.
3. **FLT:** single character to specify a filter for this observation.
4. **CCD Gain:** Number of photo-electrons per ADU or DN.
5. **CCD Noise:** CCD read noise in photo-electrons, per pixel. This term is usually much smaller than the SKYSIG term below.

---

<sup>2</sup>“SIMLIB” is an abbreviation for SIMulation LIBrary.

6. **SKYSIG:** Standard deviation of sky (including dark current), in ADU (or DN) per pixel. See §4.12 for noise calculation. You can optionally enter the skysig values per arcsec<sup>2</sup> by specifying the simlib header key

SKYSIG\_UNIT: ADU\_PER\_SQARCSEC

The simulated README file includes the SKYSIG\_UNIT value.

7. **PSF1,2 and RATIO:** The PSF is specified by a double-Gaussian with  $\sigma$ -widths (pixels) of  $\sigma_1 = \text{PSF1}$  and  $\sigma_2 = \text{PSF2}$ . **RATIO** refers to the ratio at the origin,  $\text{PSF2}(\text{origin})/\text{PSF1}(\text{origin})$ . Note that the default PSF unit is in pixels, not arcsec. You can optionally give the PSF values in the more astronomy-friendly units of arcsec-FWHM by specifying the simlib header key

PSF\_UNIT: ARCSEC\_FWHM

The simulated README file includes the PSF\_UNIT value. These PSF values are used to determine a noise-equivalent area ( $A$  in Eq. 1 and Eq. 6). If you have computed  $A$  from the measured PSF, then you can simply define  $\text{PSF1} = \sqrt{A/4\pi}$  and set **PSF2=RATIO=0**.

8. **ZPTAVG:** Zero point relating the source magnitude ( $m$ ) to the CCD flux measured in ADU:

$$\text{Flux(ADU)} = 10^{-0.4(m - \text{ZPTAVG})} .$$

For example, if  $F_{20}$  is the flux (in ADU) for a point source with  $\text{mag} = 20$ , then **ZPTAVG** =  $20 + 2.5 \log_{10}(F_{20})$ . Note that **ZPTAVG** encodes information about the atmospheric transparency, telescope aperture & efficiency, and the exposure time. For any given simlib file, the simulated **ZPTAVG** can be changed globally or by filter as explained in §4.17. Note that the zeropoint in photoelectrons is given by  $\text{ZPT}_{\text{pe}} = \text{ZPTAVG} + 2.5 \log_{10}(\text{GAIN})$ .

9. **ZPTSIG:** See  $\hat{\sigma}_{\text{ZPT}}$  term in Eq. 1.

A sequence of MJDs that span the survey constitutes one entry in the SIMLIB, and the index “LIBID” labels each entry. A SIMLIB can have one LIBID, or hundreds. Large-area surveys, like SDSS-II, need hundreds of LIBIDs to properly sample the sky. A small area survey, like DES, may need just one LIBID per pointing, and per season.

#### 4.6.1 SIMLIB Options in the Sim-Input File

```
SIMLIB_IDSTART: nnn      # start at LIBID nnn
SIMLIB_IDLOCK:  nnn      # use only LIBID nnn ; skip all others
SIMLIB_NSkipMJD:  n       # use every 'n+1'th observation
SIMLIB_IDSKIP:  nn1       # ignore LIBID nn1
SIMLIB_IDSKIP:  nn2       # ignore LIBID nn2 (add as many as you want)
SIMLIB_DUMP:     0        # dump summary of simlib
USE_SIMLIB_PEAKMJD: 1     # use peakMJD in header (if there)
USE_SIMLIB_REDSHIFT: 1    # use redshift in header (if there)
```

```

SURVEY: SDSS      FILTERS: gri
USER: rkessler    HOST: sdssdp47.fnal.gov
COMMENT: 'Extract random RA,DECL,MJD from MYSQL: year=2005'
BEGIN LIBGEN Tue Apr 17 13:32:33 2007

# -----
LIBID: 1
RA: 26.430172    DECL: 0.844033    NOBS: 42    MWEBV: 0.026    PIXSIZE: 0.400
FIELD: 82N

#          CCD  CCD          PSF1 PSF2 PSF2/1
#  MJD      IDEXPT FLT GAIN NOISE SKYSIG (pixels)  RATIO  ZPTAVG ZPTSIG
S: 53616.383 556600405 g  4.05  4.25  4.04  1.85 3.61 0.247 28.36 0.020
S: 53616.383 556600405 r  4.72  4.25  5.28  1.64 3.62 0.142 28.17 0.022
S: 53616.383 556600405 i  4.64 12.99  6.95  1.60 3.81 0.103 27.84 0.017
FIELD: 82S
S: 53622.395 558200552 g  4.03  5.45  4.09  1.58 3.31 0.107 28.45 0.018
S: 53622.395 558200552 r  4.89  4.65  5.00  1.46 3.55 0.065 28.15 0.028
S: 53622.395 558200552 i  4.76 10.71  6.43  1.53 3.65 0.075 27.85 0.029
S: 53626.359 560300625 g  4.05  4.25  4.40  1.83 3.50 0.282 28.24 0.020
etc ...

```

Figure 1: Header and part of first entry of the SIMLIB file used for the SDSS-II survey.

More information about the SIMLIB\_DUMP option is in §4.25.1.

## 4.7 Simulating Overlapping Fields

The simulation can handle overlapping fields, such as for the SDSS **82N/82S** overlap, and the 20% overlap of the LSST fields. Overlapping fields are specified in the SIMLIB file by specifying the FIELD keyword as needed. Figure 1 above illustrates an overlap between the SDSS fields **82N** and **82S**. Note that overlapping fields make no sense if there is just one simlib entry per field with the position selected at the (non-overlapping) center of the field. To generate light curves in overlapping fields, the SIMLIB should include many LIBID entries per field, where each LIBID is associated with a random RA & DEC. This mechanism accounts for dithering as well as variations within a field from Galactic extinction and observing conditions.

If the fields are small and non-overlapping (e.g., SNLS), then a single LIBID per field, with the RA & DEC at the center, may work reasonably well. However, this simlib will not probe variations in Galactic extinction that can occur even on small angular scales.

The &SNLCINP namelist includes two FIELD-selection variables that work for both data and simulations. First you can pick specific fields with

```
SNDFIELD_LIST = 'field1', 'field2', 'field3'
```

By default all fields are analyzed when running the fitting program. The second option is CUTWIN\_NFIELD so that you can select SN that overlap more than 1 field.

## 4.8 Simulating Multiple Instruments

SN photometric observations may come from more than one instrument, such as optical and infrared observations. To simulate all of the light curves together, each simlib entry can contain information from multiple telescopes. The only caveat is the that telescope name and pixel size must be re-defined as illustrated in Fig. 4.8.

Since the default units for the noise (CCD and SKY) and the PSF are both in pixels, the PIXSIZE value has no practical effect. However, when using optional units of arcsec (§4.6), the correct PIXSIZE values are important.

```

LIBID: 4
FIELD: F4 RA: 0.50 DECL: -43.0 MWEBV: 0.008
TELESCOPE: CTIO PIXSIZE: 0.270 asec
NOBS: 120
#          CCD CCD          PSF1 PSF2 PSF2/1
#      MJD  IDEXPT FLT GAIN NOISE  SKYSIG  (pixels)  RATIO ZPTAVG ZPTSIG MAG
S: 56249.039 1002  g  1.00 10.00  91.90  1.93 0.00 0.000  33.02  0.020  99.
S: 56249.000 1003  r  1.00 10.00 151.40  1.49 0.00 0.000  33.29  0.020  99.
S: 56249.008 1004  i  1.00 10.00 275.66  1.62 0.00 0.000  33.42  0.020  99.
S: 56249.016 1005  z  1.00 10.00 442.18  1.40 0.00 0.000  34.31  0.020  99.
TELESCOPE: VIDEO PIXSIZE: 0.339
S: 56250.323 1006  Y  4.0 160.0 114.22  1.09 0.0  0.0   31.70  0.010  99.
S: 56256.033 1007  J  4.0 160.0 282.14  1.16 0.0  0.0   31.99  0.010  99.

```

Figure 2: Excerpt from SIMLIB with two instruments: DES optical (*griz*) and VIDEO infrared (*YJ*).

## 4.9 Simulating Multiple Seasons

Multiple seasons can be simulated with a separate SIMLIB file for each season, but this strategy requires multiple generations and bookkeeping to generate a full multi-season sample. An alternative is to construct a single SIMLIB file containing all seasons, either as separate LIBID entries for each season or as long LIBID entries that each spans all of the seasons/years.

For the latter option using a single SIMLIB for multiple seasons, there are typically long MJD gaps with no observations, leading to inefficient generation. MJD masks can be specified in the SIMLIB header, as illustrated here for the SDSS-II,

```
GENSKIP_PEAKMJD: 53710 53970 # skip the off-season
GENSKIP_PEAKMJD: 54070 54340 # idem
```

Each GENSKIP\_PEAKMJD key must appear before the “BEGIN LIBGEN” key, and it specifies an MJD-range to ignore in the generation.

## 4.10 Simulating a Filter as a Sum of Components

There are some cases where the flux in a filter should be simulated as a sum of components in order to avoid ambiguities in the zeropoint. Examples are red-leakage in a UV filter, cross-correlation filters, or a very broad filter. If the filter transmission for '0' is the sum of filter-transmissions 1+2+3+4, the following SIMLIB entries are needed:

```
#          CCD  CCD          PSF1 PSF2 PSF2/1
#          MJD  IDEXPT FLT GAIN NOISE  SKYSIG (pixels)  RATIO ZPTAVG ZPTSIG MAG
...
S: 56190.000  1000  0  1.00  10  100.00  2.00  0  0  1+2+3+4  0.020  99
S: 56190.000  1001  1  1.00  10  47.63  2.35  0  0  32.98  0.020  99
S: 56190.000  1002  2  1.00  10  98.63  2.35  0  0  32.33  0.020  99
S: 56190.000  1003  3  1.00  10  122.63  2.35  0  0  32.21  0.020  99
S: 56190.000  1004  4  1.00  10  147.63  2.35  0  0  32.16  0.020  99
...
```

The ZPTAVG value for filter-0 is assumed to be ambiguous because it depends on the magnitude of the object (see below), and hence this entry is replaced by 1+2+3+4 to indicate that its flux is a sum of filters that have well-determined properties. Filters 01234 must all be defined in the usual way in the kcor/calib file, and the GENFILTERS key must include these five filters. The MJDs for 012345 must be exactly the same; if not, the simulation will abort. The SIMLIB entries for 1234 must be accurately defined, and any reasonable values for '0' are sufficient since the filter-0 flux will get over-written with the sum of fluxes from 1+2+3+4. The zeropoint ( $Z_0$ ) for filter-0 is calculated to be

$$Z_0 = 2.5 \times \log_{10} \left[ \sum_i 10^{-0.4(m_i - M_0 - Z_i)} \right] \quad (2)$$

where  $m_i$  are the simulated magnitudes in filter components  $i = 1, 2, 3, 4$ ,  $Z_i$  are the user-determined zeropoints in  $i = 1, 2, 3, 4$ , and  $M_0$  is the simulated magnitude in filter-0. Note that for a coadd we have  $m_i = M_0$  and recover the usual expression for the co-added zeropoint.

## 4.11 Global Noise Corrections (Optional)

Global noise-terms  $\hat{\sigma}_0$  and  $\hat{S}_{\text{SNR}}$  (Eq. 1) can be specified in the simlib header. The  $\hat{\sigma}_0$  term can be used to account for additional noise from the SN-photometry.  $\hat{S}_{\text{SNR}}$  is a global correction as a function of the log of the signal-to-noise ratio,  $\log_{10}(\text{SNR})$ ; this correction may be useful for PSF forms that are highly non-Gaussian (e.g., in space), or as a global correction so that the simulated uncertainties better match the those from the data.

```

FLUXERR_ADD:  YJH  33  55 85   # sig_0 term for each filter

# Below is the S_SNR map,
#           FILTs  LOG10(SNR)   ERROR/ERROR(SNANA)
FLUXERR_COR:  YJH   -5.00     1.0000  1.0000  1.0000
FLUXERR_COR:  YJH   -4.80     1.0000  1.0000  1.0000
FLUXERR_COR:  YJH   -4.60     1.0000  1.0000  1.0000
...
FLUXERR_COR:  YJH    0.40     1.1387  1.1719  1.1775
FLUXERR_COR:  YJH    0.60     1.1368  1.1704  1.1751
FLUXERR_COR:  YJH    0.80     1.1295  1.1611  1.1654
FLUXERR_COR:  YJH    1.00     1.1189  1.1470  1.1512

BEGIN LIBGEN

```

For SNR values outside the map-range, the correction at the min or max edge of the map is used. Thus for the above map,  $\text{SNR} > 10$  results in corrections corresponding to the last FLUXERR\_COR row.

To determine  $\hat{\sigma}_0$ , plot the calibrated flux-uncertainty “FLUXCAL\_ERRTOT” (or FLUXERR in ntuple 7799) for both data and simulation; adding the best-fit  $\hat{\sigma}_0$  in quadrature to the simulated uncertainty should match the measured distribution. It is recommended to check the data-simulation comparison in PSF bins. There is a utility in the SNANA fitting program that calculates the noise analytically, allowing a more direct comparison of the true uncertainty to the uncertainty that would be computed in a simulation. This feature is automatically enabled for verbose-formatted data that includes the SKY, PSF and ZPTAVG for each observation. See the SDSS data as an example of the verbose format. The quantity “ERRTEST,” the ratio of calculated-to-true uncertainty, is included in ntuple 7799 for each observation.

To determine  $\hat{S}_{\text{SNR}}$ , plot ERRTEST vs.  $\log_{10}(\text{SNR})$  and the construct the FLUXERR\_COR map from a polynomial fit or spline fit.

Finally, note that there are no SNANA utilities to construct these maps.

## 4.12 Noise Calculation

Here is an analytic calculation of the noise from the signal and sky-background. The error on the signal (in photoelectrons) is simply  $\sqrt{N_{\text{pe}}}$ . To get the error in observed CCD counts (ADU), we start by relating the signal counts in ADU to the number of photoelectrons,

$$\mathcal{N}_{\text{ADU}} = N_{\text{pe}} \times G^{-1} \times S_{ZP} \quad (3)$$

where  $N_{\text{pe}}$  is the number of observed photoelectrons,  $G$  is the number of photoelectrons per ADU (CCD gain), and  $S_{ZP}$  is a scale factor applied to the signal so that the SN magnitude is referenced to the template zeropoint. This factor is  $S_{ZP} = 10^{0.4(ZP_t - ZP_s)}$ , where  $ZP_{s,t}$  are the zeropoints for the search and template runs. In cloudy conditions,  $S_{ZP} \gg 1$  because the signal is much smaller than it would have been in the template run. If no template is given, then  $S_{ZP} = 1$ . The error on the signal (in ADU) is

$$\sigma^2(\mathcal{N}_{\text{ADU}}) = \mathcal{N}_{\text{ADU}} \times G^{-1} \times S_{ZP} \quad (4)$$

The sky-background error is computed from

$$\sigma_{\text{skytot}} = S_{ZP} \times \sqrt{A \times (\sigma_{\text{skypix}}^2 + \bar{\sigma}_{\text{skypix}}^2)} \quad (5)$$

where  $\sigma_{\text{skypix}}$  is the search-run skynoise per pixel,  $\bar{\sigma}_{\text{skypix}}$  is the template-run skynoise,  $A$  is the noise-equivalent area in square-pixels, and the units are ADU. There is a similar term for the CCD readout noise per pixel (summed over the area), but it is left out here in this example.

Using double-Gaussian fit parameters for the PSF, in which  $\sigma_{1,2}$  are the PSF-sigma for the two Gaussians, and  $h_{1,2}$  are the heights at the origin<sup>3</sup>, the noise-equivalent area is

$$A = \frac{1}{\int PSF^2(r, \theta) r dr d\theta} = \frac{4\pi(\sigma_1^2 + \sigma_2^2)}{1 + 4\pi^2\sigma_1^2\sigma_2^2(h_1 + h_2)^2} = 4\pi\sigma_1^2 \left[ \frac{(1 + R_\sigma^2)(1 + R_\sigma^2 r_h)^2}{R_\sigma^2(1 + r_h)^2 + (1 + R_\sigma^2 r_h)^2} \right], \quad (6)$$

where in the second step  $R_\sigma \equiv \sigma_2/\sigma_1$  and  $r_h \equiv h_2/h_1$ . For a single-Gaussian PSF,  $r_h \rightarrow 0$  for any value of  $R_\sigma$ , and the area is just  $4\pi\sigma^2$ . For typical ground-based surveys  $A/(4\pi\sigma_1^2) \sim 1.5$ .

The rest of this section will perform an explicit calculation of the noise, using an example from a DES simulation. Here is the information for a random epoch on a random simulated lightcurve:

---

<sup>3</sup>A double-Gaussian PSF with normalization  $\int PSF(r, \theta) r dr d\theta = 1$  has  $2\pi(\sigma_1^2 h_1 + \sigma_2^2 h_2) = 1$

PASSBAND:	g	r	i	z	Y	
GAIN:	1.000	1.000	1.000	1.000	1.000	e/ADU
RDNOISE:	10.000	10.000	10.000	10.000	10.000	e-
SKY_SIG:	108.81	105.36	217.58	378.84	848.53	ADU
PSF_SIG1:	1.500	1.500	1.500	1.500	1.500	pixels
FLUX:	14707.89	12515.67	30756.55	28334.23	62748.97	ADU
FLUX_ERRTOT:	590.695	571.406	1170.485	2022.083	4518.783	ADU
FLUXCAL:	18.52	42.21	46.13	46.59	51.71	(x10 <sup>11</sup> )
FLUXCAL_ERRTOT:	0.986	2.403	2.385	3.683	4.141	
MAG:	24.3311	23.4364	23.3402	23.3292	23.2160	
MAG_ERRPLUS:	0.0569	0.0624	0.0565	0.0898	0.0892	
MAG_ERRMINUS:	0.0569	0.0624	0.0565	0.0898	0.0892	
ZEROPT:	34.7500	33.6800	34.5600	34.4600	35.2100	
ZEROPT_SIG:	0.0350	0.0340	0.0350	0.0340	0.0350	

For simplicity, note that the PSF is modeled as a single-Gaussian, and that the gain is unity. Now let's compute that flux and noise for *i*-band.

The measured flux (in ADU) and the calibrated flux (for lightcurve fits) are given in terms of the magnitude ( $m_i$ ) and zeropoint ( $Z_i$ ),

$$\text{FLUX} = 10^{-0.4(m_i - Z_i)} = 10^{-0.4(23.3402 - 34.56)} = 30755 \text{ ADU} \quad (7)$$

$$\text{FLUXCAL} = 10^{-0.4m_i} \times 10^{11} = 46.13 \quad (8)$$

which agrees with the simulated values above. Since the gain is unity, 1 ADU = 1 photoelectron (p.e.), and the noise from signal-photostatistics is  $\sigma_{sig} = \sqrt{N_{pe}} = 175.4 \text{ p.e.}$ .

To include the sky-noise, we use the following information from the simulation library (see above dump): SKYSIG = 217.58 ADU/pixel,  $\text{PSF}(\sigma) = 1.50\sqrt{\text{pixels}}$ , and 1 pixel is  $0.27'' \times 0.27''$ . The effective aperture area is  $A = 4\pi \times \text{PSF}^2 = 28.27 \text{ pixels}$ . The total skynoise is thus  $\sigma_{sky} = \text{SKYSIG} \times \sqrt{A} = 1156.95 \text{ p.e.}$ . The total noise on the flux is  $\text{FLUX\_ERRTOT} = \sqrt{\sigma_{sig}^2 + \sigma_{sky}^2} = \sqrt{1156.94^2 + 175.4^2} = 1170.2 \text{ p.e.} = 1170.2 \text{ ADU}$ . The signal-to-noise ratio (SNR) is  $30755/1170 \simeq 26$ .

## 4.13 K-corrections

For the stretch and MLCS2k2 models, K-corrections are needed in both the simulation and the fitter. K-corrections are applied using a technique very similar to that used in [8, 1]. The basic idea is that for each epoch and each pass-band, the spectral template is warped by applying the CCM89 extinction law with a variable  $A_V$ ; “ $A_V$ -warp” is the value of  $A_V$  for which the synthetic color matches the color of the rest-frame lightcurve. The K-correction is then determined from this  $A_V$ -warped spectral template. Note that this method is model-independent and can therefore be applied to any lightcurve model.

The K-corrections and synthetic magnitudes are NOT computed internally because this would result in slow code, especially for the fitter. To speed up the calculations of these convolution-integrals, K-corrections and synthetic mags are read from a lookup table generated with `SNANA_DIR/bin/kcor.exe`. Before running the simulation or fitter, the program `kcor.exe` must run, although it runs once and only once until you need K-corrections that are not defined. The `kcor` program reads a self-documented input file such as

```
$SNDATA_ROOT/analysis/sample_input_files/kcor/kcor_[SURVEY].input
```

and then generates lookup tables as a function of redshift, epoch, and extinction parameter  $A_V$ . A typical table binning is 0.05 in redshift, 1 day in rest-frame epoch, and 0.25 in  $A_V$ ; this binning is used to store every user-defined  $K_{xy}$ , and to store synthetic lightcurves for every user-defined filter.

A linear interpolation routine<sup>4</sup> determines a K-correction for arbitrary  $z, \text{epoch}, A_V$ . A special function, `GET_AVWARP`, finds the magic  $A_V$ -warp parameter such that the warped spectral template has the same color as your lightcurve. The table format is CERNLIB’s `HBOOK`, and is stored in

```
$SNDATA_ROOT/kcor .
```

The subroutines that read and interpolate the `kcor` tables are written in fortran, and are stored in `snana.car`. The `SNANA` product includes a fortran library `./lib/libsnana.a`, which allows C programs to use the fortran utilities. The `extern` statements at the top of `snlc_sim.c` declare the fortran functions used to lookup K-corrections.

---

<sup>4</sup>a double precision version of CERNLIB’s `FINT` is used for multi-dimensional linear interpolations.

## 4.14 Intrinsic Brightness Variations

There are six general methods to introduce intrinsic variations that result in anomalous scatter in the Hubble diagram:

```
# method 1: coherent variation in all epochs & passbands
#           note: colors are not varied.
GENMAG_SMEAR: 0.1 # coherent mag-smear in all measurements

# method 2: independent variation in each passband (coherent among epochs)
#           that results in color variations
GENMODEL_ERRSCALE: 1.1 # scale MLCS peak-model error
GENMAG_SMEAR_FILTER: UBV 0.05 # and/or fixed smearing per filter
GENMAG_SMEAR_FILTER: RI 0.08 # and/or fixed smearing per filter

# method 3: Pick model name from specialized code in genSmear_models.c
GENMAG_SMEAR_MODELNAME: G10 # options are G10, C11, PRIVATE, NONE

# method 4: vary RV with asymmetric Gaussian distribution
GENMEAN_RV: 1.6 # location of Gauss peak
GENSIGMA_RV: 0.1 0.9 # lower, upper Gaussian-sigmas
GENRANGE_RV: 1.3 4.5 # gen-range for RV

# method 5: apply intrinsic scatter matrix (SALT2 only)
COVMAT_SCATTER_SQRT[0][0]: 0.10 ! mB : sqrt(COV) = uncertainty
COVMAT_SCATTER_SQRT[1][1]: 0.22 ! x1 : sqrt(COV) = uncertainty
COVMAT_SCATTER_SQRT[2][2]: 0.05 ! color : sqrt(COV) = uncertainty
COVMAT_SCATTER_REDUCED[0][2]: 0.50 ! rho(mB,c) = COV/[sig(mB)*sig(c)]

# method 6: function of wavelegth (SALT2 only); see text for details
GENMAG_SMEAR_FUNPAR: <SIGCOH> <A5500> <LAMSEP> <LAMPHASE> <TAU_LAM> <TAU_DAY>
0. 0.
```

Note that methods 1&2 can be used together, as well as methods 1&3. However, methods 2&3 cannot be used together. Methods 5-6 (scatter matrix) cannot be combined with any other method.

For rest-frame models, the argument of `GENMAG_SMEAR_FILTER` should be a list of rest-frame filters; for observer-frame models, the argument is a list of observer-frame filters.

`GENMAG_SMEAR_MODELNAME` allows the most flexibility because this option allows for an arbitrarily complex function to describe the smearing as a function of wavelength. These functions are in a separately compiled module (`genSmear_models.c`) so that updates are relatively easy and so that these functions can in principle be used in non-SNANA applications. The models include a “PRIVATE” option so that anyone can quickly implement a model of intrinsic smearing by adding code to the blank functions `init_genSmear_private` and `get_genSmear_private`. The “NONE” option can be used

as a command-line override to turn off this option. While these smearing models are functions of rest-frame wavelength, there are two implementation modes for the SNIa models. The mode is controlled by `FLAG_GENSMEAR` that is internally initialized in the simulation. The first mode is to properly include the wavelength dependence, and this mode is currently set only for the SALT-II model. The second mode is an approximation in which the smearing value at  $\tilde{\lambda}_{\text{obs}}/(1+z)$  is applied to the magnitude of the entire passband, where  $\tilde{\lambda}_{\text{obs}}$  is the central wavelength of the passband. This mode is set for rest-frame models (i.e., `SNooPY`, `MLCS2k2`). If/when the wavelength-dependent smearing is upgraded to work for rest-frame models, `FLAG_GENSMEAR` will be modified accordingly.

Each element of the intrinsic scatter matrix (method 5) can be entered as a covariance, as an uncertainty, or as a reduced covariance. It is recommended to enter uncertainties for the diagonal elements, and to enter reduced covariances ( $-1$  to  $+1$ ) for the off-diagonal terms. This model-smearing option currently works only for SALT2, but can easily be extended to other models as needed.

The function for method 6 is as follows. `SIGCOH` is a coherent scatter term that is added independent of the other parameters. The wavelength-dependent part is first defined at  $\lambda$ -nodes separated by `LAMSEP` Å with an initial phase of `LAMPHASE` Å. The  $1\sigma$  scatter magnitude ( $\sigma_{\text{mag}}$ ) at each  $\lambda$ -node ( $\lambda_n$ ) is

$$\sigma_{\text{mag}} = A_{5500} \times \exp[-(\lambda_n - 5500)/\tau_\lambda] \times \exp[T_{\text{rest}}/\tau_{\text{day}}] \quad (9)$$

where  $\tau_\lambda = \text{TAU\_LAM}$  and  $\tau_{\text{day}} = \text{TAU\_DAY}$ . The last two zeros (7th and 8th params) are reserved for future upgrades. After a Gaussian random scatter at each node is selected, a continuous function of  $\lambda$  is made by connecting the nodes with cosine functions that ensure that the derivative is zero at each node.

## 4.15 Search Efficiency

The simulation includes options to model the search efficiency of the survey. The search efficiency is broken into two parts: (i) image subtraction pipelines and (ii) human efficiency that includes visual scanning and spectroscopic targeting and selection. Some explicit examples of sim-input options will be given at the end of this section. To ensure that your settings are correct, it is recommended to always check that the simulated redshift distribution matches that of the data.

All of the default efficiency files are stored in the directory `$SNDDATA_ROOT/models/searcheff`, and the file-name includes the name of the survey. However, you can specify an efficiency file in your private directory.

The image subtraction pipeline efficiency,  $\epsilon_{subtr}$ , is based on software algorithms and therefore in principle this part of the efficiency can be rigorously determined. The simplest specification of the search efficiency is based on requiring a minimum number of observations with the sim-input keyword “MINOBS\_SEARCH: <MINOBS>”. The simulation parametrizes  $\epsilon_{subtr}$  as a function of signal-to-noise (SNR) or magnitude in each filter. The motivation is that fake SNe overlaid onto images during the survey can be used to measure the efficiency curves. Examples of each type of efficiency curve (vs. SNR and vs. MAG) are in

```
$SNDDATA_ROOT/models/searcheff/SEARCHEFF_PIPELINE_SDSS.DAT (vs SNR)
$SNDDATA_ROOT/models/searcheff/SEARCHEFF_PIPELINE_HST.DAT (vs. MAG)
```

and one can add more files corresponding to different surveys. If no SEARCHEFF file is found, then  $\epsilon_{subtr} = 1$ . Warning: do NOT define both the SNR-based and MAG-based efficiency for a survey. For ground-based surveys we recommend using the SNR-based efficiency to properly account for variations in observing conditions. The sim-input keyword “SEARCHEFF\_PIPELINE\_FILE:” can be used to specify any file with efficiency curves, and this option is useful for testing and for efficiency curves that vary with time during a survey. The simulation always checks first if a file exists in your private directory; if not there then the above directory is checked.

The above SEARCHEFF information tells us if a given epoch is detected in a particular filter, but does not specify if the supernova would have been discovered. The *discovery* logic is defined for each survey in the file:

```
more $SNDDATA_ROOT/models/searcheff/SEARCHEFF_PIPELINE_LOGIC.DAT
SDSS: 3 gr+ri+gi # require 3 epochs, each with detection in two bands.
HST: 1 6 # require 1 epoch with detection in filter '6' = F850LP_ACS
```

where the first number is the minimum number of epochs required, and the second string defines the logic for a single-epoch detection. In the above examples, the SDSS discovery logic requires three epochs, where each epoch has a detection in at least two of the three *gri* filters. The HST discovery logic requires a single detection in filter ‘6’. A detection is defined by the SEARCHEFF\_PIPELINE\_[SURVEY].DAT files described above. The image-subtraction efficiency can be applied in the simulation, or the results can be stored in the data files for future analysis: the control flag APPLY\_SEARCHEFF\_OPT is discussed below after the spectroscopic efficiency is discussed.

The human/spectroscopic efficiency ( $\epsilon_{spec}$ ) cannot be rigorously computed since it involves human decision making during the survey. The SNANA simulation parametrizes  $\epsilon_{spec}$  as an arbitrary function of redshift and peak-magnitudes in each passband, and §4.15.1 suggests how to determine this function from the data and simulations. Rather than using an analytical form for  $\epsilon_{spec}$ , the simulated efficiency is defined on a multi-dimensional grid of redshift, peak-magnitudes and peak-colors. An example is shown here,

```
$SNDATA_ROOT/models/searcheff/SEARCHEFF_SPEC_SDSS.DAT
```

illustrating both the format and the default location. If this file does not exist for a particular survey (i.e, “SDSS” replaced by your survey in the filename) then  $\epsilon_{spec} = 1$ . For each simulated SN,  $\epsilon_{spec}$  is determined from multi-dimensional (linear) interpolation. If more than one grid is given the logical OR among the efficiencies is used; this feature may be useful in cases where different telescopes take spectra for SNe in different magnitude ranges. It is important to note that  $\epsilon_{spec} = 0$  outside the redshift/magnitude range given by the grid. This feature allows using different telescopes to cover different magnitude ranges, but be careful that very bright SNe can have  $\epsilon_{spec} = 0$  if the magnitude range is not wide enough at the bright end. To avoid mistakenly losing very bright SNe, we recommend adding an artificial grid with 100% efficiency for low redshifts; this is the first grid-map in Fig. 3.

Fig. 3 below illustrates a spectroscopic-efficiency grid with three tables. The first table ensures 100% efficiency for redshifts  $z < 0.1$ . The second table describes the efficiency for  $r$ -band magnitudes below 22 and the last table defines the efficiency for  $i$  band magnitudes above 22. The second map depends on the absolute peak  $r$ -band magnitude and the  $g - r$  color at peak. The 3rd map depends on the  $i$ -band magnitude and redshift, and applies only for the field named ‘DEEPFIELD’: the ‘FIELD:’ option allows for field-dependent maps. There essentially is no limit to the complexity: for example, one could define “NVAR: 7” and “VARNAMES: g r i g-r r-i REDSHIFT SPECEFF” assuming that such a function could be determined. A list of valid VARNAMES is shown in Fig. 4.

In principle there is just one function of redshift and peak-magnitudes to describe the spectroscopic efficiency. In practice, however, this efficiency can depend on the particular SN model for two reasons. First is an overall magnitude offset between different SN models. The second reason is that asymmetric (dim-side) tails in the stretch and color distributions may not be properly modeled. To correct for the first case there is a sim-input parameter “MAGSHIFT\_SPECEFF: <shift>” to shift the peak magnitude used to determine  $\epsilon_{spec} = 0$  from the grid-map. The second problem can be fixed only by using distributions with appropriate tails.

For each simulated SN Ia, the search algorithm is evaluated separately for the image-subtraction and spectroscopic efficiencies. Random numbers are compared against the efficiencies to determine which epochs/SNe are selected. The results of these two search algorithms are stored in a bit-mask in each data file, with the following possibilities:

SIM_SEARCHEFF_MASK: 1	# detected by image-subtr; failed spec follow-up
SIM_SEARCHEFF_MASK: 2	# failed image-subtr; passed spec follow-up
SIM_SEARCHEFF_MASK: 3	# detected by both image-subtr & spec follow-up

Note that `SIM_SEARCHEFF_MASK=2` corresponds to an unphysical case since it is unlikely to get a spectrum of a SN that was not identified by the image-subtraction pipeline. Although the search efficiencies are always evaluated, the user has the option to apply these efficiencies in the simulation, or to write out all simulated SNe and use the `SIM_SEARCHEFF_MASK` for further investigation. The following sim-input keys control trigger selection:

```

APPLY_SEARCHEFF_OPT: 0 # keep all SNe (default)
APPLY_SEARCHEFF_OPT: 1 # keep SN if software trigger passes
APPLY_SEARCHEFF_OPT: 3 # keep SN if software and spec triggers pass

```

When “`APPLY_SEARCHEFF_OPT: 3`” is set, then `SIM_SEARCHEFF_MASK=3` for all SNe because those that fail either of the search criteria are rejected. Setting “`APPLY_SEARCHEFF_OPT: 1`” results in SNe with `SIM_SEARCHEFF_MASK=1` and 3; i.e., SNe with and without spectroscopic confirmation. For the unconfirmed SNe, i.e., those with `SIM_SEARCHEFF_MASK=1`, the default redshift is assumed to have the same precision as for the confirmed SNe, presumably from a host-galaxy spectroscopic redshift. However, a redshift-dependent fraction of unconfirmed spectroscopic (host-galaxy) redshifts can be specified with

```

UNCONFIRMED_SPECZ_FRACPOLY: P0 P1 P2 ,

```

resulting in an unconfirmed spec-Z probability of  $P_0 + z \times P_1 + z^2 \times P_2$ . The default values are  $P_1=1$  and  $P_2=P_3=0$ . Note that this option can be used even if there is no host-galaxy simulation (§4.18). For unconfirmed SNe that have no redshift information, the redshift error is set to  $-9$ ; this value flags the photo-z fitting option `OPT_PHOTOZ=2` (§5.10) to ignore any redshift-prior information.

To keep track of the trigger selection in the analysis, the search-efficiency mask “**SIMEFMSK**” can be added as a `SIMGEN_DUMP` variable (§4.25.3) and it appears in the analysis ntuples (§5.23).

Figure 3: Illustration of spectroscopic efficiency format for the SNANA simulation.

```

# TABLE 1 -- ensure 100% eff at low redshift
NVAR: 2
VARNAMES: REDSHIFT SPECEFF
SPECEFF: 0.0 1.00
SPECEFF: 0.1 1.00

# TABLE 2 -- 4m telescope
NVAR: 3
VARNAMES: g-r r SPECEFF # can add comments here
SPECEFF: -0.5 18.0 1.0
SPECEFF: -0.5 20.0 0.6
SPECEFF: -0.5 22.0 0.3
SPECEFF: +0.5 18.0 0.2
SPECEFF: +0.5 20.0 0.08
SPECEFF: +0.5 22.0 0.02

# TABLE 3 -- 8m telescope
NVAR: 3
VARNAMES: REDSHIFT i SPECEFF
FIELD: DEEPFIELD
SPECEFF: 0.0 22.0 1.0
SPECEFF: 0.0 24.0 0.66
SPECEFF: 0.0 26.0 0.31 # can add comments here
SPECEFF: 0.5 22.0 0.22
SPECEFF: 0.5 24.0 0.14
SPECEFF: 0.5 26.0 0.022

```

Figure 4: Valid VARNAMES to describe  $\epsilon_{spec}$ .

- \* any filter; i.e, g r i
- \* any color; i.e, g-r g-i
- \* REDSHIFT
- \* PEAKMJD
- \* DTPEAK (closest T-Tpeak; can be pos or negative)

Below are a few examples of sim-input settings. First, to simulate a spectroscopically confirmed sample using private search-efficiency files,

```

APPLY_SEARCHEFF_OPT: 3          # apply all search efficiencies
SEARCHEFF_PIPELINE_FILE: TESTEFF_PIPELINE.DAT # private file
SEARCHEFF_SPEC_FILE:    TESTEFF_SPEC.DAT     # private file

```

To simulate a mix of confirmed and unconfirmed SN Ia, and to calculate  $\epsilon_{spec}$  using peak-magnitudes shifted by 0.1 mag,

```

APPLY_SEARCHEFF_OPT: 1          # software trigger only
UNCONFIRMED_SPECZ_FRACPOLY: 1 0 -0.5 # PROB(specZ) = 1 - z^2/2
MAGSHIFT_SPECEFF: 0.1         # applied to SPEC-EFF calc only

```

Lastly we recommend putting measured efficiency files in the public area `$SNDATA_ROOT/models/searcheff`.

#### 4.15.1 Determining $\epsilon_{spec}$

Since there is no rigorous method to determine  $\epsilon_{spec}$ , one must essentially start with a guess at the functional form and fit for parameters such as an exponential slope or power law. The spectroscopic efficiency must have the form  $\epsilon_{spec}(z, \vec{m}, \vec{E})$ , where  $z$  is the redshift,  $\vec{m}$  are the peak magnitudes and  $\vec{E}$  are efficiency-function parameters to be determined. Next generate a large simulated sample that includes the pipeline efficiency (i.e, `APPLY_SEARCHEFF_OPT: 1`) and the same selection requirements that are applied to the data. The last step is to fit for  $\vec{E}$  by minimizing

$$\chi^2 = \sum_i [(N_{DATA}^i - N_{SIM}^i \times E_0 \epsilon_{spec}(z, \vec{m}, \vec{E})) / \sigma_i^2] \quad (10)$$

where  $E_0$  is an overall scale such that the integrated data and simulation have the same statistics,  $i$  is an index over bins, and  $\sigma_i$  is the statistical uncertainty on the data. The bins can be simply redshift bins, or multi-dimensional bins of redshift and the peak magnitude(s). We suggest starting with the simple case of redshift bins, and trying more complex binning only if the simple case is not adequate. After determining  $\vec{E}$  from the fit, it is important to check data-simulation comparisons on distributions of other quantities, namely the fitted stretch and color parameters.

### 4.15.2 Legacy Spectroscopic Efficiency Options

Here we discuss some older (legacy) features to analytically calculate the spectroscopic efficiency. We recommend switching away from these methods and using the efficiency grid-map that has no analytical limitations.

#### SPECTYPE:

```
# EFF = EFF0 * (1-x^EFFEXP), x=(PKMAG-MAGMIN)/(MAGMAX-MAXMIN)
```

	FILT	EFF0	MAGMIN	MAXMAG	EFFEXP	ZERR
SPECTYPE:	r	0.40	16.0	21.5	5	0.005
SPECTYPE:	i	0.40	21.5	23.5	6	0.005

gives the magnitude ranges and efficiency function ( $\epsilon_{spec} = \epsilon_0(1 - x^n)$ ) for each filter. The factors  $1 - x^{5,6}$  cause a sharp efficiency roll-off near the magnitude limit; by definition,  $\epsilon_{spec} = 100\%$  at MAGMIN and zero at MAGMAX. For a given filter, a spectroscopic type is given (i.e., SNTYPE) if the calculated  $\epsilon_{spec}$  is larger than a random number between 0 and 1. The logical-OR is taken when multiple filters are specified. For a spectroscopically typed SN, the measured (i.e., smeared) REDSHIFT\_FINAL is determined from ZERR above. For un-typed SNe, SNTYPE is set to -999. Unless the BLINDING option is selected (§4.24.4), the SIM\_NON1a index always appears regardless of whether SNTYPE is set.

As described in the previous section, the key UNCONFIRMED\_SPECZ\_FRACPOLY can be used to determine the fraction of unconfirmed SNe that have a precise spectroscopic (host-galaxy) redshift.

**HUMAN\_SEARCH\_OPT:** The simulation allows for  $\epsilon_{spec}$  to be defined as an exponential function of redshift or “magdim” ( $\mathcal{M}_{dim}$ ), where  $\mathcal{M}_{dim}$  is the relative intrinsic brightness of a SN Ia. For MLCS2k2,  $\mathcal{M}_{dim}$  depends on  $\Delta$  and  $A_V$ : for SALT-II,  $\mathcal{M}_{dim}$  depends on  $x_1$  and  $c$ : The exponential parameters for all surveys are defined in the file HUMAN\_SEARCHEFF\_V3.DAT in the same directory as the other efficiency files. The option is specified in the sim-input file with “HUMAN\_SEARCHEFF\_OPT: <opt>”

## 4.16 Selection Cuts

Although selection cuts are usually applied with the fitting program (§ 5) or some external program, the simulation allows for some basic selection cuts. This feature is particularly useful, for example, to simplify and speed up the generation of a large efficiency grid, and to estimate rates. The global flag to implement the “CUTWIN\_XXXX” selection criteria is

```
APPLY_CUTWIN_OPT: 0 # => ignore selection cuts (default)
APPLY_CUTWIN_OPT: 1 # => implement selection cuts
APPLY_CUTWIN_OPT: 3 # => apply cuts to data files; NOT to SIMGEN_DUMP
```

For option 1 the selection cuts are applied to both the data files and to the entries in the SIMGEN\_DUMP file (§4.25.3). The last option (3) is useful for keeping track of absolute efficiencies. In this case, “CUTMASK” should be include among the SIMGEN\_DUMP variable list to tag which SNe have data files written out.

A list of available cut-commands are as follows:

```
EPCUTWIN_LAMREST: 3000 9500 # cut-window for <lamobs>/(1+z)
EPCUTWIN_SNRMIN: +3 1E8 # min SNR for each observation
CUTWIN_TRESTMIN: -19 -5 # at least 1 epoch before -5 d (rest-frame)
CUTWIN_TRESTMAX: +30 +80 # at least 1 epoch past +30 d
CUTWIN_TGAPMAX: 0 20 # largest Trest gap (days)
CUTWIN_TOGAPMAX: 0 10 # largest Trest gap near peak (days)
CUTWIN_NOBSDIF: 6 999 # Number of obs passing MJDDIF cut
CUTWIN_MJDDIF: 0.4 999 # NOBSDIF++ if this much later than last MJD
CUTWIN_NEPOCH: 7 +5 # require 7 epochs with SNR>5
CUTWIN_SNRMAX: 10 griz 1 -20 60 # SNR>10 for at least 1 of griz filters
CUTWIN_SNRMAX: 5 griz 3 -20 60 # SNR>5 for at least 3 of griz filters
CUTWIN_SNRMAX: 5 griz 3 0 60 # idem, but after max
CUTWIN_SNRMAX: 5 ri 2 -20 60 # r & i must each have SNR > 5
CUTWIN_MWEBV: 0.0 0.1 # Galactic E(B-V)
CUTWIN_PEAKMAG: 10 27 # any filter-peakmag between 10 and 27
```

Each cut-command can be specified in the sim-input file, or using the command-line override (§12.2.1) without the colon. Any CUTWIN\_XXX that is not specified results in no cut. The cuts beginning with EPCUTWIN apply to every epoch (observation), and only measurements passing these EPCUTWIN\_XXX requirements are used to evaluate cuts on global light curve properties. CUTWIN\_NEPOCH includes its own SNR requirement; thus you could set “EPCUTWIN\_SNRMIN: -5 1E8” so that all measurements, regardless of SNR, are used for cuts on TRESTMIN, TRESTMAX and TGAPMAX, while still requiring 7 observations to have SNR > 5.

Multiple CUTWIN\_SNRMAX requirements can be specified. Note that this cut requires a certain number of passbands to have a minimum SNR value, but does not specify which bands. For the example above, “CUTWIN\_SNRMAX: 5 griz 3 -20 60” is satisfied if the maximum SNR is > 5 for either *gri*,

*grz*, *giz*, or *riz*. You can require SNR cuts for specific filters as illustrated above with “CUTWIN\_SNRMAX: 5 ri 2 -20 60”; this requires both *r* and *i* to have a measurement with  $\text{SNR} > 5$ .

The TGAPMAX requirement applies to observations between the lower-TRESTMIN and upper-TRESTMAX cuts; i.e.,  $-19$  to  $+80$  days in the example above. The TOGAPMAX requirement applies to observations near peak, meaning that the gap must overlap the range between the upper-TRESTMIN and lower-TRESTMAX cuts; i.e.,  $-5$  to  $+30$  days. In this example, a gap defined by  $-12$  to  $-6$  days is included in the evaluation of the TGAPMAX cut, but not in the TOGAPMAX cut. Gaps of  $-6$  to  $+2$  and  $+20$  to  $+35$  days are included in the evaluation of both cuts. A gap of  $+8.0$  to  $+85$  is ignored for both cuts.

The generation and cut-selection statistics are printed at the end of the sim-README file (§ 4.2). Here is an example when selection cuts are applied, while the search efficiency is not:

```
Generation Statistics:
  Generated   250 simulated light curves.
  Wrote       100 simulated light curves to SNDATA files.
Rejection Statistics:
  1 rejected by GEN-RANGE cuts.
  0 rejected by SEARCH-TRIGGER
 149 rejected by CUTWIN-SELECTION
SEARCH+CUTS Efficiency: 0.402 +- 0.031
```

An efficiency grid can be quickly computed by looping over the variables of interest (redshift, SN brightness, etc ) and using `grep "SEARCH+CUTS"` to extract the efficiencies.

The number of generated events is specified by the keyword `"NGEN_LC: 100"`, which instructs the simulation to generate 100 lightcurves that pass the `SEARCH-TRIGGER & CUTWIN-SELECTION`. To prevent an infinite loop when the efficiency is (accidentally) zero, specify

```
EFFERR_STOPGEN: 0.001 # stop sim when effic error is this small
```

so that the simulation will stop gracefully after generating 1,000 lightcurves that all fail cuts. To avoid unwanted program exits, make sure that the `EFFERR_STOPGEN` value is much smaller than the expected efficiency.

## 4.17 Varying the Exposure Time/Aperture/Efficiency

For testing future (i.e, non-existent) surveys, the exposure times can be varied relative to that of the SIMLIB, and avoids the need to create a new SIMLIB for each sequence of exposure times. The sim-input syntax is

```
EXPOSURE_TIME:      2.0          # global increase for all filters
EXPOSURE_TIME_FILTER: g   3.0      # x3.0 more exposure in g-band
EXPOSURE_TIME_FILTER: r   4.6      # x4.6 more exposure in r-band
EXPOSURE_TIME_FILTER izY 8.0      # x8 more exposure in izY bands
```

Since the global EXPOSURE\_TIME multiplies the filter-dependent exposure times, the net exposure-time increase for the above example is 6 and 9.2 for *g* and *r*, respectively, and 16 for the *izY* filters. The default exposure-time increase is one.

This option is equivalent running each exposure longer by the specified amount, or increasing the aperture or efficiency. Technically, the simulation does the following for each filter:

```
ZPT(SIMLIB)  -> ZPT + 2.5*LOG10(EXPOSURE_TIME)
SKYSIG       -> SKYSIG * sqrt(EXPOSURE_TIME)
CCDNOISE     -> CCDNOISE * sqrt(EXPOSURE_TIME)
```

The changes in the ZPT and SKYSIG are unambiguous for a given EXPOSURE\_TIME, but the CCDNOISE should not change if the EXPOSURE\_TIME is assumed to increase the efficiency without increasing the number of times that the CCDs are read out. There is an option-mask to control which parameters to modify,

```
EXPOSURE_TIME_MSKOPT:  7 # bits 1,2,3 => ZPT, SKYSIG, CCDNOIST
```

The default is to modify all three SIMLIB parameters. To modify ZPT and SKYSIG while leaving the CCDNOISE fixed, set “EXPOSURE\_TIME\_MSKOPT: 3”

## 4.18 Simulating the Host Galaxy

Starting with `SNANA v9_30G` there is a host-galaxy package designed to simulate the following effects:

- select host galaxy based on arbitrary user-function of host properties.
- shift SN magnitudes (coherent for all epochs and wavelengths) based on arbitrary user-function of host properties.
- host-galaxy photo-Z to use as a photoZ fitting prior (§ 5.10)
- select random SN location (near host galaxy) weighted by surface brightness.
- host-galaxy contribution to [SN] Poisson noise at each epoch.

The host-galaxy information is stored in a library, hereafter denoted “HOSTLIB”. An example HOSTLIB is shown in Fig. 5. This self-documented file gives the number of variables describing each host (`NVAR`) and a list of descriptive “`VARNAMES`.” There are two mandatory variables: `GALID` and `ZTRUE`; the simulation will abort if either variable is missing. The integer `GALID` must be the first element, but `ZTRUE` can be anywhere on the `VARNAMES` list. The library need not be sorted by `ZTRUE` since the simulation internally sorts the library by redshift.

In principle a HOSTLIB needs to contain only `GALID` and `ZTRUE`, but such a library would not be useful to simulate interesting effects. The remaining `VARNAMES` in Fig. 5 are optional, and control what kinds of effects can be simulated. The `ZPHOT` and `ZPHOTERR` keys give the externally-computed photometric redshift, and “[`filt`]`_obs`” are the observer-frame galaxy magnitudes needed to compute the noise. The galaxy shape is described by an arbitrary sum of Sersic profiles. The galaxy size is defined by `VARNAMES`

```
a0_Sersic - a9_Sersic
b0_Sersic - b9_Sersic ;
```

these are major- and minor-axis half-light sizes (arcseconds) for up to 10 Sersic components. There are two options for defining the the Sersic index `n0_Sersic - n9_Sersic`. First, the Sersic index can be included as one of the `VARNAMES` as done with `n0_Sersic` in Fig. 5. This option allows a different Sersic index for each host galaxy. The second option is to define a fixed Sersic index after the `VARNAMES` list as done with `n4_Sersic` in Fig. 5; this same value is used for each host galaxy. Commonly used Sersic indices are  $n = 0.5, 1, 4$  for Gaussian, exponential and deVaucouleurs, respectively.

Finally, the weight “`w0_Sersic`” give the flux-ratio between this component and the total flux. The weight `w4_Sersic` need not be given since  $w_4 = 1 - w_0$ . Finally “`a_rot`” is the rotation angle of the major axis w.r.t. the RA coordinate.

The next set of keys in Fig. 5 describe the optional weight map. In this example, galaxies are weighted by the absolute r-band magnitude, and the SN brightness is also adjusted according to the absolute galaxy mag. The weight map can be a function of many `VARNAMES`. If no weight map is given the simulation will assign a weight of unity to each host galaxy. After the optional weight map, the `NVAR` parameters for each galaxy must follow a “`GAL:`” key.

Figure 5: Example HOSTLIB for the SNANA simulation.

```
NVAR: 25
VARNAMES: GALID RA DEC u_obs g_obs r_obs i_obs z_obs Y_obs
          u_ABS g_ABS r_ABS i_ABS z_ABS Y_ABS ZTRUE ZPHOT ZPHOTERR
          a0_Sersic b0_Sersic n0_Sersic w0_Sersic
          a4_Sersic b4_Sersic      a_rot

n4_Sersic: 4

NVAR_WGTMAP: 1
VARNAMES_WGTMAP: R_ABS
#      R_ABS  WGT dSNMAG
WGT:  -25.0  1.2  -0.05
WGT:  -23.0  1.0  -0.05
WGT:  -21.0  0.8  -0.05
WGT:  -19.0  0.6  +0.05
WGT:  -17.0  0.4  +0.05
WGT:  -15.0  0.2  +0.05

GAL:  384519  -24.701300 -42.860699 99 21.265400 20.194201 19.764299 19.543
      19.419901 99 -18.6369 -19.3917 -19.7638 -19.9781 -20.0809 0.226236
      0.226236 0.595291 0.441795 241.220139
GAL:  387438  -24.703699 -42.855801 99 20.922899 20.2365 19.9093 19.820499
      19.744301 99 -18.811199 -19.264799 -19.553499 -19.6514 -19.7331 0.226407
      0.226407 0.610256 0.427152 194.591675
GAL:  389400  -24.667601 -42.896099 99 20.3599 19.2935 18.802500 18.4839
      18.329599 99 -19.5123 -20.3393 -20.7794 -21.0847 -21.222099 0.226327
      0.226327 0.800331 0.694809 241.607605
GAL:  392965  -24.750099 -42.897598 99 21.9799 21.083 20.694300 20.508499
      20.399099 99 -17.836000 -18.4713 -18.815201 -18.997801 -19.0954 0.226194
      0.226194 0.844543 0.694027 202.914688
etc ...
```

The host-galaxy selection starts by finding a “near- $z$ ” subset of host galaxies within  $\pm 0.01$  of the SN redshift. A random host among this near- $z$  subset is picked based on the weight map. A HOSTLIB should have adequate statistics to densely cover the redshift range and parameter space used by the weight map.

The following HOSTLIB options can be set in the sim-input file (or via command-line override):

HOSTLIB_FILE:	DES.HOSTLIB	# required
HOSTLIB_WGTMAP_FILE:	TEST.WGTMAP	# optional (default=BLANK)
HOSTLIB_MAXREAD:	10000	# optional (default=billion)
HOSTLIB_MXINTFLUX_SNPOS:	.999	# optional (default=.99)
HOSTLIB_MSKOPT:	8	# optional (default=0)
HOSTLIB_GENRANGE_RA:	xxx yyy	# optional (default= -999 to +999)
HOSTLIB_GENRANGE_DECL:	xxx yyy	# optional (default= -999 to +999)
# debug options		
HOSTLIB_GALID_FORCE:	####	# optional forced GALID
HOSTLIB_FIXRAN_RADIUS:	#.##	# optional fix radial random number (0-1)
HOSTLIB_FIXRAN_PHI:	#.##	# optional fix phi random number (0-1)

Only the HOSTLIB\_FILE key is required, while the other keys are optional. Ideally all surveys would use the same HOSTLIB\_FILE, but in practice each survey will have its own HOSTLIB with a specific focus. Also note that there is no standard method for creating a HOSTLIB. The optional WGTMAP\_FILE overrides the default weight map embedded in the HOSTLIB file. For very large HOSTLIB files, the MAXREAD key may be useful to reduce the initialization time or limit memory usage. The next option (MXINTFLUX\_SNPOS) sets the fraction of total galaxy flux used to generate the SN position; this option truncates extreme galaxy-SN separations. Finally, The HOSTLIB\_MSKOPT options can be viewed with the grep-command

```
grep MSKOPT SNANA_DIR/src/snhost.c | grep #
```

- MSKOPT=2 : compute the host-galaxy noise within an aperture of radius  $2\sigma_{\text{PSF}}$ . This option requires defining HOSTLIB header keys “f\_obs” for each filter (see VARNAMES in Fig. 5), where ‘f’ is the one-character filter representation. These observation-frame magnitudes must be corrected for Galactic extinction and correspond to the entire galaxy flux.
- MSKOPT=4 : apply SN mag shift based on host properties.
- MSKOPT=8 : replace originally selected SN coordinates (i.e., randomly selected in field or from SIMLIB) by SN coordinates near the host galaxy (i.e., randomly selected from the surface brightness profile). This option is useful to generate SNe for image simulations.
- MSKOPT=16 : adjust the SN redshift, distance modulus and magnitudes to ZTRUE (useful for image simulations).
- MSKOPT=32 : allow only one SN per host galaxy (results in abort if library is too small)

- MSKOPT=256 : increase verbosity to screen.
- MSKOPT=1024 : detailed screen dump for each selected host.

Options can be combined, such as HOSTLIB\_MSKOPT=10 will transfer the SN redshift and coordinates to that of the galaxy, and compute the galaxy noise.

### Notes on CPU Resources:

The CPU generation time per host galaxy is dominated by the noise calculation that includes a convolution of the galaxy flux within a separate PSF-aperture for each epoch. The generation time is about 3 msec per host for a single Sersic profile, and 5 msec for a sum of 2 Sersic profiles. The main tool to minimize the generation time is to pre-compute integral tables for 2-dimensional Gaussians, and for Sersic profiles. Defining a reduced radius  $\rho \equiv R/R_{1/2}$  in terms of the half-light radius  $R_{1/2}$ , the dimensionless Sersic integrals

$$\mathcal{S}_n(\rho) = 2\pi \int_0^\rho e^{(-B_n x^{1/n} - 1)} x dx \quad (11)$$

are tabulated and stored on a uniform grid as a function of  $1/n$  ( $n = 0.3 - 5$ ) and as a function of  $\log_{10}(\rho)$  ( $\rho = 10^{-4}$  to 100). The  $B_n$  coefficients are chosen such that  $\mathcal{S}_n(1)/\mathcal{S}_n(\infty) = 1/2$ . The galaxy flux ( $F$ ) at local galaxy coordinates  $x_{\text{gal}}$  and  $y_{\text{gal}}$  is the sum over Sersic components,

$$F = \sum_n w_n \frac{e^{(-B_n \rho^{1/n} - 1)}}{a_n b_n \mathcal{S}_n(\infty)}$$

where  $w_n$  is the Sersic weight such that  $\sum_n w_n = 1$ ,  $a_n$  and  $b_n$  are the major and minor half-light axes, respectively, and  $\rho^2 = (x_{\text{gal}}/a_n)^2 + (y_{\text{gal}}/b_n)^2$  is the reduced radius.

### Ensuring Host PhotoZ in Fitting Program:

To ensure that the light-curve fitter cannot cheat when doing photoZ fits on simulated SNe, there is an option to replace the output REDSHIFT\_FINAL with the host-galaxy (photoZ) redshift so that the spectroscopic redshift is not available in the data file. This option is invoked by setting GENSIGMA\_REDSHIFT to any negative number. To go a step further and do SN-only photoZ fits (no host) without any risk of cheating, simply set GENSIGMA\_REDSHIFT to a large value like 1.0.

### HOSTLIB Variables for SIMGEN\_DUMP file:

The SIMGEN\_DUMP option is described in §4.25.3. Here is an example showing the HOSTLIB variables:

```
SIMGEN_DUMP: 7 CID Z GALZTRUE GALZPHOT GALSNDM GALWGT r_ABS
```

The GAL\* quantities can always be added, along with the subset of variables used to define the weight map.

## 4.19 Simulating the SN Rate: Volumetric and per Season

To simulate a constant volumetric rate at all redshifts, include the following in your sim-input file,

```
DNDZ: HUBBLE
```

and to simulate a redshift-dependent rate that depends on a power of  $1+z$ ,

```
DNDZ: POWERLAW 2.6E-5 1.5 # SN rate ~ 2.6E-5*(1+z)^1.5
```

Note that setting the second POWERLAW argument to zero is equivalent to the HUBBLE option of a constant rate. Finally, to simulate redshft-dependent power laws,

```
#           R0      Beta  Zmin Zmax
DNDZ: POWERLAW2 2.2E-5 2.15 0.0 1.0 # rate = R0(1+z)^Beta
DNDZ: POWERLAW2 9.76E-5 0.0 1.0 2.0 # constant rate for z>1
```

where the last two entries give the min/max redshift range. As a convenience, the output README file includes a dump of the SN volumetric rate in redshift bins of 0.1 (grep “MODEL-RATE”).

Highly distorted redshift distribution for special tests can be obtained with

```
DNDZ: FLAT
      or
DNDZ_ZEXP_REWGT: -2.0          # DNDZ *= 1/z^2
      or
DNDZ_ZPOLY_REWGT: 1.0 -0.2 0.003 # DNDZ *= [1 - 0.2*z + 0.003*z^2]
```

The “FLAT” command results in a flat redshift distribution. The next two examples re-weight the distribution defined by the DNDZ key above. The example with “DNDZ\_ZEXP\_REWGT: -2” re-weights by  $z^{-2}$  and will give a roughly flat redshift distribution. The second option allows the user to multiply the “DNDZ” redshift distribution by an arbitrary 2nd-order polynomial function of the redshift.

As a convenience, the absolute number of SN per season within your survey ( $N_{\text{season}}$ ) is written into the output README file as follows:

```
Number of SN per season = 12345
```

This value does not depend on NGEN\_LC or NGENTOT\_LC,<sup>5</sup> and it is not used in the simulation. This calculated value depends on the MJD range (GENRANGE\_PEAKMJD), redshift range (GENRANGE\_REDSHIFT), DNDZ option above, and coordinate ranges (GENRANGE\_RA and GENRANGE\_DECL). The sky area specified by the RA and DECL ranges can be overwritten by explicitly defining a solid angle in your sim-input file using

```
SOLID_ANGLE: 0.0204 # solid angle (steridian) for SN/season estimate
```

---

<sup>5</sup>NGEN\_LC is the number of SNe generated after trigger cuts and NGENTOT\_LC is the total number generated regardless of the trigger and cuts.

The `SOLID_ANGLE` option is useful when the survey consists of several dis-connected patches of sky, thereby requiring the RA and DECL ranges to represent a solid angle that is much larger than that of the survey.  $N_{\text{season}}$  can be used generate an arbitrary number of SN seasons. For example, to simulate 3 seasons set the following:

```
NGENTOT_LC: 37035 # 3*12345 = 3 seasons
```

## 4.20 Simulating Galactic Extinction

The following sim-input parameters control MilkyWay Galactic extinction:

EXTINC_MILKYWAY:	1	# 0,1 => MW extinction off,on
GENSIGMA_MWEBV_RATIO:	0.16	# smear by sig(MWEBV) = .16*MWEBV
GENSIGMA_MWEBV:	0.0	# smear by fixed sig(MWEBV)
GENSHIFT_MWEBV	0.0	# fixed shift relative to dust map

To turn off Galactic extinction, the first two (default) parameters must be explicitly set to zero (the last parameters are zero by default). You can specify a fractional error (default is 16%) on the Galactic extinction, or a fixed (absolute) error. If the smearing for a given SN results in negative extinction, the extinction is set to zero. The value of  $E(B - V)$  is taken from the simlib “MWEBV: xxx” key. If this key is missing, or the entry value is zero, then MWEBV is calculated internally using the dust maps from [7]. For each simulated SN the value of MWEBV and its error are written to the header.

## 4.21 “Perfect” Simulations

To make detailed numerical crosschecks, there is a “perfect” option to simulate light curves with  $\times 10^4$  nominal photostatistics, no galactic extinction (Milky Way and host), and no intrinsic mag-smearing. The light curve fitter should determine the shape and color parameters with very high precision, and the cosmology fitter should determine cosmological parameters that agree well with the input. This option is invoked with

```
GENPERFECT: 1
```

and it automatically overrides the relevant parameters so that you need not change your sim-input file. The top of the sim-README file summarizes the modified quantities. You can also unselect some of the “PERFECT” options by specifying a bit-mask as the GENPERFECT argument. To see the bit-mask options,

```
snlc_sim.exe mysim.input GENPERFECT -1
```

will list the current bit-mask options and then quit without generating any SNe. You can then run, for example,

```
snlc_sim.exe mysim.input GENPERFECT 6 # = 2+4 (bits 1 & 2)
```

which selects the  $\times 10^4$  exposure-time option (bit 1) and turns off intrinsic mag-smearing (bit 2), but leaves Galactic and host-galaxy extinction as defined in your sim-input file.

## 4.22 Redshift-Dependent Parameters

Although the default simulation parameters are independent of redshift, you can specify an arbitrary  $z$ -dependence for SN-related parameters such as dust parameters  $R_V$  &  $\tau_V$ , SALT-II parameters  $\alpha$  &  $\beta$ , and the mean luminosity parameter for any model.

The  $z$ -dependence is specified as an additive shift. If the function is simple, you can specify a polynomial function up to 3rd order. For more complex functions you can specify the function explicitly in redshift bins. Your function must give a shift of zero at  $z = 0$  so that the sim-input parameters are clearly defined at  $z = 0$ . Examples for specifying both types of parameter-shift functions are given in this file,

```
$SNDATA_ROOT/analysis/sample_input_files/SALT2/SIM_ZVARIATION.PAR .
```

To get a complete list of parameters that can have a  $z$ -dependence, type

```
> snlc_sim.exe mysim.input ZVARIATION_FILE 0
```

Next, copy the `SIM_ZVARIATION.PAR` above to your working area, and modify as desired. Then add the following keyword to your sim-input file,

```
ZVARIATION_FILE: SIM_ZVARIATION.PAR
```

or use the command-line override (§12.2.1). You can also change the name of the “ZVARIATION” file.

## 4.23 Generating Efficiency Maps

An efficiency map as a function of SN parameters may be needed as part of a fitting prior, or as part of an MC-based correction such as correcting the SN rate for the selection efficiency. The simulation can be used to generate an arbitrary efficiency map using the command

```
SIMEFF_MAPGEN.pl <SIMEFF input file>
```

and examples of the `SIMEFF` input file are in

```
$SNDATA_ROOT/analysis/sample_input_files/simeff_mapgen/
```

“`sntools.c`” contains functions read the generated efficiency map (`init_SIMEFFMAP`) and to evaluate the efficiency for an arbitrary set of SN parameters (`get_SIMEFF`). The efficiency is determined by multi-dimensional interpolation. Since the generation of a multi-dimensional efficiency map can be CPU intensive, this script distributes jobs on several nodes defined by the `NODELIST` key, and the simulation runs in a mode where there are no output files, and hence no secondary `SNANA` jobs are needed. Your sim-input file (specified inside the `SIMEFF` input file) must apply selection cuts as described in § 4.16. It is assumed that the selection efficiency does not depend on the result of the light curve fit.

The critical part of the SIMEFF input file is shown below,

#		sim-input	key	out	key	NBIN	MIN	MAX	
#	-----								
GENVAR:	LIN	GENRANGE_MWEBV	MWEBV	2	0.0	0.3			
GENVAR:	LIN	GENRANGE_REDSHIFT	Z	23	0.05	1.15			
GENVAR:	LOG	GENRANGE_AV	AV	19	-3.0	0.6	(0.001 < AV < 4)		
GENVAR:	LIN	GENRANGE_DELTA	DELTA	14	-0.5	2.1			
GENVAR:	INV	GENRANGE_RV	RV	3	0.25	0.75	(4 > RV > 1.33)		

Each GENVAR key specifies one dimension of the multi-dimensional efficiency map. Following each GENVAR key is a key defining whether that variable is stored linearly (LIN), logarithmically-base10 (LOG), or as the inverse (INV). For example, the efficiency is quite linear as a function of  $1/R_V$  and hence fewer  $R_V$  bins are needed to describe the efficiency as a function of  $1/R_V$  compared to using  $R_V$ . The GENRANGE\_XXX key is the simulation key used to specify that particular parameter. For example, a specific value of DELTA is simulated using

```
snlc_sim.exe <sim-input file> GENRANGE_DELTA -0.1 -0.1
```

All of the GENRANGE\_XXX commands are catenated and given as input to the simulation. The output-key is the name given in the output efficiency-map file. Any output key-name is valid, but to use this map for a fitting prior the key-names must correspond to one of the following: (i) any fit-parameter name in snlc\_sim.exe such as AV, DELTA, x1, c, (ii) REDSHIFT or Z, (iii) MWEBV.

The last three entries are the number of bins to define the efficiency map in each dimension (NBIN), and the min/max range for each dimension. In the above example,  $1/R_V$  is generated for values 0.25, 0.50, and 0.75 corresponding to  $R_V = 4, 2, 1.33$ , respectively. When the resulting efficiency map is used as part of a fitting prior, fit-values outside the min/max range are pulled to the edge for evaluating the efficiency. For example, if the fitting program tries to evaluate the  $\chi^2$  for DELTA= 2.4, the efficiency is evaluated at the boundary DELTA= 2.1.

Finally, one must be careful allocating appropriate resources since the computing time can be long. In the above example the total number of bins in this efficiency map is  $2 \times 23 \times 19 \times 14 \times 3 = 36708$ . If the efficiency-uncertainty (see SIMGEN\_EFFERR key) is set so that each simulation job takes 10 seconds, then the total computing time needed for this map is 4.2 CPU-days, or about 10 wall-clock hours with 10 cores.

## 4.24 Light Curve Output Formats

Each simulated light curve is written to the directory

```
$SNDATA_ROOT/SIM/[GENVERSION]
```

where `GENVERSION` is the user-supplied version name. For the default FITS format two FITS files are created: a “HEADER” file containing global information for each SN (SNID, redshift, RA, etc ...) and a “PHOT” file containing all of the light curves. Pointers in the HEADER file are used to extract the appropriate light curve from the PHOT file. These files can be visually examined with the product “fv.” For text-output options each light curve is written to a separate file, `[GENVERSION]_SN#####.DAT`, where “#####” is a six digit identifier; the text-option is useful for testing small samples and debugging.

The output format is controlled by the sim-input keyword `FORMAT_MASK`, and the various options are described below. Note that `FORMAT_MASK` is a bit-mask so that multiple format options can be included. Note, however, that either TEXT or FITS format can be selected, but not both. Here is a quick summary of the bit-mask format options,

```
FORMAT_MASK:  2  # 2 = terse, plain text (default for version < v9_82)
FORMAT_MASK:  1  # 1 = verbose
FORMAT_MASK:  6  # 2(terse) + 4(model-mag)
FORMAT_MASK: 18  # 2(terse) + 16(RANDOM CID)
FORMAT_MASK: 26  # 2(terse) + 8(BLIND) + 16(RANDOM CID)
FORMAT_MASK: 32  # FITS format (default for version >= v9_82)
FORMAT_MASK: 48  # 32(FITS) + 16(RANDOM CID)
```

and the sub-sections below give more details.

### 4.24.1 Verbose Light Curve Output

“**FORMAT\_MASK: 1**” results in a verbose text-output that includes extra meta-data such as the PSF, sky-noise, K-correction values, etc ...

#### 4.24.2 Terse Light Curve Output (Default)

“**FORMAT\_MASK: 2**” This option results in a simplified one-line-per-observation output for the light curves. Header information includes RA, DECL, redshift, etc ... This output is recommended for analysis with non-SNANA programs that need to parse data files generated by the SNANA simulation. Below is a sample of the output.

```
# TERSE LIGHT CURVE OUTPUT:
#
NVAR: 9
VARLIST:  MJD  FLT FIELD  FLUXCAL  FLUXCALERR  SNR  MAG  MAGERR  SIM_MAG
OBS: 49562.316  Y 1694 -1.333e+03  5.891e+02 -2.26 128.000  0.000  27.313
OBS: 49572.430  z 1694  6.500e+01  1.708e+02  0.38  26.628 101.372  25.385
OBS: 49590.422  Y 1694  1.492e+02  1.635e+02  0.91  24.236 103.764  24.064
OBS: 49591.387  i 1694  3.733e+03  4.644e+02  8.04  23.970  0.144  24.198
OBS: 49591.414  z 1694  1.644e+03  3.271e+02  5.03  23.850  0.241  24.200
...
```

It is recommended to use the fluxes instead of mags because the mags are not defined for negative fluxes, and are ill-defined for very small fluxes. The FIELD is given for each measurement to properly label overlapping fields, SNR is the signal-to-noise ratio ( $FLUXCAL/FLUXCALERR$ ) and SIM\_MAG is the exact magnitude (without noise fluctuations) computed from the SN model.

To get both the verbose and terse formats, note that **FORMAT\_MASK** is actually a bit-mask; therefore, specifying “**FORMAT\_MASK: 3**” results in both outputs in each **SNDATA** file.

#### 4.24.3 Model-Mag Light Curve Output

“**FORMAT\_MASK: 4**” Dump out the model mag info. Set value to 6 to dump both the the nominal output and the model-mag output. A sample output is as follows:

```
# MODEL MAG OUTPUT:
NVAR: 7
VARLIST:  TOBS  FLT  MAGOBS  MAGERR  MAGREST  KCOR(SYM,VAL)
OBS:    -1.328  u   21.547  0.055  -19.810  K_Uu  1.379
OBS:    -1.328  g   20.228  0.035  -19.396  K_Bg  -0.205
OBS:    -1.328  r   20.182  0.037  -19.421  K_Vr  -0.157
etc ...
```

#### 4.24.4 Suppress SIM\_XXX Info

“**FORMAT\_MASK: 8**” Suppress **SIM\_XXX** header info. This option is useful for things like blind-testing photometric classifiers.

#### 4.24.5 Random CID

“**FORMAT\_MASK: 16**” Generate random (integer) CID from 1-999,000 instead of the default sequential generation. The purpose of this option is that mixing different SN samples (Ia,II,Ibc) cannot be sorted by CID. Any random subset of the combined SN sample will contain similar fractions of each SN type.

Note that the keyword `CIDOFF` plays the role of selecting a unique set of random CIDs so that combined SN samples will not have overlapping CIDs. For example, suppose you generate 1000 type Ia SNe with `CIDOFF: 0`. The CIDs will be the first 1000 randomly selected (and non-repeating) integers between 1 and 999,000. Now suppose you generate 1000 type II with `CIDOFF: 1000`. The simulation will generate 2000 CIDs, but only use the last 1000 on the list (i.e., skip the first 1000). When the type Ia and type II SNe are combined (see `sim_SNmix.pl`), the CIDs will not overlap and the SN types (Ia and II) will be perfectly mixed with no correlation between type and CID. It is up to the user to pick the correct `CIDOFF` value for each SN type, although `sim_SNmix.pl` will automatically assign the appropriate `CIDOFF` values. After combining the SN samples, a useful unitarity check is to do ``ls *.DAT | wc`` and verify that the total number of files matches the expected sum from the simulation jobs.

#### 4.24.6 FITS Format

“**FORMAT\_MASK: 32**” While the above `TERSE` and `VERBOSE` options use `TEXT` (ASCII) format, this (default) option uses the more compact binary-FITS format that is processed using the `cfitsio` library. The advantage of this format is that there are very few files to manage, reading is much faster compared to the `TEXT` options, the compact binary files are portable to any computing platform, and there are public fits-viewing utilities such as “fv.”

Two fits files are created by the simulation. First is a `HEAD` file with a one-row summary for each SN. The summary info includes the `SNID`, sky coordinates, Galactic extinction, and many other quantities. The `HEAD` file also contains the name of the second “`PHOT`” file which contains the photometric light curves. All of the light curves are written sequentially into one `PHOT`-table, and pointers from the `HEAD` file (see `PTROBS_MIN` and `PTROBS_MAX`) are used to select the appropriate rows from the `PHOT` table. For a given SN-data version, the `[VERSION].LIST` file contains a list of `HEAD` fits-files. Usually there is only one such fits-file, but several can be combined into one version, such as combining files from different seasons.

Finally, the binary-FITS format can be used for data as well as for simulations. However, `SNANA` does not have tools to translate `TEXT` format into `FITS` format; such tools may become available in a future update.

## 4.25 Simulation Dump Options

### 4.25.1 SIMLIB\_DUMP Utility

To quickly check a SIMLIB, a screen-dump summary is obtained with the command:

```
> snlc_sim.exe mysim.input SIMLIB_DUMP 0
*****
SIMLIB_DUMP

LIBID  MJD-range      NEPOCH(all,gri) GAPMAX(frac) <GAP>
-----
001    53616-53705     126,42 42 42    11.0(0.12)  2.2
002    53622-53700     51,17 17 17    19.0(0.24)  4.9
003    53622-53705     69,23 23 23    10.1(0.12)  3.8
004    53622-53705     54,18 18 18    15.0(0.18)  4.9
...
050    53622-53705     57,19 19 19    15.0(0.18)  4.6

Done reading 496 SIMLIB entries.

LIBRARY AVERAGES PER FILTER:
          <PSF>
    <ZPT-pe> FWHM  <SKYSIG>  <SKYMAG>          Cadence
FLT  (mag)  (asec) (ADU/pix) (asec^-2) <m5sig> <Nep>  FoM
-----
  u   30.86  0.866      9.0   22.65   19.99   2.86   0.036
  g   34.31  0.828    117.8   20.75   19.99  29.71   0.123
  r   35.04  0.820    173.5   20.50   19.99  29.71   0.130
  i   34.81  0.829    216.2   19.74   19.99  31.43   0.128
  z   34.18  0.823    205.7   19.20   19.99  32.14   0.135
  Y   32.93  0.822    197.9   17.99   19.99  30.14   0.127

LIBRARY MIN-MAX RANGES:
  RA:    -59.994 to 58.766 deg
  DECL:  -1.253 to 1.257 deg
  MJD:   53616.2 to 53705.4

CUT-WARNING: 46 SIMLIBS will fail user-cut on 'RA'
```

GAPMAX and <GAP> are the maximum and average gaps (days) between epochs in the SIMLIB. The “frac” after GAPMAX is the fraction of the MJD-range consumed by the largest gap. The LIBRARY MIN-MAX RANGES show you the ranges needed to include all SIMLIB entries. The CUT-WARNINGS shows

how many SIMLIB entries are excluded by the selection ranges in your sim-input file. CUT-WARNINGs are checked for RA, DECL and PEAKMJD.

In addition to the screen-dump, a one-line summary for each LIBID is written to [simlib].DUMP where [simlib] is the name of the SIMLIB file that you specify. This file is self-documented like the “fitres” files, and can be converted into an ntuple using the “combine\_fitres.exe” program (§12.1.1).

#### 4.25.2 Cadence Figure of Merit Utility

A figure of merit for the cadence can be determined in two ways. First, you can extract the function “SNcadenceFoM” from sntools.c and pass the arguments from your own wrapper function. The second method is to simply analyze any SIMLIB using the SIMLIB\_DUMP option (§ 4.25.1). The FoM is appended to each entry in the output [simlib].DUMP file. The FoM for each simlib entry is a function of the MJD and the  $5\sigma$  limiting magnitude for each observation.

#### 4.25.3 SIMGEN\_DUMP File

To quickly analyze generated distributions, there is an option to dump generated quantities to a column-formatted text file. For example, to check the generated redshift and SALT-II parameters, add the following to your sim-input file:

```
SIMGEN_DUMP: 5  CID  Z  S2x0 S2x1 S2c
or
snlc_sim.exe mysim.input  SIMGEN_DUMP 5  CID Z S2x0 S2x1 S2c
```

which produces an auxiliary file [VERSION].DUMP in the same directory as the SNDATA files. The self-documented dump-file looks like:

```
NVAR: 4
VARNAMES:  Z S2x0 S2x1 S2c
SN:  50001  1.3820e-01 3.8970e-04 1.0906e+00 -1.5431e-01
SN:  50002  3.1260e-01 1.0278e-04 1.8671e-01 -3.9044e-01
SN:  50003  2.4248e-01 1.9417e-04 8.8190e-01 -3.8711e-01
SN:  50004  2.5666e-01 8.3385e-05 -6.7122e-01 -1.5304e-01
```

A full list of allowed SIMGEN\_DUMP variables can be printed to the screen by specifying zero variables as follows:

```
snlc_sim.exe mysim.input SIMGEN_DUMP 0
```

After printing the variables, the program quits.

#### 4.25.4 Rest-Frame Model Dump

```
GENRANGE_DMPREST: -20 80 # dump rest-frame model for this Trest range
GENMAG_SMEAR:      0.0
GENMODEL_ERRSCALE: 0.0
```

## 4.26 Including a Second Sim-Input File

A sim-input file can be split into two files using the keyword

```
INPUT_FILE_INCLUDE: my2nd.input
```

which instructs the simulation to read and parse “my2nd.input” in exactly the same way as the original sim-input file. To see why this might be useful, consider the non-Ia simulation that has many “NON1A:” keywords. The NON1A keys can be stripped out into a separate file such as NON1A\_keys.input, and then included in many sim-input files. Thus a dozen sim-input files can each include NON1A\_keys.input. To modify or add a NON1A key for all of the sim-input files, only one file needs to be modified.

## 4.27 Multi-dimensional GRID Option

Instead of generating random distributions in the variables describing each SN (redshift, luminosity parameter, color, etc ..), the simulation can generate SNe on a well-defined grid for each parameter using the following sim-input options,

```
GENSOURCE:      GRID    # replaces RANDOM option
NGRID_LOGZ:     20     # log10(redshift)
NGRID_LUMIPAR:  10     # x1, Delta, stretch,dm15 ...
NGRID_COLORPAR: 2      # AV or SALT2 color
NGRID_COLORLAW: 1      # RV or BETA
NGRID_TREST:    56     # rest-frame epoch
GRID_FORMAT:    FITS   # TEXT or FITS

GENRANGE_REDSHIFT: 0.01 1.2      # redshift range
GENRANGE_DELTA:   -0.4 1.8      # delta-range (mlcs only)
GENRANGE_RV:      2.2 2.2      # range of CCM89-RV
GENRANGE_AV:      0.0 2.00     # AV range
GENRANGE_TREST:   -20.0 90.0    # test epoch relative to peak (days)
GENFILTERS:      griz
```

and explicit examples of complete sim-input files are in

```
$SNDATA_ROOT/analysis/sample_input_files/GRID
```

This GRID option allows external (non-SNANA) fitting programs to use the SNANA models. The original motivation is for the photometric SN id program (§8). Each NGRID\_XXX value divides the corresponding GENRANGE\_XXX range into the specified number of bins for the grid. The “GRID\_FORMAT: TEXT” option produces a human-readable file intended only for visual inspection. The “GRID\_FORMAT: FITS” option produces a platform-independent file to be read by external programs. To save memory for programs reading the FITS tables, the magnitudes and errors have been multiplied by 1000 and stored as 16-bit (2-byte) integers. Magnitudes dimmer than 32 are written as 32000, and undefined model magnitudes are stored as -9000 (i.e, mag= -9).

The GRID file is written in the same directory as the auxiliary files

```
$SNDATA_ROOT/SIM/MY_VERSION/MY_VERSION.GRID
```

where “GENVERSION: MY\_VERSION” is specified in the sim-input file. Note that only the GRID file is written; no light curve files are written out.

The FITS tables can be visually examined using a utility such as “fdump” or “fv.” SNANA has a “fits\_read\_SNGRID” utility to read in the generated GRID; to use this utility the following code-lines must be included,

```
#define SNGRIDREAD // use only the read utilities in sngridtools.c
#ifdef SNGRIDREAD
#include "fitsio.h"
#include "sngridtools.h"
#include "sngridtools.c" // fits_read_SNGRID is in here
#endif
```

The read-back utility fills the following global arrays/structures in sngridtools.h,

```
GRIDGEN_INFO
GRIDGEN_SURVEY  GRIDGEN_MODEL  GRIDGEN_FILTERS
PTR_GRIDGEN_LC  I2GRIDGEN_LCMAG I2GRIDGEN_LCERR
```

Also note that genmag\_snoopy.c illustrated how to read and access the GRID.

Here is a brief description of the FITS tables and how to look up the correct magnitude and error from a set of SN parameters. Technically only the first (SNPAR-INFO) and last (I2LCMAG) tables are needed; the intermediate tables provide additional information that you would otherwise have to compute on your own. The SNPAR-INFO columns NBIN, VALMIN and VALMAX are simply copied from the sim-input parameters. The BINSIZE is calculated from the previous parameters, and the ILCOFF are used to determine the absolute light curve index (ILC) as a function of the SN parameters as follows:

$$\text{ILC} = 1 + \sum_{i=1}^4 \text{ILCOFF}_i \times (\text{INDX}_i - 1) \quad (12)$$

The parameter index  $i = 1, 4$  runs over (1) redshift, (2) color ( $A_V$  or  $c$ ), (3) color law ( $R_V$  or  $\beta$ ) and (4) luminosity parameter. Each integer index  $\text{INDX}_i$  runs from 1 to  $\text{NGRID}_i$  for parameter  $i$ . Do NOT extend the summation to include the filter and epoch indices. While the physical grid-values corresponding to each  $\text{INDX}_i$  can be computed from the SNPAR-INFO table, these grid values have been store in the intermediate tables that have a GRID suffix (and include FILTER-GRID and TREST-GRID).

Note that the  $\text{ILCOFF}_i$  are fixed, while the  $\text{INDX}_i$  depend on the set of SN parameters. For example, consider a redshift range of 0.01 to 1 and 200 bins; in logz space we have  $-2 \leq \log_{10}(z) \leq 0$  and a logz binsize of 0.01. For  $z = 0.1$ ,  $\log_{10}(z) = -1$  and the GRID-index is  $\text{INDX}_1 = 100$ .

Now we have an ILC index corresponding to a Supernova described by the four parameters above. Each SN light curve is written out in all of the GENFILTERS, and all of the SNe are strung together in the

I2LCMAG table. This table contains one column of model magnitudes and another column of model errors, each multiplied by 1000 to maintain millimag precision in 2-byte integer storage. The starting location in the I2LCMAG table is given in a separate 'pointer table' by PTR\_I2LCMAG(ILC). Note that you could compute this pointer as

$$\text{PTR\_I2LCMAG[ILC]} = 1 + ((\text{NGRID\_FILT} * \text{NGRID\_TREST}) + \text{NWDPAD}) * (\text{ILC}-1);$$

where NGRID\_FILT is the number of "GENFILTERS" and NWDPAD= 4 is the number of pad-words. Starting at the specified pointer location for ILC, the first word is a pad-word (-1111) and the second word is the first 8 bits of ILC; read-programs should verify these words to avoid getting lost. The next NGRID\_TREST words are the magnitudes ( $\times 1000$ ) for the first filter (g), the next NGRID\_TREST words are the magnitudes for the second filter (r), etc.. Finally, after reading all of the light curve magnitudes there are two end-of-lightcurve pad-words with values of -9999.

The I2LCMAG storage for a single SN light curve is illustrated below:

```

pad1 = -1111 <== start I2LCMAG address is PTR_I2LCMAG(ILC)
pad2 = first 8 bits of ILC
I2LCMAG(g,ep1) = mag * 1000
I2LCMAG(g,ep2)
I2LCMAG(g,ep3)
...
I2LCMAG(g,NGRID_TREST)
I2LCMAG(r,ep1)
...
I2LCMAG(r,NGRID_TREST)
...
...
I2LCMAG(z,NGRID_TREST)
pad3 = -9999
pad4 = -9999

```

While pointers are provided to compute ILC and to determine the starting I2LCMAG address from ILC, you are on your own to find the sub-index corresponding to the filter and epoch.

For the non-Ia GRID, there is no physically meaningful luminosity parameter; this parameter is therefore used to store a sparse index that runs from 1 to the number of non-Ia templates that are specified with the "NON1A:" keyword in the sim-input file. The FITS file includes an additional NONIa-INFO table that gives the SNANA index, a character-string type (e.g., II, Ib, Ibc), and a character-string name of the underlying SN used to create the template (e.g., 'SDSS-002744').

## 5 The SNANA Fitter: `snlc_fit.exe`

### 5.1 Getting Started Quickly

Here you will perform lightcurve fits, hopefully in under a minute. To get started,

```
> cp $SNDATA_ROOT/analysis/sample_input_files/mlcs2k2/snfit_SDSS.nml .  
    or  
> cp $SNDATA_ROOT/analysis/sample_input_files/SALT2/snfit_SDSS.nml .
```

(Edit `.nml` file and put in correct `VERSION_PHOTOMETRY = 'xxx'`)

```
> snlc_fit.exe snfit_SDSS.nml >! snfit_SDSS.log &
```

When the unix command “ps” shows that the job has finished, congratulations ! You have fit your lightcurves with CERNLIB’s MINUIT program.<sup>6</sup> To create a pdf file showing the light curve fits (§5.8),

```
mkfitplots.pl --h snfit_SDSS.his
```

creates `snfit_SDSS_fits.pdf`. If you are shaking your head wondering what the heck you just did, that’s a good sign.

### 5.2 Discussion of Lightcurve Fits

Before reading this section, make sure you have successfully run the commands described in §5.1. Let’s start the discussion by checking the end of the log-file,

```
> tail snfit_SDSS.log
```

The very last line should be “ENDING PROGRAM GRACEFULLY.” If you do not see this, check that your namelist variable `VERSION_PHOTOMETRY` is really pointing to an existing version in `$SNDATA_ROOT/SIM`. If you still have trouble, contact an expert for help.

Inside the input file `snfit_SDSS.nml`, the namelist variable

```
FITRES_DMPFILE = 'snfit_SDSS.fitres'
```

results in a dump of the fit parameters for each SN in a self-documented “fitres” file. Go ahead and “more `snfit_SDSS.fitres`” to see the results. The header keywords `NVAR` and `VARNAMES` specify the columns. The SNANA library includes a utility called `RDFITRES` to read these files. You can “cat” multiple fitres files together and add comments, and still read them with the same parsing code.

To figure out what you did, you need to check the namelist options in `snfit_SDSS.nml`. There are two separate namelists:

---

<sup>6</sup><http://wwwasdoc.web.cern.ch/wwwasdoc/minuit/minmain.html>

- `&SNLCINP`: defines selection of SNe and epochs by specifying criteria for the number of epochs, earliest & latest times relative to peak, maximum signal-to-noise, etc ... All namelist options are defined and commented inside `SNANA_DIR/src/snana.car`.
- `&FITINP`: defines fitting options such as priors, marginalization, and range of  $T_{\text{rest}}$  in the second fit-iteration. All namelist options are defined and commented inside `SNANA_DIR/src/snlc_fit.car`.

In the sample namelist file, a prior on  $A_V$  is used by setting `PRIOR_AVEXP = 0.40`, which translates into a prior of the form  $\exp(-A_V/0.4)$ . There is no marginalization so that your first fits run much faster. To marginalize, set the number of integration bins per variable, `NGRID_PDF = 11`. Since the marginalization is over four fit variables ( $t_0, A_V, \Delta, \mu$ ), the CPU-time goes as the fourth power of `NGRID_PDF`; previous studies indicate that 11 bins per fit-variable is a good compromise between accuracy and CPU time.

### 5.3 Methods of Fit-Parameter Estimation

There are three methods that can be used to estimate light curve fit-parameters:

1. **MINIMIZATION** based on CERNLIB's MINUIT program<sup>7</sup>. `&SNLC_INP` namelist parameter `NFIT_ITERATION` specifies the number of iterations (2 is recommended). You must always use this option, even if you use the options below.
2. **MARGINALIZATION** using multi-dimensional integration in SNANA function `MARG_DRIVER`. `&FITINP` namelist parameter `NGRID_PDF` controls the number of grid-points per fit-parameter (11 is recommended). You must run the minimizer first (`NFIT_ITERATION=2`) to get starting values and integration ranges. After marginalizing, the following crosschecks are performed: probability at the boundaries and number of bins with zero probability; if either is too large, the integration ranges are adjusted and the marginalization repeats.
3. **Monte Carlo Markov Chain (MCMC)**: See `&MCMCINP` namelist parameters.

---

<sup>7</sup><http://wwwasdoc.web.cern.ch/wwwasdoc/minuit/minmain.html>

## 5.4 Initial Fit-Parameter Estimation

For MINUIT to converge, initial fit-parameters must be chosen so that the initial model light curve roughly overlaps the data. The color and shape parameter are fixed to an average value, and the distance modulus (or  $x_0$  for SALT-II) is adjusted so that the model matches the data.

The estimate of the epoch of peak brightness ( $t_0$ ) is usually read from the data file from the keyword SEARCH\_PEAQMJD. If this keyword is missing, then SNANA will try to estimate  $t_0$  by fitting a general function of the form (see SNANA function SET\_PEAQMJD)

$$f(t) = A[1 + a_1(t - \bar{t}) + a_2(t - \bar{t})^2] \times \frac{\exp[-(t - \bar{t})/T_{\text{fall}}]}{1 + \exp[-(t - \bar{t})/T_{\text{rise}}]}, \quad (13)$$

as suggested in the SNLS CC rate paper (Bazin et al, 2008). The fitted parameters for each filter are  $A, a_1, a_2, \bar{t}, T_{\text{fall}}, T_{\text{rise}}$ . The time at peak is obtained by setting the derivative equal to zero:  $t_0 = \bar{t} + T_{\text{rise}} \ln(T_{\text{fall}}/T_{\text{rise}} - 1)$ . The initial  $\bar{t}$  value is estimated to be the epoch with maximum flux. Each filter is fit independently and the final  $t_0$  is the filter-weighted average. A filter's  $t_0$  is dropped from the average if: (1) its max-flux measurement has the smallest  $S/N$  ratio among filters and has  $S/N < 10$ , or (2) its  $t_0$  value is more than 30 days away from the average of the other filter- $t_0$  values (tested only if there are 3 or more filters). The max-flux epoch must have  $S/N > 4$  to make an estimate of  $t_0$ .

Bit-mask options (lsb=1) via &SNLCINP namelist variable OPT\_SETPKMJD are: bit-1) fix  $a_1 = a_2 = 0$ , bit-2) float  $a_1$  and  $a_2$ , and bit-8) dump info for each SN. Explicit examples are :

```

OPT_SETPKMJD = 1    # fix a1 = a2 = 0 (default => no polynomial term)
OPT_SETPKMJD = 2    # float a1 & a2
OPT_SETPKMJD = 129 # fix a1=a2=0 and DUMP info for each SN

```

Users should compare the initial  $t_0$  (from above) to the final  $t_0$  from the light curve fit and check for outliers; outliers can be fixed by visual inspection of the light curve and setting the SEARCH\_PEAQMJD keyword in the data file. To avoid clogging up the standard log-file, the MINUIT output for these fits is dumped to a separate log-file specified by namelist variable MNFIT\_PKMJD\_LOGFILE: the default filename is MNFIT\_PKMJD.LOG.

WARNING (Jun 2010): The SN classifier challenge has uncovered two serious problems with the initial  $t_0$  estimate for SNe Ia using Eq. 13. First, the filter-dependence of  $t_0$  is not accounted for. Second, Eq. 13 can sometimes be very nonIa-like (even for a perfectly good fit), leading to  $t_0$  estimates off by more than a week. Will need to add an option to use a more Ia-like function.

## 5.5 Galactic Extinction Uncertainties (as of v9\_96)

Galactic extinction uncertainties are included with the `&FITINP` namelist option

```
OPT_XTMW_ERR = 1
```

The reduced  $N_{\text{obs}} \times N_{\text{obs}}$  covariance matrix is 1 everywhere, and each diagonal element is the MWEBV uncertainty at the central wavelength of the observer-frame passband. The MWEBV uncertainty is read from each data file header; if this value does not exist in the header then the uncertainty is internally set to  $0.16 \times \text{MWEBV}$ . The covariance matrix is computed and inverted between fit iterations.

Current default is `OPT_XTMW_ERR=0`, but this will change in a few versions.

## 5.6 Fitting Priors

The fitting prior options in `snlc_fit.exe` are mainly designed to prevent catastrophic fits. There are also options related to the host-galaxy extinction. Photo- $z$  priors are described in §5.10, and a brief description of the other `&FITINP` namelist prior options are given below.

- **PRIOR\_MJDSIG**: sigma on Gaussian prior for MJD at peak brightness. Default is 20 days.
- **PRIOR\_LUMIPAR\_RANGE(2)**: range of flat prior for luminosity parameter to prevent crazy values. Typical prior-range values are  $\{-0.5, 2.0\}$  for the MLCS2k2 parameter  $\Delta$ , and  $\{-5, +3\}$  for SALT-II parameter  $x_1$ . Default range is  $\{-9, +9\}$ . The parameter below defines a smooth roll-off at the edges.
- **PRIOR\_LUMIPAR\_SIGMA**: sigma of Gaussian roll-off at edge of flat prior defined above. Default is 0.1.
- **PRIOR\_DELTA\_PROFILE(4)**: Used only for MLCS2k2  $\Delta$ , the first two elements are  $-\sigma$  and  $+\sigma$  for the asymmetric Gaussian prior, the 3rd element is the  $\Delta$  value at the Gaussian peak, and the 4th element is the minimum 'flat' probability for all  $\Delta$  values within `PRIOR_LUMIPAR_RANGE`. A flat  $\Delta$  prior is obtained by simply setting the 4th element to 1.0. Setting the 4th element to  $\sim 0.1$  results in a Gaussian prior with a flat tail for large  $\Delta$  values; this tail prevents the suppression of very fast decliners (91bg-like).
- **OPT\_PRIOR**: Default is 1  $\rightarrow$  use priors. Setting to zero turns off ALL priors regardless of their values.
- **OPT\_PRIOR\_AV**: Used only for MLCS2k2 model, default is 1  $\rightarrow$  use  $A_V$  priors. Setting to zero turns off  $A_V$ -related priors.
- **PRIOR\_AVEXP(2)**: For MLCS2k2 only, defines up to two exponential slopes for  $A_V$  prior.
- **PRIOR\_AVWGT(2)**: For MLCS2k2 only, defines weight for the two  $A_V$ -exponential terms.
- **PRIOR\_AVRES**: Since the  $A_V$  prior has a sharp boundary at  $A_V = 0$  and therefore a discontinuity in the fitting function, this `PRIOR_AVRES` option allows a Gaussian smearing of the prior function that results in a continuous function. Recommended values are .001 to 0.01.

## 5.7 Selecting an Efficiency Map for a Fitting Prior

The light curve fitting prior includes an optional simulated efficiency as a function of redshift, Galactic extinction (MWEBV) and model-dependent parameters that describe the SN brightness. For example, the MLCS2k2 model parameters are shape ( $\Delta$ ), extinction ( $A_V$ ), and color law ( $R_V$ ). The fitting prior and efficiency map can be applied to other models too. An efficiency map can be generated using the SNANA script `SIMEFF_MAPGEN.pl` (§4.23), and a map is selected via the `&FITINP` namelist:

```
! Select file name explicitly. Will first check YOUR current working
! directory; if not there, then fitter checks public area
! $SNANA_ROOT/models/simeff/mysimeff.dat
SIMEFF_FILE = 'mysimeff.dat'
```

The efficiency map is defined on a multi-dimensional grid, and interpolation is used to determine the efficiency for any set of SN model parameters. Note that these maps depend on the selection requirements and are therefore analysis-specific; these maps should therefore not be considered as general purpose files such as those describing K-corrections (§7) or search-efficiencies (§4.15).

## 5.8 Viewing Lightcurve Fits: `mkfitplots.pl`

There is an after-burner script to prepare a pdf file showing each light curve (data + fit) for each pass-band,

```
> mkfitplots.pl --h snlc_fit.his
```

The “`snlc_fit.his`” argument above is the name of the hbook file that was specified with the namelist argument `HFILE_OUT` inside the `&SNLCINP` namelist. This script creates a file called `snlc_fit_fits.pdf`. More generally, the pdf file name has the same prefix as the hbook file name. On each plot the black dots are data (or simulated data) and the green curve is best-fit model. There are many plotting options; see instructions with

```
more \ $SNANA_DIR/util/mkfitplots.pl
```

To view the light curves without doing any fits, set `&SNLCINP` namelist variable `OPT_LCPLOT=1`, run the `snana.exe` program, and then run the above `mkfitplots.pl` command.

For versions prior to `v10_17` you need to run a once-in-a-lifetime command

```
> paw_setup.cmd
```

For later versions there is no need to run this script, and there is no need to maintain a private `/kumacs` directory.

## 5.9 Tracking SN versus Cuts

Typically the final number of processed SN is smaller than than the number read in, and thus it is often useful to track the losses, particularly if there seem to be too few (e.g., zero) SNe. At the end of each snana job, the stdout includes a statistics summary for each SN “Type” showing the number of SN vs. incremental cut. An example is shown here,

CUT NAME	ITYPE=	Number of SN passing incremental cut for		
		118	119	120
CID		8	37	502
Trestmin		5	36	476
Trestmax		5	34	374
NFILT(SNRmax)		4	31	370
FIT + CUTS		4	30	369

The last row labelled 'FIT+CUTS' is shown for the `snlc_fit.exe` job, and the previous rows are shown for both the `snana.exe` and `snlc_fit.exe` jobs. Additional information is given by the following snana-ntuple (ntid 7100) variables:

```
CUTFLAG_SNANA = 0    -> failed snana cuts
CUTFLAG_SNANA bit0  -> passed snana cuts
CUTFLAG_SNANA bit1  -> passed fit & cuts

ERRFLAG_FIT = -9    -> no fit done (i.e., CUTFLAG_SNANA=0 or snana.exe job)
ERRFLAG_FIT = 0     -> no fitting errors
ERRFLAG_FIT > 0    -> see error codes with
                    grep 'ERRFLAG_' \${SNANA_DIR}/src/snlc_fit.car | grep '&'
```

where this ntuple is created with `LTUP_SNANA=T` (§5.23) and the variables can be extracted into text format using the `ntdump.pl` utility (§12.1.2). `CUTFLAG_SNANA=1` means that the snana cut-requirements were satisfied, but the fit either failed or was not attempted. `CUTFLAG_SNANA=3` means that the SN satisfied the snana cuts and has a valid fit whose results are stored in the `fitres` file and the `fitres` ntuple.

`ERRFLAG_FIT` is zero if the fit is valid and the fit-cuts are satisfied. A positive flag indicates an error; `grep` the source code for error definitions. `ERRFLAG_FIT=-9` means that the fit was not attempted; this occurs if the snana cuts fail (`CUTFLAG_SNANA=0`) or when running `snana.exe`.

## 5.10 PhotoZ Fits

Here we describe light curve fits that determine the SN Ia redshift ( $z$ ) from photometry, called “SN-photoZ” fits. There are two fundamental methods to perform photoZ fits. The first method, called a “constrained photoZ fit,” is designed to identify SNe Ia that do not have a spectroscopic redshift: uses include SN rates and targeting host-galaxy redshifts for unconfirmed SN Ia. For MLCS2k2 photoZ fits, there are four floated parameters:  $z$ ,  $t_0$ ,  $\Delta$ , and  $A_V$ . For SALT-II photoZ fits, the four floated parameters are  $z$ ,  $t_0$ ,  $x_1$ , and  $c$ . The distance modulus is constrained (calculated) assuming a particular cosmology:  $\mu = \mu(z, \Omega_M, \Omega_\Lambda, w)$  where  $z$  is floated in the fit, and  $\Omega_M, \Omega_\Lambda, w$  are fixed by the user. The cosmology can be specified with &SNLCINP namelist parameters H0\_REF, OMAT\_REF, and OLAM\_REF. For SALT-II photoZ fits, the distance modulus is converted into the  $x_0$  parameter, and therefore SALT2alpha & SALT2beta must be specified as &FITINP namelist parameters.

The second method, called a “cosmology-photoZ” fit, involves floating five parameters:  $\mu$ ,  $z$ ,  $t_0$ ,  $\Delta$ , and  $A_V$  for MLCS2k2, and  $x_0$ ,  $z$ ,  $t_0$ ,  $x_1$ , and  $c$  for SALT-II. This method is designed to use large photometric samples to measure distance moduli that can be used to measure cosmological parameters. One of the difficulties with the 5-parameter fit is CPU time: the marginalization takes a few minutes per fit, so studying the bias on a sample of  $10^4$  simulated SNe Ia requires about a CPU-month of resources.

In addition to the two main methods above, there are variations that involve using the host-galaxy photoZ (host-photoZ) as a prior to help constrain the redshift. A distance-modulus prior can be applied to the second method (5-parameter fit); this is essentially a constrained-photoZ fit, but the photoZ errors will include uncertainties from the cosmological parameters. Needless to say, *never* run a cosmology fit on output where a distance-modulus prior is used !

There are five &FITINP namelist parameters that control photoZ fits. The default values are set so that photoZ fits are turned off,

DOFIT_PHOTOZ	=	F	
OPT_PHOTOZ	=	0	! 1=>hostgal photZ prior; 2=> specZ prior
INISTP_DLMAG	=	0.1	! 0=> constrain DLMAG; non-zero => float DLMAG
PRIOR_ZERRSCALE	=	100.0	! scale error on host-photoZ prior
PRIOR_MUERRSCALE	=	100.0	! scale error on distance modulus prior

Setting DOFIT\_PHOTOZ=T and INISTP\_DLMAG=0.0 results in a 4-parameter constrained-photoZ fit. Since PRIOR\_ZERRSCALE=100.0 by default, the host-photoZ errors are multiplied by 100 and therefore have no impact on the fits. Setting PRIOR\_ZERRSCALE = 1.0 results in using the host-photoZ prior described by a Gaussian distribution<sup>8</sup>.

To switch from a constrained-photoZ fit to a 5-parameter cosmology-photoZ fit, set INISTP\_DLMAG to any non-zero value such as 0.1. The use (or non-use) of the host-photoZ prior is controlled by the value of PRIOR\_ZERRSCALE. The parameter OPT\_PHOTOZ controls the source of the redshift prior. When you set DOFIT\_PHOTOZ=T, OPT\_PHOTOZ is automatically set to 1 so that the host-photoZ prior is used. If you set OPT\_PHOTOZ=2, the spectroscopic redshift is used as a prior: this option is designed solely as a sanity check on the light curve fitter, and is not meant to use for science. If you set OPT\_PHOTOZ > 0,

<sup>8</sup>Depending on user interest, non-Gaussian tails may be added later.

the `DOFIT_PHOTOZ` flag is automatically set, and vice-versa: thus you can turn on the photoZ option with either namelist variable.

If `PRIOR_MUERRSCALE` is 100 or larger (the default), then there is no prior applied to the distance modulus ( $\mu$ ). Setting `PRIOR_MUERRSCALE = 2` will apply a  $\mu$ -prior using a Gaussian profile of width  $\sigma = 2\sigma_\mu$ , where  $\sigma_\mu$  is calculated from the user-specified uncertainties in the cosmological parameters. Note that `OMAT_REF`, and `W0_REF` are two-dimensional arrays that specify both the value and error. For example,

```
OMAT_REF = 0.3, 0.03
W0_REF   = -1.0, 0.1
```

would use  $\sigma_w = 0.1$  and  $\sigma_M = 0.03$  to calculate the  $\mu$ -error for the photoZ at each fit-iteration.

To get going quickly, some useful examples of setting the namelist options are given below in Fig. 6.

A few other photoZ-related issues are:

- To test the photoZ methods in simulated SN Ia samples, the simulation includes an option to include host-galaxy photoZs based on an externally-supplied library (§ 4.18). To test SN-only photoZ fits (without host), increase `GENSIGMA_REDSHIFT` so that the initial redshift estimate is poor.
- To test the photoZ sensitivity to the initial redshift estimate, you can arbitrarily shift the redshifts using `&SNLCINP` namelist parameter `REDSHIFT_FINAL_SHIFT`.

Figure 6: Examples of setting photoZ options within the &FITINP namelist.

```
! constrained photoZ fit, ignore host-galaxy photoZ:
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 100.0 ! inflate error on photoZ prior
INISTP_DLMAG      = 0.0   ! fix MU = MU(zphot,cosmology)

! equivalent way to do the above
OPT_PHOTOZ        = 1
PRIOR_ZERRSCALE   = 100.0 ! inflate error on photoZ prior
INISTP_DLMAG      = 0.0   ! fix MU = MU(zphot,cosmology)

! constrained photoZ fit using host-galaxy photoZ:
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 1.0
INISTP_DLMAG      = 0.0   ! fix MU = MU(zphot,cosmology)

! constrained photoZ fit, host-galaxy photoZ errors inflated by 1.3
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 1.3
INISTP_DLMAG      = 0.0   ! fix MU = MU(zphot,cosmology)

! cosmology photoZ fit, ignore host-galaxy photoZ:
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 100.0 ! inflate error on photoZ prior
INISTP_DLMAG      = 0.1   ! float DLMAG

! cosmology photoZ fit using host-galaxy photoZ:
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 1.0
INISTP_DLMAG      = 0.1   ! float DLMAG

! cosmology photoZ fit with priors on both distance & host-galaxy photoZ:
DOFIT_PHOTOZ      = T
PRIOR_ZERRSCALE   = 1.0   ! use host-galaxy photoZ errors
PRIOR_MUERRSCALE  = 3.0   ! use x3 mu-error calculated from dw & dOM
INISTP_DLMAG      = 0.1   ! float DLMAG
```

### 5.10.1 Redshift-Dependent Selection in PhotoZ Fits

There is a subtle fitting issue concerning the usable observer-frame filters for which  $\lambda_{\text{obs}}/(1+z)$  is within the valid  $\lambda_{\text{rest}}$ -range of the light curve model, and for which  $T_{\text{rest}} = T_{\text{obs}}/(1+z)$  are valid. In addition, requirements on quantities such as the min & max  $T_{\text{rest}}$  are ambiguous before the photoZ fit has finished, yet it is useful to make such cuts before fitting to prevent fitting pathological light curves and to reduce processing time. Here we discuss how to select filters and how to make cuts on  $T_{\text{rest}}$ -dependent quantities.

For regular cosmology fits using spectroscopic redshifts, a list of usable filters & the  $T_{\text{rest}}$ -range is made before the fit starts. For a photoZ fit, however, it is not clear which filters & epochs are valid until the fit has finished. For example, consider a *gri* photoZ fit using SALT-II: when  $Z_{\text{phot}} < 0.072$ , *i*-band maps to rest-frame wavelengths greater than the 7000 Å cutoff in SALT-II. Including *i*-band in the fit results in using an unphysical region of the model, while dropping *i*-band measurements results in a discontinuous drop in the  $\chi^2$ . In the latter case, the minimizer is trapped in this low- $\chi^2$  well, and quite often the minimum occurs at the drop-out boundary  $Z_{\text{phot}} = 0.072$ . Another example is in DES & LSST, where *g*-band maps below 3000 Å at redshifts above about 0.5.

To make initial  $T_{\text{rest}}$ -dependent cuts before the photoZ fit has started, the cuts are loosened by a factor of “ $1 + Z_{\text{max}}$ ”, where  $Z_{\text{max}} = \text{PHOTOZ\_BOUND}(2)$  is the maximum allowed redshift (specified in &FITINP). The  $T_{\text{rest}}$ -cuts are therefore loosened to be valid for any redshift in the range specified by PHOTOZ\_BOUND. For example, consider PHOTOZ\_BOUND = 0,1 and a requirement that the min- $T_{\text{rest}}$  is before  $-4$  days; the initial cut would be a requirement of an epoch before  $-2$  days using whatever REDSHIFT\_FINAL is in the data file, and the  $-4$  day requirement is applied after the photoZ fit has finished. Similarly, a max- $T_{\text{rest}}$  requirement of 30-60 days is loosened to 15-120 days before the photoZ fit. If a filter is dropped after the first fit-iteration (see below), the loose  $T_{\text{rest}}$ -cuts are re-applied before fitting again.

To select observer-frame filters, the basic strategy is to perform the first-iteration photoZ fit with all filters except for those in the UV with  $\lambda < 4000$  Å. A reasonable analytical extrapolation of the model beyond the defined wavelength range is required. This possibly biased photoZ is then used to determine which filters to exclude (or add in case of UV filter) in the next iteration. Technically, when one more filter is excluded, the first-iteration is repeated so that two complete fit-iterations are performed with the correct filters. The basic assumption in this strategy is that it does not matter if there is an unknown bias in choosing the redshift to drop a filter ... as long as the fit, with or without the filter in question, is unbiased. As an example, consider photoZ fits with *griz* filters. For  $z > 0.49$ , *g*-band should be excluded. As a safety margin, one might exclude *g*-band when the 1st-iteration photoZ value is above 0.43, or 0.44, or 0.45 ... the cut does not matter as long as we are confident that when *g*-band is used, it is within the valid range of the model.

The redshift safety margin is controlled by namelist parameters

PHOTOZ_ITER1_LAMRANGE	= 4000, 25000	! 1st-iter obs-frame lambda range
PHOTOZ_BOUND	= 1.0E06, 1.4	! hard MINUIT bound
PHOTODZ_REJECT	= 0.05	! dz cut
PHOTODZ1Z_REJECT	= -99.	! dz/(1+z) cut; default is no cut

The default `PHOTOZ_ITER1_LAMRANGE` cut is set to exclude any UV filter on the first iteration since this filter is used only at the lowest redshifts. Note that the excluded UV filter can be added back after the first fit-iteration if the photoZ value is low enough. `PHOTOZ_BOUND` is a hard MINUIT bound to prevent crazy excursions during the minimization, and is also used to loosen the  $T_{\text{rest}}$ -related cuts before the fit has started.

The next two cut-parameters define the redshift safety margin, and can be defined as a cut on  $dz$  and/or  $dz/(1+z)$ . The SNANA default is to use only the cut on  $dz$ , so we use this for discussion, noting that the other cut works in a similar manner. In principle it would be better to cut on the number of fitted  $\sigma_z$ , but on the first iteration the MINUIT errors are sometimes pathological; it is therefore safer to make a fixed cut.

The  $dz$  cut is illustrated here using  $g$ -band and the MLCS2k2 model for which the valid wavelength range is 3200-9500 Å. Since the mean filter wavelength is  $\lambda_g = 4790$  Å, the valid rest-frame redshifts are given by  $Z_{\text{min}} = \lambda_g/9500 - 1 = -0.50$  and  $Z_{\text{max}} = \lambda_g/3200 - 1 = +0.50$ . The  $g$ -band is excluded if the 1st-iteration photoZ satisfies  $Z_{\text{phot}} > Z_{\text{max}} - \text{PHOTODZ\_REJECT}$ ; in this example,  $Z_{\text{phot}} > 0.45$ . For  $Y$ -band ( $\lambda_Y = 10095$  Å),  $Z_{\text{min}} = 0.062$  and this filter is excluded if  $Z_{\text{phot}} < Z_{\text{min}} + \text{PHOTODZ\_REJECT}$ , or  $Z_{\text{phot}} < 0.11$ . Note that `PHOTODZ_REJECT` is defined to be positive when adding a safety margin, regardless of whether a blue or red band is being tested. Setting `PHOTODZ_REJECT` to a large negative value (i.e.,  $-99$ ) disables the cut. Finally, the `PHOTODZ_REJECT` cut is applied to all observer-frame filters, and often more than one filter (i.e.,  $ugr$ ) is rejected.

A quick list of dropped filters can be viewed from the log-file with the following 'grep' command:

```
grep "DROPPED" myjob.log
      WARNING for SN 40002 : DROPPED obs-filter=g
      WARNING for SN 40002 : DROPPED obs-filter=r
```

and similarly grepping for “ADDED” will find any (UV) filters that were added.

To study filter dropouts in more detail, five variables are include in the fitres-ntuple (ntid 7788):

- NEARDROP: index of filter that is closest to being dropped. Positive (negative) value indicates that this filter was included (excluded) after the first fit-iteration.
- DZMIN1: redshift safety margin for filter NEARDROP after 1st iteration. Positive value always indicates that it is within the valid region of the model; negative value indicates invalid region.
- DZMIN2: same as above, but after 2nd fit-iteration.
- DZ1ZMIN1 & DZ1ZMIN2: same as above, but for  $dz/(1+z)$ .

### TRAINING & TUNING CUTS:

SNe with spectroscopic redshifts ( $Z_{\text{spec}}$ ) should be use to train the filter-selection cuts. For each filter labeled by index “IFILT,” plot  $Z_{\text{spec}}$  when `NEARDROP = IFILT` and check that there are no (or very few) entries in the undefined redshift-region of the model. Also plot  $Z_{\text{spec}}$  when `NEARDROP = -IFILT`, and you will see entries in the undefined region, but this is OK since the filter was dropped.  $Z_{\text{spec}}$  values in

the defined region indicates that this dropped filter could have been safely used, but was rejected based on its photoZ value from the first fit-iteration. Users will have to decide the trade off between including a particular filter more often in the defined region versus using that filter more often in the undefined region.

### 5.10.2 Initial Parameter Estimate

To estimate initial parameters near the global-minimum  $\chi^2$ , a multi-dimensional grid-search is made over redshift (0.025 bin-width), color (0.2 bin-width) and 3 bins in the shape/luminosity parameter ( $x_1, \Delta M_{15}, \text{stretch} \dots$ ). The minimization over the distance parameter is done analytically as follows. For each grid point (photoZ, color, shape) the model fluxes are first evaluated using the distance (SALT2- $\bar{x}_0$  or distance modulus  $\bar{\mu}$ ) calculated from a reference set of cosmological parameters. To minimize the  $\chi^2$  w.r.t. the distance parameter, a distance-scale is determined as

$$d_{\text{scale}} = x_0/\bar{x}_0 \quad \text{or} \quad 10^{0.4(\mu-\bar{\mu})} = \sum_i (F_{\text{data}}^i F_{\text{model}}^i / \sigma_i^2) / \sum_i (F_{\text{model}}^i / \sigma_i)^2 \quad (14)$$

where  $\sigma_i$  is the quadrature sum of data and model errors and 'i' is the epoch index.<sup>9</sup> The resulting  $x_0$  or  $\mu$  value is used to re-evaluate the  $\chi^2$  at this grid-point. This approximation assumes that the model-error does not depend on  $d_{\text{scale}}$ . Also, prior to snana version v9\_93 the grid-search was over photoZ and color only.

Cheater option: OPT\_PHOTOZ = 16 sets initial photoZ to spectroscopic redshift and skips the redshift grid-search. This option is for debugging only.

### 5.10.3 Smooth Model Error Transition Across Filter Boundaries

For rest-frame models such as MLCS2k2, the model error changes abruptly when a photoZ variation results in an observer-frame filter mapping into a different rest-frame filter. This abrupt change in the model error causes a small kink in the  $\chi^2$ , and can cause fitting pathologies. This pathology is treated by smoothly transitioning the model error between  $\pm 200 \text{ \AA}$  of the transition wavelength ( $\bar{\lambda}$ ). The 200  $\text{\AA}$  range can be modified with the namelist parameter LAMREST\_MODEL\_SMOOTH. The transition weight-function is an arc-tangent that is scaled to equal 0 at  $\bar{\lambda} - 200$  and 1 at  $\bar{\lambda} + 200$ . (see function RESTFILT\_WGT for more info).

### 5.10.4 Don't Fool Yourself when PhotoZ-Fitting Simulations

When studying photoZ fits with simulations, avoid fooling yourself with simulations outside the valid wavelength range. If you use the default light curve model in the simulation, measurements outside the valid wavelength range are excluded, and therefore the fitter has the same "initialization" advantage in picking filters as using a spectroscopic redshift. For testing photoZ fits, a "wavelength-extended" model should be used as explained in §5.14. Even with a wavelength-extended model, you can fool yourself because the same model is used in both the simulation and in the fit; hence even if the extrapolated

<sup>9</sup>See EP\_FFSUM\_XXX variables in snlc\_fit.car.

model (i.e, below 3200 and above 9500 Å for MLCS2k2) is wrong in reality, it is by definition correct when fitting the simulation. This forced correctness of the extrapolated model means that the 1st fit-iteration using all of the filters is guaranteed to give good results. A better test is to fit with a light curve model that deviates from the simulated model in the extrapolated regions. The 1st fit-iteration should then produce biased photoZ estimates, and ideally the filter-exclusion cut will work well enough so that there is no bias after the 2nd fit-iteration.

## 5.11 Including the $\log(\sigma)$ Term in the $\chi^2$

The default fits minimize the usual function  $\chi^2 = \sum_i \Delta F_i / \sigma_i^2$  where  $\Delta F_i$  is the data-model flux-difference for observation  $i$ , and  $\sigma_i^2$  is the quadrature-sum of the measurement plus model errors. If the model error depends on one or more fit parameters, there is a namelist option (DOCHI2\_SIGMA = T/F) to add in the extra term,  $2\ln(\sigma_i)$ . To avoid adding or subtracting large values to the  $\chi^2$ , the actual term added is  $\Delta\chi_{\sigma_i}^2 = 2\ln(\sigma_i/\sigma_i^{\text{last}})$  where  $\sigma_i^{\text{last}}$  is the uncertainty from the previous fit-iteration. Typically  $\Delta\chi_{\sigma_i}^2$  adds  $\sim 1$  to the total  $\chi^2$ , but sometimes  $\Delta\chi_{\sigma_i}^2$  can be very large (positive or negative) if a fitted uncertainty undergoes a significant change between fit iterations. Fits with a large  $\Delta\chi_{\sigma_i}^2$  value can be rejected using the &FITINP namelist variable FITWIN\_CHI2SIGMA. Values of  $\Delta\chi_{\sigma_i}^2$  can be visually examined with

```
grep MINUIT fit.log | grep SIGMA | more
```

DOCHI2\_SIGMA=F by default for fits with a fixed redshift. Currently (snana v9\_30) the only model affected by setting DOCHI2\_SIGMA=T is SALT-II because the model-error depends on the stretch/shape parameter. For the default fits with DOCHI2\_SIGMA=F, the stretch parameter in the error term is fixed to the value from the previous iteration; it is therefore recommended to set NFIT\_ITERATION=3 for SALT-II fits.

For photoZ fits DOCHI2\_SIGMA=T is automatically set because the model errors change as the photoZ sweeps through different rest-frame epochs and wavelengths for each observation. If a filter is added after the first fit-iteration then  $\sigma_i^{\text{last}}$  is undefined and  $\Delta\chi_{\sigma_i}^2$  is set to zero for this filter. In this case, LREPEAT\_ITER is set to force a repeat fit-iteration with the added filter; this logic ensures that  $\Delta\chi_{\sigma_i}^2$  is defined for each filter in the last fit-iteration.

## 5.12 Optional Redshift Sources

Some surveys may include more than one redshift source such as spectroscopic redshifts from an external collaborator (e.g., BOSS redshifts for SDSS targets), or photometric redshifts obtained from different methods. While the “REDSHIFT\_FINAL” key specifies the nominal redshift, optional redshifts and associated typings can be used with the namelist variable

```
&SNLCINP
  ZEXTRA_SOURCE = 'ZZZ'
  ...
&END
```

where “ZZZ” is the name of the redshift source. The SNANA programs will then search each SN data file for the optional keywords

```
[ZZZ]_TYPE: nnn # optional
[ZZZ]_REDSHIFT_HELIO: 0.04592 +- 0.0002 (Helio) # optional
[ZZZ]_REDSHIFT_CMB: 0.04500 +- 0.0002 (CMB) # optional
[ZZZ]_END: # mandatory key
```

where “nnn” is the SN type based on using the optional redshift. The first three keys are optional, meaning that they only need to be specified in data files where such information exists. The “[ZZZ]\_END:” key is required in every data file; if this key is missing, the program will abort. Several different sets of optional redshifts can exist in each data file, but they must all go at the end of the header (just before the first observation).

These optional redshifts may be used only by a list of valid users specified in a global file called

```
more $SNDATA_ROOT/[SURVEY]/[ZZZ]_USERS.LIST
USER: <user1>
USER: <user2>
USER: <user3>
etc ...
```

This file-system is not a security system, but is only intended to prevent accidental mis-use.

### 5.13 Excluding/Downweighting Filters and Epoch Ranges

Here we discuss options to exclude or down-weight data in the light curve fitter. The filter selection is based on rest-frame wavelength so that a uniform cut can be applied to different filter systems. To globally exclude the rest-frame ultraviolet (UV) region, for example, set \$SNLCINP namelist variable

```
CUTWIN_RESTLAM = 3900. 20000.
```

which excludes filter(s) whose central wavelength satisfies  $\lambda_{\text{obs}}/(1+z) < 3900 \text{ \AA}$ . This option excludes data as if it were not part of the data file; hence the corresponding passband is not used for spectral warping,  $T_{\text{rest}}$  cuts, etc ...

One problem with the above option is that there is no way to extrapolate the model back to excluded filter and check data-model fit residuals. To visually monitor data-model residuals for the excluded region, it is better to simply down-weight rather than exclude a particular range in wavelength or epoch. A set of FUDGE\_FITERR\_XXX variables allow the user to down-weight arbitrary wavelength and epoch ranges. These &FITINP namelist variables are illustrated in the following example:

FUDGE_FITERR_TREST	= -20., 100.	! rest-frame range (days)
FUDGE_FITERR_RESTLAM	= 1000., .3900.	! wavelength range (A)
FUDGE_FITERR_PASSBANDS	= 'UBVRI'	! observer filters
FUDGE_FITERR_MAXFRAC	= 100.	! error -> 10*maxFlux

This example does essentially the same thing as the above example using “CUTWIN\_RESTLAM = 3900., 20000.” However, using the FUDGE\_FITERR\_XXX variables means that filters mapping onto the rest-frame UV region are used in the spectral warping for K-corrections, and these filters are also used in the  $T_{\text{rest}}$  sampling requirements. In evaluating the lightcurve fit- $\chi^2$ , an additional uncertainty of  $100\times$  the maximum flux (FUDGE\_FITERR\_MAXXFRAC=100) is added in quadrature to each epoch-uncertainty that satisfies the RESTLAM and TREST windows, and hence such epochs are effectively excluded from the fit. Note that FUDGE\_FITERR\_PASSBANDS specifies the observer-frame filters for which the other criteria apply. If you set the observer passbands to 'U', then the RESTLAM and TREST criteria are applied only for observer-U and not the other filters. One can therefore choose to down-weight data based on filters or based on rest-frame wavelength ranges.

Here is another example in which epochs before  $-5$  days and after  $+50$  days are excluded for all filters:

FUDGE_FITERR_TREST	= -99,-5.0, +50., 999.
FUDGE_FITERR_RESTLAM	= 1000., 20000.
FUDGE_FITERR_PASSBANDS	= 'UBVRI'
FUDGE_FITERR_MAXFRAC	= 100

Two sets of “RESTLAM” windows can be defined in the same that the two “TREST” window are defined above. To downweight all measurements more easily there are two global options,

FUDGEALL_MAXFRAC	= 0.3	# all epoch, all fit-iterations
FUDGEALL_ITER1_MAXFRAC	= 0.3	# all epochs, 1st fit-iter only

where the “ITER1” option inflates the errors only on the first fit-iteration. These options may be useful in cases where the data uncertainties are smaller than the model errors, resulting in unstable fits. Inflating the errors generally results in more stable fits. The “ITER1” option is intended to give a reliable initial-parameter estimate for the 2nd fit-iteration that uses the nominal uncertainties.

## 5.14 Rest-Frame Wavelength Range

Each SN model includes a function that returns the valid rest-frame wavelength range ( $\lambda_{\text{rest-range}}$ ) to the fitting program. There are two different ways to change the  $\lambda_{\text{rest-range}}$ , but note that you can only make the  $\lambda_{\text{rest-range}}$  *more restrictive*; i.e., you cannot arbitrarily loosen the  $\lambda_{\text{rest-range}}$  for a SN model. The two equivalent namelist options to change the  $\lambda_{\text{rest-range}}$  (Å) are

```
&SNLCINP
  CUTWIN_RESTLAM = 2000 , 25000
&END

&FITINP
  RESTLAMBDA_FITRANGE = 3500 , 9500
&END
```

The first option rejects measurements as part of the pre-fitting selection, and is useful if you want to apply this requirement without running the fit; i.e., using only the `snana.exe` program. The second option is applied during the fitting stage.

The  $\lambda_{\text{rest-range}}$ s appear in your log-file as follows,

```
CUTWIN_RESTLAM = 2000. 25000.
SN-MODEL LAMBDA RANGE: 3200. - 9500.
USER-FIT LAMBDA RANGE: 3500. - 9500.
```

and again note that the *most restrictive* range is used. If you specify a wide open range such as 1000 to 30000, this simply tells the fitting program to use the default range.

Finally, if you really insist on changing the default  $\lambda_{\text{rest-range}}$  for a particular SN model,<sup>10</sup> you can modify the default range by editing the file

```
$SNDATA_ROOT/models/[model-subdir]/RESTLAMBDA_RANGE.DAT
```

Such changes should be used with great caution because this change affects all users. Before making such a change, consider creating a private model-version (i.e., `m1cs2k2.LAM2800`).

---

<sup>10</sup>All maintenance warranties are null & void if you change the default  $\lambda_{\text{rest-range}}$ .

## 5.15 Extracting Light Curve Shape from the Fit

The light curve shape, defined as the flux-to-peakFlux ratio ( $F/F_0$ ) versus epoch, is determined after each light curve fit. In principle, this ratio is unity at the epoch of peak brightness. The residual-ntuple variable is FPKRAT (ntuple id 7799) and the fortran variables are EP\_FPKRAT and EP\_FPKRATERR. See fortran subroutine FPKRAT for example on how to access these arrays.

While the flux values are from the data, the estimate of the peakFlux is based on the best-fit model. The peakFlux estimate can be significantly off, particularly for multi-band light curves that fit one of the colors poorly. Such peakFlux errors are evident for light curves in which the the data points in a given filter lie well above or below the best-fit model. To get a better estimate of the peakFlux, one can correct for the average data/model ratio in a particular epoch-range. This correction is implemented using the \$FITINP namelist variable

```
TREST_PEAKRENORM = -10.0, +20. # rest-frame days
```

which specifies the rest-frame range for which to include epochs in the data/model correction. The default values are 0,0  $\rightarrow$  no correction. A data/model weighted-average is taken over the epochs within TREST\_PEAKRENORM. The weight ( $w_i$ ) at each epoch “ $i$ ” is defined to be

$$w_i \equiv \frac{1}{\sigma_i^2 \times (|T_{\text{rest}}| + 1)}, \quad (15)$$

where  $\sigma_i$  is the data/model flux-ratio uncertainty based on the data-flux error (i.e., ignores the model error), and “ $|T_{\text{rest}}| + 1$ ” is an arbitrary factor (in days) used to downweight epochs away from peak.

To see the peakFlux estimates on the light curve fit (§ 5.8), use the command

```
mkfitplots.pl --h <hisfile> PEAKFLUX
or
PAW > snana#fitres pkf=1
```

and a pink horizontal line is drawn through the peakFlux estimate for each filter and SN. The user is encouraged to vary TREST\_PEAKRENORM to check the sensitivity of the epoch range.

## 5.16 Landolt ↔ Bessell Color Transformations

As explained in the first-season SDSS–II results paper (Appendix B of arXiv:0908.4274), the nearby SNe Ia magnitudes are reported in the Landolt system, but there are no *UBVRI* filter responses for this system. We therefore define color transformations between the Landolt system and synthetic *UBVRI* magnitudes using Bessell (1990) filter response functions (See Eqs. B1-B2 in above reference). These color transformations can be implemented in the fitter with the `&FITINP` namelist flag

```
OPT_LANDOLT = 1 ! transform rest-frame Landolt model mags -> Bessell90
OPT_LANDOLT = 2 ! transform obs-frame Bessell90 mags -> Landolt
OPT_LANDOLT = 3 ! both of the above
```

For example, to analyze the JRK07 sample with MLCS2k2, set `OPT_LANDOLT=3`; to analyze with SALT-II, set `OPT_LANDOLT=2`. To analyze ESSENCE data with MLCS2k2, `OPT_LANDOLT=1`. In the last case, the ESSENCE *RI* filter response functions are well known, so there is no need to use Bessell90 filter responses. Note that the 2nd bit should be used only when the *UBVRI* filter response is not known. Thus, for example, do not set the 2nd bit for CFA3 & CFA4.

All of the above use BD17 as the primary reference. To use Vega, add 4 (3rd bit) to each `OPT_LANDOLT` value. You are also responsible for using the appropriate K-correction file with the matching primary reference.

## 5.17 Interpolating Fluxes and Magnitudes

There are two methods for interpolating the SN flux at a particular observation time (MJD). The first method is to use the generic 'any-LC' function from §5.4, and the second method is to use an SN Ia light curve model. For either method prepare a two-column file that lists each SNID and MJD to interpolate. To interpolate all SN at a particular MJD replace the SNID with the keyword 'ALL'. In the following example of a two-column input file,

```
1      53616.0
2      53610.0
2      53626.0
ALL    53640.0
```

one epoch is interpolated for SN 1, two epochs are interpolated for SN 2, and one epoch (53640) is interpolated for all SNe. Specify input and output files with the following &SNLCINP namelist variables,

```
SNMJD_LIST_FILE = 'KECK-SDSS.LIST'
SNMJD_OUT_FILE  = 'KECK-SDSS.OUT'
```

Errors on the interpolated fluxes account for the errors and covariances among the fitted parameters. The interpolated results are dumped into a human-readable output file with keywords to simplify parsing. To interpolate the flux at peak brightness, specify `MJD = 0` in the `SNMJD_LIST_FILE`.

The “any-LC” function is recommended in most cases since each filter is fit separately and since a more general class of light curve shapes can be accurately fit. Note that you must run `snana.exe` instead of `snlc_fit.exe`! The &SNLCINP namelist parameters are

```
OPT_SETPKMJD = 1 # fit with any-LC function
OPT_LCPLOT   = 1 # make plots for PAW > snana#fitres .
```

Filters can be ignored, for example, setting `EPCUT_SNRMIN = 'g 99999 r 99999'` to skip the *g* and *r* bands.

When fitting with an SNIa model (2nd method) it is recommended to fit a single passband by either specifying one passband with the &FITINP namelist variable `FILTLIST_FIT`, or by downweighting the other passbands using the `FUDGE_FITERR_XXX` options (§5.13). The latter is recommended because some SNIa models return garbage if only one filter is included. Thus, to interpolate the flux in each of the *griz* passbands, four separate fits should be done. The &FITINP namelist parameters to interpolate *g*-band by downweighting the other bands are as follows:

```
FILTLIST_FIT = 'griz'
FUDGE_FITERR_TREST      = -20. 80. 0. 0.
FUDGE_FITERR_PASSBANDS = 'riz'
FUDGE_FITERR_MAXFRAC   = 100.
```

Note that you can use command-line arguments (§12.2.1) to write a wrapper that loops over the filters,

```

snlc_fit.exe  myfit.nml  FUDGE_FITERR_PASSBANDS  riz  SNMJD_OUT_FILE  g.OUT
snlc_fit.exe  myfit.nml  FUDGE_FITERR_PASSBANDS  giz  SNMJD_OUT_FILE  r.OUT
snlc_fit.exe  myfit.nml  FUDGE_FITERR_PASSBANDS  grz  SNMJD_OUT_FILE  i.OUT
snlc_fit.exe  myfit.nml  FUDGE_FITERR_PASSBANDS  gri  SNMJD_OUT_FILE  z.OUT

```

## 5.18 Fitting Rest-Frame Peak-Magnitudes and Colors

There are two ways to define rest-frame peak magnitudes ( $M^*$ ): 1) from the best-fit model and 2) the true mag. While the former is trivial to obtain after fitting a light curve, the resulting rest-frame colors are simply pegged to the SNIa model and may not reflect variations from intrinsic scatter. The true  $M^*$  and associated colors include intrinsic variations. This section focuses on obtaining the true  $M^*$  and colors by specifying the following &FITINP options for `snlc_fit.exe`,

```

FILTLIST_FITRESTMAG = 'UBV'
LAMEXTRAP_FITRESTMAG = 250 ! (A) allow this much rest-frame extrapolation
LUMIFIX_FITRESTMAG = -0.2 ! => fix LUMIPAR for PEAKMAG calc

```

The `kcov/calibration` file must include these extra (UBV) filters in addition to the observer-frame filters. Each  $M_{U,B,V}^*$  is determined by fitting only the two nearest observer-frame bands that bracket the rest-filter in wavelength. `LAMEXTRAP_FITRESTMAG` allows wavelength slop in defining the observer-frame filters, and increases the redshift range for which all of the  $M_{U,B,V}^*$  can be determined. For example, suppose that the bluest observer-frame filter is *g* band with a central wavelength of  $\lambda_g = 4800 \text{ \AA}$ , and the rest-frame *U* band has  $\lambda_U = 3600 \text{ \AA}$ . If `LAMEXTRAP_FITRESTMAG`= 0 (default) then the minimum redshift for which  $M_U^*$  can be determined is  $z_{\min} = 4800/3600 - 1 = 0.333$ ; with `LAMEXTRAP_FITRESTMAG`= 250  $\text{\AA}$  we have  $z_{\min} = 4800/(3600 + 250) - 1 = 0.247$ . Similar logic is applied to the reddest filter.

The fitting for each SN proceeds as follows. The first “NFIT\_ITERATION” fit-iterations are used to determine the time-of-peak ( $t_0$ ) and stretch parameter ( $s_0$ ) from a fit to all of the observer-frame bands. One additional fit-iteration is then performed for each  $M^*$  using only the two nearest bands, holding  $t_0$  and  $s_0$  fixed from the global fit. The floated color and distance parameters provide the flexibility to fit both observer-frame bands. After each two-band fit  $M^*$  is computed as follows,

$$M_X^* \equiv \text{modelMag}(T_X, t_0, s_0, \text{color}, \text{distance}) - \mu_z \quad (16)$$

where  $T_X$  is the filter-transmission for filter  $X = U, B, V$ , and  $s_0$  is replaced by `LUMIFIX_FITRESTMAG` if this namelist parameter is set.  $\mu_z$  is the calculated distance modulus using the known redshift and an assumed cosmology. The subtraction of  $\mu_z$  is done so that the  $M_X^*$  have reasonable values ( $\sim -19$ ), but it has no impact on the rest-colors since the same  $\mu_z$  is subtracted for each (UBV) filter. After all of the two-band fits are done, a final fit-iteration is performed using all of the filters so that the analysis variables (`Ndof`, `SNRMAX`, `FITPROB`, etc ...) correspond to the global fit.

The rest-mags are written to the `fitres`-file as `M0_X` and `ERRM0_X` where *X* are the rest-filter character names (U,B,V). &SNLCINP namelist variable `CUTWIN_SNRMAX2` is applied to the maximum S/N for each filter in each 2-band fit.

## 5.19 Selecting Telescope, Field and SNe

Here are example namelist options to select a telescope, field and SNe to process:

```
SNTEL_LIST      = 'SDSS'           ! list of telescopes
SNFIELD_LIST    = '82S' , '82N'    ! list of fields to process
CUTWIN_NFIELD   = 1, 99           ! number of overlapping fields

CUTWIN_CID      = 700, 2000        ! Cand-ID range to process
SNCID_LIST      = 5944, 10550      ! list of SN to process
SNCCID_LIST     = '5944', '10550'  ! same as above

SNCID_IGNORE    = 4524, 8151, 7017 ! list of SN to ignore
SNCCID_IGNORE   = '4524', '8151', '7017' ! same as above
```

The list of valid telescopes and fields is in `$SNDATA_ROOT/SURVEY.DEF`, and you can select multiple telescopes and fields as illustrated above with `SNFIELD_LIST`. If you do not specify `SNTEL_LIST` or `SNFIELD_LIST`, then all telescopes or fields are processed by default; therefore, use these options only if you want to select a subset. If you select an invalid telescope or field, the code aborts. `CUTWIN_NFIELD` selects the number of overlapping fields. For example, to select SDSS SNe that overlap both 82N & 82S, set `CUTWIN_NFIELD = 2, 2`; i.e., require two overlapping fields.

The principle method for selecting SNe is the namelist variable `CUTWIN_CID` (`CID` = Candidate ID). If the SNe are identified by integers, as in the SDSS-II survey, then `CUTWIN_CID` simply selects the `CID` range. If the SNe are identified by character strings, then they are given internal `CID` values 1,2, ... up to the number of SNe. If you have 100 SNe defined as character strings, then set `CUTWIN_CID = 1, 100`. You can always open this window (1, 100000) to ensure that all SNe are processed. In addition to the `CID` range, you can specify a specific list of SNe to process (`SNCID_LIST` and/or `SNCCID_LIST`) and a specific list of SNe to ignore (`SNCID_IGNORE` and/or `SNCCID_IGNORE`). Both of these lists can be specified as integer or string. If SNe are specified with `CUTWIN_CID` and a list, then all specified SNe are processed; i.e., the logical-AND of all SN-selection. To process only those SNe in the `SNCID_LIST`, you must explicitly set `CUTWIN_CID = 0,0`.

In the “\_LIST” variables, a zero or blank acts as a terminator. For example, `SNCID_LIST = 5944, 0, 1032, 10550` would process SN 5944 and ignore the rest.

### 5.19.1 Quickly Analyzing a few SNe from a Large Sample

This section is for the text-output option only (`FORMAT_MASK = 1` or `2`).

It is often useful to select a few SNe for fitting and analysis, such as for debugging a particular problem. From §5.19 above, the namelist variable `SNCID_LIST` can be used to select an arbitrary subset, but the `snana.exe` and `snlc_fit.exe` programs must read every SN data file up to the point where each SN in the list has been found. The time to simply read these data files can be relatively slow (~ minute) if a large number of data files must be screened. To read in the data files more quickly, a temporary `DEBUG_XXX` version can be created where `XXX` are your initials or some other identifier. Just three files need to be created:

```
$SNDATA_ROOT/lcmerge/DEBUG_XXX.README
$SNDATA_ROOT/lcmerge/DEBUG_XXX.IGNORE
$SNDATA_ROOT/lcmerge/DEBUG_XXX.LIST
```

and for simulations replace “lcmerge” with “SIM” and create a sub-directory SIM\_DEBUG\_XXX with the appropriate files. The IGNORE and README files can be blank. The LIST file contains the name of each data file to analyze. You can then analyze this debug version by specifying \$SNLCINP namelist variables

```
VERSION_PHOTOMETRY = 'DEBUG_XXX'
CUTWIN_CID = 1, 100000
```

and there is no need to specify SNCID\_LIST.

## 5.20 Mag-Shifts in Zero Points and Primary Reference Star

The `snlc_fit.exe` program provides an interface to modify zero-point offsets as well as the magnitudes of the primary reference star. The primary magnitudes can be adjusted using the `&SNLCINP` namelist option

```
MAGOBS_SHIFT_PRIMARY = 'B .02 V .02'
MAGREST_SHIFT_PRIMARY = 'B .02 V .02'
```

which shifts the primary mags by 0.02 for *B* and *V* in the example above. Note that separate strings are used for the observer-frame and rest-frame to allow for the same filter-character to be used in each reference frame. This option is equivalent to re-generating the K-correction tables with these shifts, but the `MAG[OBS,REST]_SHIFT_PRIMARY` option is much quicker since you can use the same K-correction tables.

For the zero-points, there are two ways to introduce shifts. The first method is to specify the offsets in a file, `ZPOFF.DAT`, located in the `filters` sub-directory. For the SDSS AB-offsets, the file location is

```
> more $SNDATA_ROOT/filters/SDSS/ZPOFF.DAT
u -0.037 g 0.024 r 0.005 i 0.018 z 0.016
```

If `ZPOFF.DAT` does not exist, the shifts are zero. This method is used for standardized shifts.

The second option is designed to probe the uncertainty in the zero-point offsets; an arbitrary shift can be specified with the `&SNLCINP` namelist string

```
MAGOBS_SHIFT_ZP = 'g .02 r .02 i .02'
```

Note that the shifts in `ZPOFF.DAT` and `MAGOBS_SHIFT_ZP` are added so that you make well-defined shifts relative to the standard shifts in `ZPOFF.DAT`.

## 5.21 Fudging the FLUXCAL Offsets and Uncertainties

Filter-dependent offsets and errors can be added to the calibrated fluxes using the following &SNLCINP namelist variables

```
FUDGE_FLUXCAL_OFFSET = 'u -0.6 g 0.4 r 1.2 i -3.7 z 2.2'  
FUDGE_FLUXCAL_ERROR  = 'u 24 g -12 r 44 i 19 z -22'
```

These fudges should be used for systematic studies only; permanent changes should be made in the data files. The error fudges are added/subtracted in quadrature based on the sign of FUDGE\_FLUXCAL\_ERROR.

## 5.22 Updating the Filter Transmission for each SN

In 2009 the SNLS reported that their filter transmission depends on the focal plane position.<sup>11</sup> While their transmission function depends only on the radius from the center, in general the SN filter transmission can be unique for each SN. Using the SNANA fitting program, an SN-dependent filter transmission can be specified, although it is assumed that each transmission function is the same for all epochs. This feature is invoked by specifying an 'update' directory from the \$SNLCINP namelist as follows:

```
FILTER_UPDATE_PATH = 'MYPATH'
```

MYPATH can be a subdirectory under \$SNDATA\_ROOT/filters, or MYPATH can be the full directory name; both directories are checked, with the former having priority. This directory must contain an instruction file called 'FILTER.INFO', along with a filter transmission text-file for each SN. Rather than giving an explicit list of filter-transmission filenames, the INFO file specifies the naming conventions for the filter-transmission directories and files:

```
FILTER_SUBDIR: filters-{SNID}  
FILETRANS_SN: SNtrans_*.dat  
FILETRANS_REF: REFtrans_*.dat
```

The first key specifies the sub-directory name for each set of filter transmissions, and the directory name depends on the name of each SN. The next two keys specify the transmission filenames residing within each FILTER\_SUBDIR. The star (\*) represents the 1-letter representation for each filter. For example, the SNLS filenames would be SNtrans\_g.dat, SNtrans\_r.dat, etc. The filter transmission can be different for the SN and for the primary references (REF). However, if only one set of transmissions is specified (either SN or REF), then these transmissions are used for both the SN and the primary reference.

**WARNING:** currently this filter-update feature works only for the SALT2 model; perhaps a future SNANA version will work for rest-frame models that use K-corrections.

---

<sup>11</sup>Regnault et al., *A&A* **506**, 999 (2009).

## 5.23 Analysis Ntuples

Analysis variables and fit-parameters are reported in several different CERNLIB ntuples set by the following namelist flags:

```
&SNLCINP
  LTUP_SNANA = T      ! 7100: snana analysis variables
  LTUP_SKY   = F      ! 7101: moon, PSF, noise vs. filter and epoch
  LTUP_PHOTOMETRY = F ! 7200: photometry for all epochs & filters
&END

&FITINP
  LTUP_FITRES = T ! 7788: analysis variables + fit-parameters
  LTUP_RESIDUAL = T ! 7799: data-model residuals vs. epoch & filter
&END
```

The default values (T or F) and ntuple id-numbers are given above. The SNANA ntuple (7100) is limited, but is useful to quickly study selection criteria without running fits (i.e., setting NFIT\_ITERATION = 0 in the &SNLCINP namelist). Ntuple 7788 is the main analysis-ntuple with more than 100 variables, and ntuple 7799 is used to study light curve residuals, and to correlate flux-residuals with epoch, passband, PSF, sky-noise, photometry flag, etc ... To extract ntuple variable into a text file, see §12.1.2. When viewing the residual ntuple (7799) make sure to require USEFIT=1 to see the data-residuals; USEFIT=0 corresponds to artificial epochs at peak brightness.

The fit-analysis ntuple (7788 above) includes the following &FITNML options,

```
OPT_NTUP_FITRES = 1 ! include accepted + rejected SN
OPT_NTUP_FITRES = 2 ! include only fitted filters (default= all filters)
OPT_NTUP_FITRES = 3 ! both options above
```

By default the fitres-ntuple includes only those SNe that pass the selection criteria, and all filters are included even if a subset is selected for fitting. Beware that the fitres-ntuple is NOT case sensitive, and therefore ntuple variables such as Nepoch\_U and Nepoch\_u cannot be distinguished.

## 5.24 Analysis Alternative for those with HBOOK-phobia

The snana variables and fitter results are packed in ‘HBOOK’ histograms and ntuples. At the end of each job, these histograms and ntuples are written into a single HBOOK file, more commonly known as a “histogram” file (hence the common extension, “.his”). The utilities to create and access histogram files are part of CERNLIB. Fortran routines are available to access histogram contents from a compiled program, and PAW<sup>12</sup> is a well-known interactive program to analyze the contents of an HBOOK file, and to make plots. If you have little or no knowledge of HBOOK and PAW, and you are terrified at the prospect of being forced to learn these ancient (i.e., 20<sup>th</sup> century) analysis tools, this section explains some

---

<sup>12</sup>PAW = Physics Analysis Workstation

simple alternatives on how to analyze the fitter outputs by simply translating the ntuple contents into column-formatted text files.

First, to see each light curve, best-fit model, and fit-parameters, run this script,

```
> mkfitplots.pl --h snlc_fit.his
```

which creates one postscript file showing the light curve fit and results for each SN, and another postscript file showing the marginalized distributions for each fit-parameter.

Analysis variables are stored in ntuples (§5.23) which can be dumped into text files as explained in §12.1.2.

## 5.25 Monitoring Fit-Jobs with “grep”

If you pipe the fitter screen dump to a log file, the unix “grep” command can be used to quickly monitor progress during the fits, as well as after the fits have completed. This is particularly useful when many fit-jobs are run in parallel so that you can monitor progress and catch aborts. Many screen outputs are explicitly designed to help monitor the job status. Below are some examples of useful grep-commands to run. A dagger ( †) indicates those commands that work only when the fit-job has finished. Note that adding “| wc” to the end of a grep command will count the number of entries.

- `grep GRACE fit*.log †`  
lists the unique string ENDING PROGRAM GRACEFULLY to identify and count jobs that have finished.
- `grep " ABORT " fit*.log †`  
identifies jobs that have aborted. Note the blank space before and after ABORT to distinguish from namelist variables that include ABORT as part of the name.
- `grep "after snana" fit*.log †`  
shows the number of SN before and after SNANA cuts.
- `grep "after fit" fit*.log †`  
shows the number of SN after fitter cuts.
- `grep "PROCESS TIME" fit*.log †`  
shows total processing time.
- `grep "PASSES FIT" fit*.log`  
lists each SN that passes fit cuts.
- `grep "Cuts REJECT" fit*.log`  
lists each SN rejected by SNANA cuts.
- `grep "FAILED FIT" fit*.log`  
lists each SN rejected by fitter cuts.

- `grep ":DLMAG" fit*.log | grep pdf`  
lists the marginalized luminosity distance for each fitted light curve. Works for any fit-parameter: AV, DELTA, PEAKKMJD, PHOTOZ. Don't forget to include the colon, or you will be overwhelmed by the screen dump.
- `grep ":DLMAG" fit*.log | grep fitpar`  
lists the minimized luminosity distance for each fitted light curve.
- `grep MARGINALIZATION fit*.log | grep TOTAL`  
`grep MARGINALIZATION fit*.log | grep PRIOR`  
`grep MARGINALIZATION fit*.log | grep DATA`  
returns marginalized  $\chi^2$ ,  $N_{dof}$  and fit-probabilities for each SN.
- `grep "MINUIT MIN" fit*.log | grep TOTAL`  
`grep "MINUIT MIN" fit*.log | grep PRIOR`  
`grep "MINUIT MIN" fit*.log | grep DATA`  
returns MINUIT-minimized  $\chi^2$ ,  $N_{dof}$  and fit-probabilities for each SN.

## 5.26 User SN Tags

User-defined SN tags can be specified with the `&SNLCINP` namelist variable

```
&SNLCINP
  USERTAGS_FILE = 'mytags.dat'
  ...
&END

more mytags.dat
SN: 1032 1
SN: 2033 1
SN: 2490 1
SN: 3088 2
etc ...
```

and these USERTAG indices will appear in the analysis ntuples. These tags may be useful, for example, to label SNe confirmed by a particular telescope.

## 6 Private Options

### 6.1 Private \$NON1A\_ROOT

Instead of using the public \$SNDATA\_ROOT for the non-Ia templates, a private “\$NON1A\_ROOT” area can be used for developing non-Ia templates, or using proprietary templates. This \$NON1A\_ROOT directory is essentially a duplicate of \$SNDATA\_ROOT, but includes only the information needed to simulate non-Ia SNe. Depending on whether you choose the “NON1A” or “non1a” model, the required private directories are

```
mkdir $NON1A_ROOT/snsed/NON1A # for ``GENMODEL: NON1A``
mkdir $NON1A_ROOT/snsed/non1a # for ``GENMODEL: non1a``
mkdir $NON1A_ROOT/kcor/non1a # idem
mkdir $NON1A_ROOT/filters # idem
mkdir $NON1A_ROOT/standards # idem
```

The NON1A model needs only one directory containing the SEDs, while the non1a model needs more directories to generate and store the K-correction tables. For the non1a model, the K-correction tables must be generated as described in §4.5.1.

You can specify \$NON1A\_ROOT either by setting an environment variable in your session, or by adding the sim-input key

```
NON1A_ROOT: /my_private_non1a_dir
```

The sim-input key above takes priority over the value of the environment variable.

### 6.2 Creating Your Private Fitter: “snlc\_fit\_private.exe”

Here we describe a simple way to add your own fortran code into the snana program or fitter, such as writing out specific information to a text file, calculating a quantity that is not available, or testing modifications to the public code. There are two steps to creating your own private executable:

```
> cp $SNANA_DIR/src/snana_private.cra .
> snmake.pl snana_private
    or
> cp $SNANA_DIR/src/snlc_fit_private.cra .
> snmake.pl snlc_fit_private
```

should result in a private executable file in your directory: snana\_private.exe or snlc\_fit\_private.exe. Now run your private version,

```
./snana_private.exe myinput.nml
    or
./snlc_fit_private.exe myinput.nml
```

You can edit your private version of `snlc_fit_private.cra` and modify `USRINI`, `USRANA` and `USREND`. The sample `USRANA` shows how to access fit results, and how to access information for each epoch used in the fit. You can also re-name the code to any other name, such as ‘`myfit.cra`’, and then compile an executable with the command “`snmake.pl myfit`” to produce the executable file `myfit.exe`.

In addition to adding private code into the `USR[INI,ANA,END]` routines, you can also modify the official public code. Copy any subroutine from `SNANA_DIR/src/snana.car` or `snlc_fit.car`, paste it into your private `snlc_fit_private.cra`, and then make modifications. Once these changes are fully tested, you can request that the modified routine(s) be installed into the next public release of `SNANA`. Your changes may be included as the default, or they may be available to other users via input namelist flags. Note that modifications can also be made by checking out the entire `SNANA` product from CVS. You are welcome to check code out of CVS (`cvscv`), but please do NOT check code in without contacting the `SNANA` manager. Working with `snlc_sim_private.cra` should be much easier than working with the entire `SNANA` product.

#### WARNINGS:

- Do not trap yourself into an obsolete version of `SNANA` if you have lots of private code that is not integrated into the public `SNANA`.
- If you find yourself doing lots of cutting & pasting as part of your analysis, this is a bad sign: ask for help!
- If you find that you are doing lots of tedious/redundant operations, ask for help. Often adding just a few lines of code into `SNANA` can greatly simplify your analysis procedure.

### 6.3 Private Data Path: `PRIVATE_DATA_PATH`

After creating or translating a new version of light curves, it is useful to run tests before copying these files to the public/official area `SNANA_ROOT/lcmerge`. `SNANA` and fitter jobs can read data from a private directory as follows:

```
&SNLCINP
  VERSION_PHOTOMETRY = 'myVersion'
  PRIVATE_DATA_PATH  = 'myDir'
  .
  .
&END
```

The version “`myVersion`” will be read from “`myDir`” in exactly the same way that it would have been read from `SNANA_ROOT/lcmerge`. Users are cautioned to use this feature only for testing, and not as long-term data storage for your analysis.

## 6.4 Private Model-Path: \$SNANA\_MODELSPATH

For a given SN model in the simulation (GENVERSION) or in the fitter (FITMODEL\_NAME), the model parameters are assumed to reside in

```
$SNANA_ROOT/models/SALT2/*
$SNANA_ROOT/models/mlcs2k2/*
$SNANA_ROOT/models/snoopy/*
$SNANA_ROOT/models/SIMSED/*
etc ...
```

While these public directories are intended for stable models, a private “setenv SNANA\_MODELSPATH” directory can be defined for testing models that are not suitable for public release. If this user-defined environment variable exists, then the model is assumed to be under this path. For example, SALT2.Guy07 would be read from

```
$SNANA_MODELSPATH/SALT2.Guy07
```

## 6.5 Private Variables in Data Files

Private variables can be included in data file headers as illustrated by the following example,

```
PRIVATE (HOST_PROPERTY) : 43.22
PRIVATE (PHOTOFLAG) : 439
PRIVATE (SEARCH_TYPE) : 27
```

These ‘PRIVATE’ keys must appear in the header before the light curve (MJD) observations. These variables are included in the FITS-translated data files (§12.4.5). Using a private fitter option (§6.2), the value of any private variable can be accessed from the function

```
REAL*8 GET_PRIVATE_VALUE ! declare function
DVAL = GET_PRIVATE_VALUE(varName,0) ! do not abort on error
DVAL = GET_PRIVATE_VALUE(varName,1) ! abort on error
```

where varName is the name of any private variable. Note that varName can be simply HOST\_PROPERTY or it can be ‘PRIVATE(HOST\_PROPERTY)’. For public data releases we recommend NOT including private variables, although new standard variables can be added along with code to use them.

## 7 Adding a New Survey

Starting with `SNANA v6_00` you can add a new survey without modifications to the software. Primary SEDs, primary magnitudes and filter transmissions are defined via K-correction files, even for models like SALT-II that do not use K-corrections (but still use primary SEDs and magnitudes). A filter is defined by a single character to simplify the handling of a wide variety of output variable names that append the filter string (i.e, `MAGT0_[filter]`), and to simplify output formatting. While the `SNANA` filter definition is limited to the 1-character strings above, arbitrary filenames can be used to define the filter transmissions. There are 62 allowed filter-characters: A-Z, a-z, and 0-9. `SNANA` versions prior to `v9_00` allowed only the legacy filters *ugriz*, *UBVRI*, *YJHK*, along with 0-9. Additional filter-characters will be allowed when we all switch to using Chinese keyboards. Here are the steps for adding a new survey:

1. Add your survey and telescope to the file `$SNANA_ROOT/SURVEY.DEF`. Check if your survey and/or telescope is already defined before making changes.
2. Add new filters in `$SNANA_ROOT/filters`. The wavelength spacings for the filter transmissions must be uniform. If the wavelength spacings are large (several hundred Å) you should prepare a finer-binned filter set using an appropriate interpolation algorithm.
3. Generate K-correction tables using `kcor.exe`,<sup>13</sup> and see §7.1 for rules about filter names. You can leave your private K-correction table(s) in your directory where you run other jobs, or you can share official K-correction tables in `$SNANA_ROOT/kcor/`. For initial testing, use a very course grid so that the tables are built quickly. Once the simulation and fitter are working, you can generate the K-correction tables with a finer grid. The grid is controlled by `REDSHIFT_BINSIZE` and `AV_BINSIZE`. **WARNING:** you must generate a K-correction file even if you plan to run an observer-frame fitter such as SALT-II that does not use K-corrections; in this case do “`kcor.exe mykcor.input SKIPKCOR`” so that the K-corrections are skipped, but the filters and primary SED are included. Finally, for rest-frame models that use K-corrections, there is a limit of 10 rest-frame filters and 62 observer-frame filters. For observer-frame models such as SALT-II, the limit is 62 observer-frame filters.
4. If you want to generate simulated samples, you need to prepare a simulation library. See examples in `$SNANA_ROOT/simlib`. This is usually the most difficult part of setting up a new survey.
5. Check for an adequate rest-frame model to describe the supernova light curve.
6. If you plan to add new data files, use an existing data version as a template. To see existing versions do “`cd $SNANA_ROOT/lcmerge ; ls *.README`”. The minimal information needed is shown in versions `ESSENCE_WV07` & `SNLS_Ast06`; the maximum information (for systematic studies) is shown in version `SDSS_HOLTZ08`. The light-curve fitter uses `FLUXCAL= 10(11-0.4m)`. The scale-factor of  $10^{11}$  is arbitrary; you can change it, but then your distance moduli will all have a common offset, although the final cosmological parameters are not affected by the choice

---

<sup>13</sup>Sample `kcor-input` files are in `$SNANA_ROOT/analysis/sample_input_files/kcor`.

of scale-factor. Use a well-measured data point in each filter to get the FLUXCAL/FLUX ratio, and then convert FLUX  $\rightarrow$  FLUXCAL for each measurement. If you try to convert magnitudes to FLUXCAL, you will have problems for small & negative fluxes.

7. To distribute survey-dependent SNDATA\_ROOT files among collaborators working on different computer systems, see §12.2.2.
8. Modify a sim-input and fitter-input file by substituting your survey and filters. Good luck. And don't forget to send me a post-card about your experience.

## 7.1 Filter Names and Rules for K-corrections

The filter names defined in the K-correction input file can be anything, but only the last character (A-Z,a-z,0-9) is propagated into the simulation and fitting program. Thus, the following filter-names are all translated into 'g': SDSS-g, SDSSg, SDSS2.5m-g.

Filters are identified and stored separately for rest-frame and observer-frame. The reference frame is implicitly defined by KCOR entries such as

```
KCOR:  Bessell-B   SDSS-g   K_Bg
```

which defines *B* as a rest-frame filter and *g* as an observer-frame filter. If a filter is not used in a KCOR entry (such as for the SALT-II model), then it is assumed to be an observer-frame filter.

Each rest-frame filter must have a unique last character, and each observer-frame filter must have a unique last character; however, the same last character can be used in both the rest and observer frames. For example, consider filter sets CSP-[ugri] and SDSS-[ugri]. These two sets cannot both be defined as observer-frame filters in the same K-correction file, but one set can be used for rest-frame filters that describe a light curve model, and the other set can be used for the observer-frame. The K-corrections defined after the "KCOR:" keyword define which filters are used for rest/observer-frame.

Consider the following example that uses the same filter character 'B'.

```
MAGSYSTEM:  VEGA      (define mag system for filters below)
FILTSYSTEM:  COUNT
FILTER:      ACS_WFC_F435W-B      ACS_WFC_F435W.dat
etc ...
```

```
MAGSYSTEM:  VEGA
FILTSYSTEM:  ENERGY  ('ENERGY' => Trans -> Trans/lambda)
FILTER:      Bessell-B   Bessell190_B.dat   0.021
etc ...
```

If no K-corrections are defined, then *B* is defined twice as an observer-frame filter and the `snlc_fit.exe` fitting program will abort because of the ambiguity. However, if a K-correction is defined using Bessell-B as a rest-frame filter, then ACS\_WFC\_F435W-B is the unambiguous observer-frame *B* band filter.

## 8 Photometric Classification: `psnid.exe`

A separate program called “`psnid.exe`” (Photometric SN id) performs photometric classification using light curve templates. This program has the same `&SNLCINP` namelist as the `snana.exe` and `snlc_fit.exe` programs so that reading light curves and applying selection requirements is done the same way. Instead of defining a `&FITINP` namelist for the `snlc_fit.exe` program, there is a `&PSNIDINP` namelist with declarations and definitions in `$SNANA_DIR/src/psnid.car`. The `psnid.exe` architecture allows for arbitrary methods based on SNIa and CC templates. The templates are created with the simulation program (`snlc_sim.exe`) as described in §4.27. The current `psnid.exe` method is based on [9], and is called “Bayesian Evidence with Supernova Templates” (BEST); other methods can be added into `psnid.exe` using either C or fortran functions. Example namelist files are in

```
$SNDATA_ROOT/analysis/sample_input_files/psnid
```

and the main namelist keys are as follows

```
&PSNIDINP
METHOD_NAME      = 'BEST'
FILTLIST_FIT     = 'griz'
OPT_ZPRIOR       = 0          ! 0=flat, 1=Zspec, 2=Zphot(HOST)
FITRES_DMPFILE   = 'PSNID_DES.fitres' ! text-output (one row per SN)
PRIVATE_TEMPLATES_PATH = 'myDir' ! optional private path for templates
TEMPLATES_SNIa   = 'GRID_DES_SALT2.FITS'
TEMPLATES_NONIa  = 'GRID_DES_NONIA.FITS'
etc ...
```

The default template location is `$SNDATA_ROOT/models/psnid` unless `PRIVATE_TEMPLATES_PATH` is specified. In addition to these control keys, there are many variables (see `psnid.car`) to control the detailed behavior of the classification algorithm.

The multi-core distribution script (§12.4.1) can be used in the same manner as for `snlc_fit.exe` by simply specifying an additional key

```
JOBNAME_LCFIT:  psnid.exe
```

at the top of the namelist file.

## 9 Light Curve Models

Here we discuss some of the available light curve models for the simulation and fitter. This is only brief a technical discussion on how to use the models in SNANA; a reference for each model is provided for more details. For each model “MMM” there is a dedicated model directory,

```
$SNDATA_ROOT/models/MMM
```

containing one or more versions of the model. The file `MMM.default` points to a default version if a generic model name (e.g., `SALT2`, `mlcs2k2`, `snoopy`) is specified. Any model version can also be selected. Examples of model selection are

```
GENMODEL:      MMM      # sim-input; use what's in MMM.default
GENMODEL:      MMM.v01  # sim-input; use this version

FITMODEL_NAME = 'MMM'      ! fit-input; use what's in MMM.default
FITMODEL_NAME = 'MMM.v02' ! fit-input; use this version
```

In the sub-sections below, “xxx” refers to a floating point number that must be specified by the user. In general, each simulated parameter “XXX” is specified by three input keys: 1) `GENMEAN_XXX`, 2) `GENRANGE_XXX`, and 3) `GENSIGMA_XXX`. The `GENSIGMA` key has two values to specify an asymmetric (or symmetric) Gaussian distribution. The `GENRANGE` values truncate the distribution.

## 9.1 MLCS2k2

Reference: Jha, Riess, Kirshner, **AJ 659**, 122 (2007).

Simulation input keys for `snlc_sim.exe` :

```
GENMEAN_DELTA:   xxx           # shape/luminosity parameter
GENRANGE_DELTA:  xxx  xxx
GENSIGMA_DELTA:  xxx  xxx

GENMEAN_RV:      xxx           # CCM89 dust parameter
GENRANGE_RV:     xxx  xxx
GENSIGMA_RV:     xxx  xxx

GENRANGE_AV:     xxx  xxx      # CCM89 V-band extinction
GENTAU_AV:       xxx           # dN/dAV = exp(-AV/xxx)
```

&FITINP namelist variables for `snlc_fit.exe` :

```
OPT_SNXT         = 1    ! Use CCM89 + ODonnell94 update
SCALE_COVAR      = 4.1  ! scale cov matrix
OPT_LANDOLT      = 1    ! 1=>transform Bessell90 <=> Landolt with color transf.
                  ! 3=> same for mlcs model & nearby-Landolt mags

OPT_PRIOR_AV     = 1    ! 0=> switch off AV prior
NGRID_PDF        = 11   ! marginalize NGRID per variable (0 => fit min only)
NSIGMA_PDF       = 4    ! initial guess at integration range: +- 4 sigma
OPT_SIMEFF       = 1    ! use simulated eff as part of prior
PRIOR_AVEXP     = 0.3   ! tau of exponential AV prior
PRIOR_AVRES     = 0.005 ! smooth Gauss rolloff for AV<0.
PRIOR_MJDSIG    = 10.   ! Gauss prior on MJD at peak

INISTP_RV        = xxx   ! 0 => fix RV to INIVAL_RV
INIVAL_RV        = 2.2   !
INISTP_AV        = xxx   ! 0 => fix AV = INIVAL_AV in fit; else float
INIVAL_AV        = xxx
INISTP_LUMIPAR   = xxx   ! 0 => fix DELTA to INIVAL_LUMIPAR; else float
INIVAL_LUMIPAR   = xxx

PRIOR_DELTA_PROFILE = xxx, xxx, xxx, xxx ! grep snlc_fit.car for details
```

## 9.2 SALT-II

Reference: J. Guy et al., *A&A* **466**, 11 (2007).

Simulation input keys for `snlc_sim.exe` :

```
GENMEAN_SALT2x1:   xxx           ! stretch parameter
GENRANGE_SALT2x1:  xxx   xxx
GENSIGMA_SALT2x1:  xxx   xxx

GENMEAN_SALT2c:    xxx           ! color parameter
GENRANGE_SALT2c:  xxx   xxx
GENSIGMA_SALT2c:  xxx   xxx

# mag = mB* + alpha*x1 - beta*color
GENMEAN_SALT2BETA: 3.2   ! color coeff
GENSIGMA_SALT2BETA: 0 0   ! default is no beta smearing
GENRANGE_SALT2BETA: 0 5   ! restrict if SIGMA is non-zero

GENMEAN_SALT2ALPHA: 0.11   ! x1 coeff
GENSIGMA_SALT2ALPHA: 0 0   ! default is no alpha smearing
GENRANGE_SALT2ALPHA: 0 .3  ! restrict if SIGMA is non-zero
```

&FITINP namelist variables for `snlc_fit.exe` :

```
SALT2alpha   = xxx   ! used only to compute mu - muref
SALT2beta    = xxx   ! idem

INISTP_COLOR = xxx   ! 0 => fix color to INIVAL_COLOR; else float
INIVAL_COLOR = xxx

INISTP_LUMIPAR = xxx ! 0 => fix x1 to INIVAL_LUMIPAR; else float
INIVAL_LUMIPAR = xxx

PRIOR_MJDSIG = xxx           ! Gaussian prior on MJD at peak
PRIOR_LUMIPAR_RANGE = xxx, xxx ! flat prior on x1 (prevents crazy values)
PRIOR_LUMIPAR_SIGMA = xxx     ! Gauss rolloff at edges of flat prior

SALT2_DICTFILE = 'xyz' ! output formatted for original hubblefit

OPT_COVAR = 0 ! 0 => COV-diag only
            ! 1 => COV fixed during fit
            ! 2 => COV re-calculated for each chi2 in fit
```

Note that the `INISTP_XXX` and `INIVAL_XXX` parameters are specified only to fix these parameters in the fit. If not specified, these parameters are floated in the fit.

### 9.3 SNooPy

Reference: C. Burns et al., arXiv:1010.4040.

Simulation input keys for `snlc_sim.exe` :

```
GENMEAN_DM15:   xxx           # shape/luminosity parameter
GENRANGE_DM15:  xxx  xxx
GENSIGMA_DM15:  xxx  xxx
GENMEAN_RV:     xxx           # CCM89 dust parameter
GENRANGE_RV:    xxx  xxx
GENSIGMA_RV:    xxx  xxx
GENRANGE_AV:    xxx  xxx     # CCM89 V-band extinction
GENTAU_AV:      xxx           # dN/dAV = exp(-AV/xxx)
```

&FITINP namelist variables for `snlc_fit.exe` :

```
OPT_PRIOR_AV    = 1    ! 0=> switch off AV prior
NGRID_PDF       = 11   ! marginalize NGRID per variable (0 => fit min only)
NSIGMA_PDF      = 4    ! initial guess at integration range: +- 4 sigma
OPT_SIMEFF      = 1    ! use simulated eff as part of prior
PRIOR_AVEXP     = 0.3  ! tau of exponential AV prior
PRIOR_AVRES     = 0.005 ! smooth Gauss rolloff for AV<0.
PRIOR_MJDSIG    = 10.  ! Gauss prior on MJD at peak
PRIOR_LUMIPAR_RANGE = 0.7, 2.0 ! constrain DM15 in this range
PRIOR_LUMIPAR_SIGMA = 0.7, 2.0 ! Gauss roll-off sigma for DM15 prior
INISTP_RV       = xxx  ! 0 => fix RV to INIVAL_RV
INIVAL_RV       = 2.2  !
INISTP_AV       = xxx  ! 0 => fix AV = INIVAL_AV in fit; else float
INIVAL_AV       = xxx
INISTP_LUMIPAR  = xxx  ! 0 => fix DELTA to INIVAL_LUMIPAR; else float
INIVAL_LUMIPAR  = xxx
```

Without a tuned ATLAS library (for `gsl`), the `SNooPy` generator is very slow such that a single light curve fits takes several minutes. To speed up the generator, the `SNooPy` model can be parameterized on a three-dimensional grid (filter, epoch,  $\Delta M_{15}$ ) using the `GRID` option from §4.27. This grid is specified by the `GRIDFILE` keyword in the `SNooPY.INFO` file that resides in the same directory as the model templates; both the simulation and fitter interpolate the `GRID` for each set of parameters. Also note that `SNooPy` returns a relative flux normalized to one at peak; the conversion to absolute magnitude is given for each filter in the `SNooPY.INFO` file.

## 9.4 SIMSED

A SIMSED model contains a full sequence of spectra for each epoch. These models typically result from specialized explosion-model codes such as FLASH, Sedona and Pheonix. Currently the SIMSED model is used only in the SNANA simulation; implementation in the fitter may come later when these models are more reliable. Each SIMSED version resides in

```
$SNDATA_ROOT/models/SIMSED
```

and contains a sequence of SEDs corresponding to parameters that describe the explosion model. The parameters are quantities such as Ni-56 mass, viewing angle, kinetic energy, and extinction. The only limit to the number of parameters (and SEDs) is the amount of memory on your computer.

Each SIMSED version contains an SED.INFO file specifying the list of parameter names, and the parameter values for each SED. An example of an SED.INFO file is as follows:

```
NPAR: 5
PARAMES: MNI COSANGLE MBSED DM15SED TEXPL
SED: GCD2D_Ni0.47_Pre80_1.SED 0.47 -0.967 -18.350 0.571 -21.628
SED: GCD2D_Ni0.47_Pre80_8.SED 0.47 -0.500 -18.413 0.574 -20.977
etc ...
```

Users must prepare the SED.INFO file and the SEDs,<sup>14</sup> as there is no standard SNANA code for this task.

To simulate a SIMSED model, the distribution for each parameter must be specified in the sim-input file as follows,

```
SIMSED_PARAM: MNI # mass of Ni56
GENMEAN_MNI: 0.9 # mean of bifurcated Gaussian
GENRANGE_MNI: 0.47 1.26 # generation range
GENSIGMA_MNI: 0.4 0.25 # bifurcated Gaussian

SIMSED_GRIDONLY: COSANGLE # cosine of viewing angle
GENMEAN_COSANGLE: 0
GENRANGE_COSANGLE: -0.96 0.96
GENSIGMA_COSANGLE: 1.E7 1.E7 # large sigmas => flat distribution

SIMSED_PATH_BINARY: <path for flux-table binary file>
```

You need to specify only those parameters that are used in the generation. For the SIMSED\_PARAM keyword, the simulation will interpolate magnitudes as a function of the explosion-model parameter, resulting in a continuous distribution of the specified bifurcated Gaussian. For the SIMSED\_GRIDONLY keyword, only values at the grid nodes are generated. This GRIDONLY option may be useful in cases

---

<sup>14</sup>The SED format is : epoch(days) wavelength(Å) flux(erg/cm<sup>2</sup>/s/Å).

where an integer flag specifies a random ignition point, and therefore continuous interpolation makes no sense. Also note that for the GRIDONLY option, the specified bi-Gaussian distribution is respected; each randomly chosen parameter is 'snapped' to the nearest grid value. Un-specified parameters are treated as "baggage" parameters. These parameters are ignored in the generation, but the interpolated values are computed and stored along with the other parameters.

To step through each grid-value sequentially,

```
SIMSED_GRIDONLY: SEQUENTIAL # sequential generation of grid values
```

and there is no need to specify parameter names or their distribution parameters. The first generated SN uses the first SED on the grid, the second SN uses the second value, etc. The number of generated SNe is internally set to be the number of SEDs.

All SIMSED parameters ("baggage" and for generation) are automatically included in the fitstuple and in the SIMGEN\_DUMP list (§4.25.3). If you specify a SIMSED parameter in the SIMGEN\_DUMP list the simulation will abort.

A binary flux-table (see below) is created in your current directory by default. Since these binary files can be quite large, there are two ways to specify a separate directory. The first method is to set the optional sim-input key SIMSED\_PATH\_BINARY as illustrated above. The second method is to define an environment variable, "setenv SIMSED\_PATH\_BINARY <myPath>." If both methods are used, the environment variable takes priority.

There are two internal binary files to speed up the initialization. In principle this all works behind the scenes, but it is explained here in case there are problems. The initialization takes about 1 second per SED, and will be rather annoying if/when there are 100's or 1000's of SEDs to initialize. About half the time is reading the SED text files, and the other half is spent doing all the flux-integrals as a function of passband, redshift, Trest, and SED surface. The first time that a new SIMSED version is simulated, the initialization will be slow, but two binary files are created to speed up future runs. The first binary file contains the SEDs (SED.BINARY), and it is stored in the same version (VVV) directory as the SED text files, \$SNDATA\_ROOT/models/SIMSED/VVV. This SED.BINARY file will be automatically used by any SNANA user who specifies the VVV version.

The second binary file is the flux-integral table, and this file is stored either in your local directory (default) or the directory specified by SIMSED\_PATH\_BINARY. The reason for storing in a user-specified directory is that the flux integrals depend on the survey and filters, and therefore this file is specific to your analysis. The validity of each binary file is verified internally; if there is an inconsistency between the two binary files and/or the requested survey parameters, the simulation will abort and recommend removing the old binary file(s) so that new ones can be created. A similar abort will occur if the user's binary flux-integral table has a time-stamp that is earlier than the SIMSED model or earlier than the K-cor file used to define the primary reference and filter transmissions.

If you are having problems with the binary files and just want to use the text files, the use of binary files can be switched off with

```
snlc_sim.exe mysim.input SIMSED_USE_BINARY 0
```

## 10 SALT-II Programs

### 10.1 Computing Distance Moduli from SALT-II fits: SALT2mu

While the `snlc_fit.exe` program includes the SALT-II model, the output does not include distance moduli. A separate program called “SALT2mu” reads the text-output of `snlc_fit.exe`, fits for  $\alpha$  and  $\beta$ , and computes a cosmology-independent distance modulus for each SN. An overview of the method is given in [10], a sample input file is in

```
$SNDATA_ROOT/analysis/sample_input_files/SALT2/SALT2mu.default
```

and a description of all input-file options are given at the top of the code in `$SNANA_DIR/src/SALT2mu.c`.

### 10.2 SALT-II Training Scripts

Scripts are `&SNANA_DIR/util/SALT2train_*`; more on this later after the paper is submitted.

## 11 Cosmology Fitters

There are currently two cosmology fitters available. First is `wfit.exe`, which fits for  $w$  and  $\Omega_M$  assuming a flat universe. Typing the `wfit.exe` command with no arguments lists the options. The BAO and CMB priors are currently hard-wired, but a more flexible method to specify priors may be added later. An example command is as follows,

```
wfit.exe <fitresFile> -zmin .02 -zerr .001 -snrms .15 -bao -cmb
```

which specifies using SNe with  $z > 0.02$ , adding a peculiar-velocity uncertainty of 300 km/s (i.e, the “zerr” arg is  $v_{\text{pec}}/c$ ), adding an anomalous distance-uncertainty of 0.15 mag (“snrms” arg), and using the BAO & CMB priors. Peculiar-velocity covariances can also be used as explained below in §11.1.

The second program, `sncosmo_mcmc.exe`, is a more general cosmology fitter based on Monte Carlo Markov chains. This fitter handles more diverse cosmologies such as time-dependent  $w$  and non-zero curvature. Sample inputs files are in

```
$SNDATA_ROOT/analysis/sample_input_files/sncosmo_mcmc/
```

Both of these cosmology fitters read self-documented “fitres” files (§12.1.1) that contain a redshift, distance modulus and survey index (other parameters in the fitres file are ignored).

The above cosmology fitters do not yet work on the SALT-II output since this light curve fitter does not produce a distance modulus. There are currently no SNANA programs that process the SALT-II fitres-output, but one can use the original “hubblefit” program (from Guy 2007) on the SNANA results if you use this namelist option in the light curve fitter:

```
SALT2_DICTFILE = 'myoutput.dictfile'
```

Recall that `hubblefit` performs a simultaneous fit for the cosmological parameters, as well as for SN properties ( $\alpha$ ,  $\beta$ ,  $M$ ). There is currently some development on new techniques for extracting cosmological results from the SALT-II light curve fits.

## 11.1 Peculiar Velocity Covariances

The `wfit.exe` program has an option to account for peculiar velocity correlations (SNANA v8\_10 and later). First prepare a file with the following syntax

```
COV:  SN1  SN2  MUCOVAR(12)
COV:  SN1  SN3  MUCOVAR(13)
COV:  SN1  SN4  MUCOVAR(14)
etc ...
```

where SN# are the SN names used in the analysis (i.e., that appear in the `fitres` file), and `MUCOVAR(ij)` are the covariances between the distance moduli, with units of  $\text{mag}^2$ . Only off-diagonal terms from this file are used; diagonal terms are specified from the `-zerr` and `-snrms` options. The syntax is

```
wfit.exe <fitresFile> -mucovar <mucovarFile> <other options>
```

where “`mucovarFile`” is the name of the file specifying the off-diagonal covariances. The `wfit.exe` program will first check your current directory for this file; if not there `wfit` will check the public area, `$SNDATA_ROOT/models/mucovar/`. The covariances for the nearby sample have been computed by the authors in [11], and these are available in

```
$SNDATA_ROOT/models/mucovar/Hui_LOWZ_mucovar.dat
```

The minimization function is given by  $\chi^2 = \sum_{ij} [\Delta_i V_{ij}^{-1} \Delta_j] - B^2/C$ ,<sup>15</sup> where  $\Delta_i \equiv \mu_i^{\text{data}} - \mu_i^{\text{model}}$  is the distance-modulus residual for the  $i$ 'th SN,  $V_{ij}^{-1}$  is the inverse of the covariance matrix, and the term  $B^2/C$  accounts for the analytic marginalization over  $H_0$  as discussed in Appendix A of [12]. The  $B$  and  $C$  parameters are

$$B = \sum_i (\Delta_i / \sigma_i^2) \longrightarrow \sum_{i,j} (\Delta_i V_{ij}^{-1}) \quad (17)$$

$$C = \sum_i 1 / \sigma_i^2 \longrightarrow \sum_{i,j} V_{ij}^{-1}, \quad (18)$$

where  $\sigma_i$  is the diagonal-uncertainty on the distance modulus. The expressions left of the arrows (Eqs. 17-18) are from [12], while the expressions right of the arrows show the `wfit` implementation that accounts for the off-diagonal covariance terms. The  $B^2/C$  term is equivalent to re-minimizing with the weighted-average distance-modulus residual subtracted from each distance-modulus residual;  $\Delta_i \rightarrow \Delta_i - \langle \Delta \rangle$ .

---

<sup>15</sup>We leave out the constant term  $\ln(C/2\pi)$ .

## 12 Miscellaneous Tools and Features

### 12.1 Analysis Variables, Fitres files and Ntuples

#### 12.1.1 Combining “Fitres” Files: `combine_fitres.exe`

As discussed in §5.2, the fit results are written to a self-documented “fitres” file. The simulation can also be used to dump generated variables into a file with the same format (§4.25.3). The utility `combine_fitres.exe` is useful to combine, or merge, multiple fitres files containing information about the same SNe. Note that fitres files with different SNe can be combined by simply using the unix “cat” command.

As an example, consider the following fitres files generated from different light curve fitters:

```
> more mlcs.fitres
NVAR: 2
VARNAMES: Z DLMAG
SN: 50001    0.1576  39.475
SN: 50002    0.2030  40.291

> more salt2.fitres
NVAR: 2
VARNAMES: x1 c
SN: 50001   -2.343  0.013
SN: 50002    0.893  0.235

> combine_fitres.exe mlcs.fitres salt2.fitres

> more combine_fitres.txt
NVAR: 5
VARNAMES: CID Z DLMAG x1 c
SN: 50001   0.157600001  39.4749985 -2.34299994  0.0130000003
SN: 50002   0.202999994  40.2910004  0.893000007  0.234999999
```

Note that although the SN candidate id (CID) is required after the “SN:” keyword, it is optional to specify CID in the VARNAMES list.

Fitres files with the same variable names can also be combined. The second repeated variable gets a “2” appended to the variable name, the third repeated variable gets a “3” appended, etc ... For example, consider two MLCS2k2 fits with slightly different options,

```

> more mlcs.fitres
NVAR: 2
VARNAMES: Z DLMAG
SN: 50001    0.1576  39.475
SN: 50002    0.2030  40.291

> more mlcs_test.fitres
NVAR: 2
VARNAMES: Z DLMAG
SN: 50001    0.1576  39.829
SN: 50002    0.2030  40.443

> combine_fitres.exe mlcs.fitres mlcs_test.fitres

> more combine_fitres.txt
NVAR: 5
VARNAMES: CID Z DLMAG Z2 DLMAG2
SN: 50001    0.157600001  39.4749985  0.157600001  39.8289986
SN: 50002    0.202999994  40.2910004  0.202999994  40.4430008

```

Up to ten fitres files can be merged together. If there are SNe in the second (3rd, 4th...) fitres file that are not in the first fitres file, then these SNe will be ignored. If there are SNe in the first fitres file that are not in the other fitres files, then values of -999 are stored for the missing SNe. In addition to creating a merged fitres file, a merged ntuple file (combine\_fitres.tup) is also created. This ntuple can be analyzed with PAW or root.

The default prefix for the output files is “combine\_fitres.” This default can be changed using the argument -outprefix,

```
> combine_fitres.exe mlcs.fitres mlcs_test.fitres -outprefix mlcs
```

which produces the files 'mlcs.tup' and 'mlcs.txt.'

### 12.1.2 Ntuple ↔ Fitres Format: ntprint.pl and ntdump.pl

In addition to the fitres file output from the light curve fitter, there are also ntuples with much more extensive information about each SN. If you are familiar with PAW, you can immediately analyze ntuples 7788 and 7799. If you are not familiar with PAW, and are not in the mood to learn, you can extract interesting variables into a “fitres-formatted” text-file. The first step is to get a list of available variable using the ntprint utility:

```
> ntprint.pl snfit.his 7788
```

Hopefully the meaning of each variable is obvious; if not, you will need to ask or look at the source code. Next, create a file containing the names of the variables that you want to extract; for example,

```
> more MYVAR.LIST
CID SNRMAX_g SNRMAX_r
    SNRMAX_i
```

The first variable *must* be CID, and the variable names can be separated by blank spaces and/or carriage returns. You can also use upper and/or lower case since the underlying extraction routine is case-insensitive. Finally, to extract the above variables from the ntuple into a text file,

```
> ntDump.pl snfit.his 7788 MYVAR.LIST HEADER
    or
> ntDump.pl snfit.his 7788 'SNRMAX_g SNRMAX_r SNRMAX_i' HEADER
    or
> ntDump.pl snfit.his 7788 MYVAR.LIST
```

The first command gives an output file in the self-documented fitres format; i.e., with the NVAR and VARNAMES header. The second command does the same thing, but passing an explicit list of variables in quote rather than passing a file-name. Note that if only one variable is listed in quotes, a blank space is needed: 'SNRMAX\_g '. The 3rd command leaves out the HEADER option, so that the output file has no header and no SN keywords. You can get help with both utilities by typing

```
> more $SNANA_DIR/util/ntprint.pl
> more $SNANA_DIR/util/ntDump.pl
```

### 12.1.3 Appending Variables to the Standard Fitres Output: ntappend2fitres.pl

If you set the &FITINP namelist variable FITRES\_DMPFILE, you will get a standard subset of variables that are adequate for a typical cosmology fitter. However, there may be more sophisticated programs that require additional information, such as covariances, error-flags, S/N ratios, etc ... Your fitres file can be appended with an arbitrary set of variables stored in the fitres ntuple (ntid 7788) or from any other ntuple. The utility script runs as follows,

```
ntappend2fitres.pl myappend.input
    or
ntappend2fitres.pl myappend.input DUMP
```

where the input file contains the relevant input information. The DUMP option prints a list of all available variables to choose from; you may have to search to code to find the exact definition. See top of \$SNANA\_DIR/util/ntappend2fitres.pl for instructions on filling out the input file.

#### 12.1.4 Extract Value from Fitres File: `get_fitresValue.pl`

See instructions at top of `$SNANA_DIR/util/get_fitresValue.pl`

## 12.2 General Misc. Tools

### 12.2.1 Command-line Overrides

The simulation and light curve fitter described in the sections above are each driven by an input file that specifies instructions and parameters. For convenience, all input-file parameters can be specified on the command line. For example, if you have

```
GENMODEL: mlcs2k2.v006
GENTAU_AV: 0.35
```

in your simulation-input file, you can override these options on the command-line without editing the input file:

```
snlc_sim.exe mysim.input GENMODEL mlcs2k2.v007 GENTAU_AV 0.45
```

These command-line overrides are useful for quick testing, and for writing wrappers that do many analysis variations without creating a new input file for each job. An invalid or unrecognized command-line option results in an immediate abort. The command-line options are printed to stdout, and therefore if you pipe your job to a log-file, you can rerun the same job as long as you have the original input file.

### 12.2.2 Synchronizing/Updating Survey Files: `survey_update.pl`

Collaborators within a survey who are working on different computer systems can keep files synchronized using the script `$SNANA_DIR/util/survey_update.pl`. See usage instructions at the top of the script. The basic idea is that a survey expert uses the script in “create tarball” mode to create a tarball containing filter transmission files, K-correction tables and data version(s). This tarball is then distributed among collaborators who use the same script in “install” mode.

### 12.2.3 Bug-Catcher: the `SNANA_tester` Script

To help verify that the SNANA code delivers the same results with each new version, there is a testing utility, `SNANA_tester.cmd` (no arguments), that runs a pre-defined list of jobs with two different versions of SNANA, and reports discrepancies. Typically this script is run just before releasing a new SNANA version, but users may want to run this script on other platforms. The goal is to catch and fix unanticipated changes (i.e, bugs) before releasing each new SNANA version. The top-level instruction file is here,

```
$SNDATA_ROOT/SNANA_TESTS/SNANA_TESTS.LIST
```

which specifies a series of test-jobs, input files, and log-file parsing instructions to extract results to compare between SNANA versions. The `SNANA_tester.cmd` script is written for the Fermilab ups product system; on other platforms, script modifications will be needed to setup the correct versions of SNANA. Users who use a particular feature of SNANA should check if the current test-jobs provide adequate protection; if not, you may request additional test-jobs.

#### 12.2.4 Data Backup/Archival: `backup_SNDATA_version.cmd`

A data version can be archived with

```
backup_SNDATA_version.cmd MYVERSION
```

which creates a tarball-backup in

```
$SNDATA_ROOT/lcmerge/archive/MYVERSION_[yyy]-[mm]-[dd].tar.gz
```

where `[yyyy]-[mm]-[dd]` is the year, month and day. This script is useful to backup a newly made data version, archive a version used in a publication, or to send a data version to a collaborator. In general the `$SNDATA_ROOT` disk is not backup up, so to have a truly protected backup you need to either (1) backup `$SNDATA_ROOT` or (2) copy the tarball-backup to another storage device.

#### 12.2.5 K-correction Dump Utility: `kcordump.exe`

The K-correction tables are stored in HBOOK files, and accessing this information can be tricky even for those familiar with PAW. The utility `kcordump.exe` can be used to check a K-correction value for specific filter, epoch, and primary reference. If you type `kcordump.exe` with no arguments, the program will ask you for all needed arguments. You can also use command-line arguments, as illustrated here for SNLS  $K_{gU}$  at peak at  $z = 0.28$ :

```
> kcordump.exe HFILE_KCOR Hsiao/kcor_SNLS_Bessell90_VEGA.his  \\
                FILT_OBS g  FILT_REST U  Z .28  TREST 0.
```

The dump utility checks your current directory and `$SNDATA_ROOT/kcor` for the existence of the K-correction file (argument of `HFILE_KCOR`). All K-correction dumps are done with no spectral warping. To see K-corrections with warping, generate simulated SNe Ia and scroll through the data files for symbols containing “KCOR” and “WARP.” The slight disadvantage with using the simulation to check K-corrections is that you cannot specify which K-corrections to check, but you can only compare to whatever the simulation generates.

## 12.3 Misc. Simulation Tools

### 12.3.1 Simulate Ia/non-Ia mix: `sim_SNmix.pl`

See instructions at top of `$SNDATA_ROOT/util/sim_SNmix.pl`. In addition to simulating a mix of Ia and non-Ia SNe, this script can also be used to generate multiple sets of simulations with different sim-options. An optional list of computing nodes can be used for parallel processing.

### 12.3.2 Co-Adding SIMLIB Observations on Same Night: `simlib_coadd.exe`

If a survey takes many exposures per filter in one night, the resulting SIMLIB can be quite large, and there may be no benefit to simulating each exposure within a night. Since one typically combines these exposures into a single co-added exposure, there is a utility to translate a SIMLIB so that there is just one effective co-added exposure per filter per night,

```
> simlib_coadd.exe MYSURVEY.SIMLIB
```

which produces an output SIMLIB called `MYSURVEY.SIMLIB.COADD`. By default, observations within 0.4 days are combined into a single co-added observation, and at least three observations are required to keep a sequence of observations (i.e, a LIBID). The CCD noise, sky-noise and zeropoint are calculated to reflect a single co-added exposure as follows,

$$\text{ZPT}(\text{COADD}) = 2.5 \log_{10} \left[ \sum_i 10^{(0.4 \cdot \text{ZPT}_i)} \right] \quad \text{NOISE}(\text{COADD}) = \sqrt{\sum_i \text{NOISE}_i} \quad , \quad (19)$$

where  $i$  is the exposure index. The co-added PSF is simply the average of the PSF values from the individual exposures.

There are additional options to change the requirement on the minimum number of observations, to change the time-separation for co-adding, and to determine the Milky Way Galactic extinction (MWEBV) from [7],

```
> simlib_coadd.exe MYSURVEY.SIMLIB MWEBV --TDIF .2 --MINOBS 5
```

When the “MWEBV” option is used, observation sequences with  $\text{MWEBV} > 2$  are rejected. Read top of `$SNANA_DIR/src/simlib_coadd.c` for additional options.

### 12.3.3 Preparing Non-Ia Templates for the Simulation

... coming soon ...

### 12.3.4 Fudging Simulated Errors and Signal-to-Noise Ratio (S/N)

Errors and  $S/N$  can be fudged in both the simulation and the fitting programs. Here is a list of SNANA “fudge-options” starting with the simulation.

```

# options to adjust exposure time (affects both SN flux and sky noise)
EXPOSURE_TIME: 20          # increase all exposure times by 20
EXPOSURE_TIME_FILTER: g 20 # increase g exposure by 20
FUDGESHIFT_ZPT: -3.0       # reduce ZPT by 3 mags

# options to increase SKY-noise while leaving SN flux unchanged
FUDGESCALE_PSF: 2          # increase PSF
FUDGESCALE_SKYNOISE: 3    # increases SKYNOISE (note that SKY *= 3^2)

# force nearest-peak S/N = 25 at all redshifts:
FUDGE_SNRMAX: 25          # adjust exposure time
FUDGE2_SNRMAX: 25         # adjust sky-noise only (don't change SN flux)

```

There are three classes of error-fudging: (1) adjust exposure time to change both the SN flux and sky-noise, (2) increase the sky-noise while leaving the SN flux unchanged, and (3) force nearest-peak  $S/N$  to be the same fixed value at all redshifts. `FUDGE2_SNRMAX` is useful for setting a nearly fixed error at all epochs, in contrast to a fixed mag-error. For both options to fix the nearest-peak  $S/N$  ratio, the simulation processes each SN twice. The first iteration is done with nominal conditions and the resulting  $\text{SNRMAX}^{16}$  is used to calculate conditions for the second iteration. Defining  $\text{SNR}_i$  to be the  $i$ 'th-iteration  $S/N$  where  $\text{SNR}_2$  is the requested `FUDGE[2]_SNRMAX` value, and  $\mathcal{R} \equiv \text{SNR}_2/\text{SNR}_1$ , the zeropoint and sky-noise are modified for each passband in the second iteration as follows:

$$\text{FUDGE\_SNRMAX} : ZPT_2 - ZPT_1 = 5 \log(\mathcal{R}) \quad (20)$$

$$\text{FUDGE\_SNRMAX} : \sigma_{\text{sky},2}/\sigma_{\text{sky},1} = \mathcal{R} \quad (21)$$

$$\text{FUDGE2\_SNRMAX} : ZPT_2 - ZPT_1 = 0 \quad (22)$$

$$\text{FUDGE2\_SNRMAX} : \sigma_{\text{sky},2}/\sigma_{\text{sky},1} = \sqrt{[(1/\text{SNR}_2)^2 - 1/\hat{F}_{\text{SN}}] / [(1/\text{SNR}_1)^2 - 1/\hat{F}_{\text{SN}}]} \quad (23)$$

where  $\hat{F}_{\text{SN}}$  is the SN flux (photoelectrons) at the epoch with the maximum  $S/N$ .

---

<sup>16</sup>`SNRMAX` is the maximum  $S/N$  ratio among all observations within a passband.

## 12.4 Misc. Fitting Tools

### 12.4.1 Fit Multiple Samples with Multiple Fit-Options: `split_and_fit.pl`

See instructions at top of `SNANA_DIR/util/split_and_fit.pl`. This script allows fitting a matrix of multiple versions and multiple fit-options. A list of nodes allows for parallel processing, and the outputs are automatically merged when the fitting jobs have finished.

### 12.4.2 Analyzing Residuals from Lightcurve Fits

There are two utilities to analyze residual from a lightcurve fit:

```
mkfitplots.pl --h <hisFile> RESIDS
dump_lcfiteOutliers.pl <hisFile> <Nsigma>
```

The first command generates plots of the normalized residuals,  $\Delta F/\sigma$  where  $\Delta F = F_{\text{data}} - F_{\text{model}}$ , and also plots  $\Delta F/\sigma$  vs.  $\log_{10}(\text{SNR})$ . Separate plots are made for each observer-frame passband. The second command dumps out a text-file of outlier observations for which the data lie more than  $N\sigma$  from the best-fit model. The format of the outlier text-file is such that entries can be pasted directly into the IGNORE file for those observations that should be ignored in the analysis.

### 12.4.3 Extracting Light Curves into ASCII Formatted Files

For both the `snana.exe` and `snlc_fit.exe` programs, the fluxes and best-fit model can be dumped into an ASCII text file with the `&SNLCINP` namelist option

```
LDMP_SNFLUX = T
```

This option creates a master file “`$SURVEY.LIST`” and a `FLUXFILE` for each filter for each SNID. The master file contains the name of each SNID, its redshift and a list of `FLUXFILES`. The format of the `FLUXFILE` is

```
Tobs    FLUXCAL    FLUXCAL_ERR    DATAFLAG
```

where `Tobs` is the time relative to peak brightness, `FLUXCAL` and `FLUXCAL_ERR` are the calibrated flux and error, and `DATAFLAG` is one for data and zero for the best-fit model.

#### 12.4.4 Translating SNDATA files into SALT-II Format

The `snana.exe` program can convert SNANA-formatted data (or simulation) files into SALT-II format needed to run the original (Guy07) SALT-II fitter code and the SALT-II training code. A sample namelist is as follows:

```
&SNLCINP
  VERSION_PHOTOMETRY = 'SDSS_HOLTZ08'
  OPT_REFORMAT_SALT2 = 2
  REFORMAT_KEYS      = '@INSTRUMENT SLOAN @MAGSYS AB'
  SNFIELD_LIST       = '82N' , '82S'
  cutwin_cid         = 0, 100000
&END
```

Note that `OPT_REFORMAT_SALT2=2` is for the newer SALT-II format with one file per SN (snfit version 2.3.0 and higher), while `OPT_REFORMAT_SALT2=1` is for the original format with one file per passband.

#### 12.4.5 Translating TEXT data-files into FITS Format

For relatively large data samples, reading text files can be somewhat slow. A more efficient storage mechanism uses FITS format. TEXT-formatted data files can be translated into FITS format using `snana.exe` and an input file with the following,

```
&SNLCINP
  VERSION_PHOTOMETRY      = 'MYSURVEY_TEXT'
  VERSION_REFORMAT_FITS   = 'MYSURVEY'
&END
```

This translation should be used only for data since the simulation is written in FITS format by default.

The intermediate TEXT-format can be skipped by writing your own C code that fills the SNDATA structure in `sndata.h` and makes calls to functions in `snfitsio.c`. Follow `WRFLAG_FITS` in `snlc_sim.c` for details.

#### 12.4.6 Fudging Fitting Errors

For the `snlc_fit.exe` fitting program there are the `FUDGE_FITTER_XXX` parameters described in detail in §5.13. The other fudge-option is the `&FITINP` namelist variable

```
FUDGE_MAGERR_MODEL = 0.1 # fix model mag-error in fitter
```

which replaces all model errors with 0.1 mag model-error at every epoch.

## 12.5 Misc. SIMSED Utilities

### 12.5.1 SIMSED **Spectrum Extraction:** SIMSED\_extractSpec.exe

For a SIMSED model, the program SIMSED\_extractSpec.exe can be used to extract a single spectrum for a particular SIMSED model version, epoch, and set of parameter values corresponding to the PARNAMES in the SED.INFO file. The current program uses the parameter values on the SED.INFO grid that are closest to the user-specified parameters; a future version may interpolate for better accuracy. See usage instructions at top of \$SNANA\_DIR/src/SIMSED\_extractSpec.c .

### 12.5.2 SIMSED **Fudge Afterburner:** SIMSED\_fudge.exe

For a given SIMSED model (§9.4), an arbitrary color law can be applied to generate a new SIMSED version. The program syntax is

```
SIMSED_fudge.exe <inFile>
```

where an example input file is here:

```
$SNDATA_ROOT/analysis/sample_input_files/SIMSED/colorFudge.input
```

### 12.5.3 SIMSED **Preparation for SNANA:** SIMSED\_prep.exe

Translate native format of SED model into SNANA format.

## 13 SNANA Updates

The SNANA software is released in incremental versions, reflecting improvements, new code, and bug fixes. Users should periodically check for updated versions. E-mail notices to SNANA@FNAL.GOV are sent when there are significant changes. Users are encouraged to sign up for the SNANA mailing list by asking any co-author of the overview paper (arXiv:0908.4280). The SNANA mailing list can also be used to ask for help, report bugs, suggest changes, etc ... Details of each release can be found by doing

```
tail -100 $SNANA_DIR/doc/README_UPDATES
```

Users should develop an automated “download & setup” script to easily maintain the most current version of SNANA, and to avoid getting trapped with an old or obsolete version. The SNANA\_ROOT tarball is updated less frequently; README\_UPDATES will indicate if and why a new SNANA\_ROOT is needed.

This manual (\$SNANA\_DIR/doc/snana\_manual.pdf) is updated mostly in response to questions from users. If you spot mistakes or obsolete information in the manual, please report it immediately !

## 14 Reporting Problems

Please don’t hesitate to report software problems or obsolete/incorrect information in the manual. If the problem is related to an abort or crash, please include a tarball that contains the input file(s) and data file(s) that reproduce the problem, as well as a log-file with the program’s screen-dump. Indicate which SNANA version was used, and try to isolate the problem in a single data file to avoid sending large numbers of data files. **WARNING: if you don’t provide enough information to reproduce the problem, we will not be able to provide assistance.**

If the problem is related to processing simulated data files, please do the following. First, manually create a special version called SIM\_DEBUG that resides in SNANA\_ROOT/SIM/SIM\_DEBUG. Copy the relevant data file(s) that cause the problem, such as unexpected abort or crazy fitted values. In the same directory, manually create the needed auxiliary files as follows:

```
touch SIM_DEBUG.README
touch SIM_DEBUG.IGNORE
ls *DAT >! SIM_DEBUG.LIST
```

Finally, send this directory, along with the needed input files, to one of the developers.

## References

- [1] S. Jha, A. G. Riess, and R. P. Kirshner. Improved Distances to Type Ia Supernovae with Multicolor Light Curve Shapes: MLCS2k2. *AJ*, 659:122, 2007.
- [2] J. Guy et al. SALT2: Using Distant Supernovae to Improve the use of Type Ia Supernovae as Distance Indicators. *A&A*, 466:11–21, 2007.
- [3] C. R. Burns et al. The Carnegie Supernova Project: Light-curve Fitting with SNooPy. *AJ*, 141:19, January 2011.
- [4] G. Goldhaber et al. Timescale Stretch Parameterization of Type Ia Supernova B-band Light Curves. *Astrophys. J.*, 558:359–368, 2001.
- [5] B. Hayden, P. Garnavich, and SDSS-SN Survey. The Rise and Fall of SDSS-II Supernovae. *Bulletin of the American Astronomical Society*, 41:254, 2009.
- [6] R. Kessler et al. SNANA: A public software package for supernova analysis. *PASP*, 121:1028, 2009.
- [7] D. J. Schlegel, D. P. Finkbeiner, and M. Davis. Maps of Dust Infrared Emission for Use in Estimation of Reddening and Cosmic Microwave Background Radiation Foregrounds. *Astrophys. J.*, 500:525, June 1998.
- [8] P. Nugent et al. K-Corrections and Extinction Corrections for Type Ia Supernovae. *PASP*, 114:803, 2002.
- [9] M. Sako et al. Photometric Type Ia Supernova Candidates from the Three-year SDSS-II SN Survey Data. *Astrophys. J.*, 738:162, September 2011.
- [10] J. Marriner et al. A More General Model for the Intrinsic Scatter in Type Ia Supernova Distance Moduli. *Astrophys. J.*, 740:72, October 2011.
- [11] L. Hui and P. B. Greene. Correlated Fluctuations in Luminosity Distance and the Importance of Peculiar Motion in Supernova Surveys. *Phys. Rev.*, 73(12):123526, 2006.
- [12] M. Goliath et al. Supernovae and the Nature of the Dark Energy. *A&A*, 380:6–18, 2001.