SNANA User's Manual: Simulation, Lightcurve Fits, BBC, Create-Covariance, Batch Utilities

Richard Kessler
University of Chicago
Kavli Institute for Cosmological Physics
Department of Astronomy & Astrophysics

December 13, 2025

Contents

1 Introduction

This manual describes how to use the lightcurve simulation and fitting programs 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 an arbitrary survey. Current SN models include MLCS2k2 [1], SALT-II [2], BAYESN [3], SNooPy [4], stretch [5], two-stretch [6], and Core Collapse [7].

The simulation is designed to be fast, generating $\sim 10^2$-10^3 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 "observing conditions" in a "SIMLIB" file that describes the seeing, sky-noise, zeropoints, and cadence. A SIMLIB file is easy to prepare post-survey; predicting the SIMLIB 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, such as scaling the flux-errors.

The underlying programs are binary executables based on a mix of fortran and C. Each program (fitting or simulation) requires an input file as an argument, plus optional command-line overrides (§12.2.3). The command-line overrides allow making small perturbations so that a new input file is not needed for each variation. For typical analyses that require many variations in both the fitting and simulations, "submit_batch_jobs.sh" is a utility (§12.2.1) to launch job sequences on multiple cores using batch systems such as "qsub" or "sbatch."

Finally, while the simulation is a stand-alone program with no user interface, the analysis programs allow for user interaction in multuple ways using the *private* option (§6.1). The idea is to layer code on top of the underlying snana code that reads data (real and sim) and applies basic selection cuts. This architecture allows users to focus on writing new analysis features without worrying about the overhead of reading data files. Three ways to use the private interface are 1) write an entire analysis or fitting package (e.g., snlc_fit or psnid), 2) use existing code, but modify an underlying algorithm, and 3) add private code in the user-interface routines (USRINI,USRANA,USREND), such as computing a new variable, writing information in a specific format, etc.

2 SNANA Basics

After installing SNANA (see install guide), make sure that the two main environment variables are defined:

```
> echo $SNANA_DIR/
> echo $SNDATA_ROOT/
```

The environment variable $SNANA_DIR points to the software that should be accessible to a group of users. The SNANA developers use githib[1] to share and distribute code, and occasional tags are released. SNANA codes are updated often with bug fixes and improvements; each github update follows a successfull regression test checking more than 100 SNANA jobs. The main science codes are:

```
  > $SNANA_DIR/src/snlc_sim.c     # simulation
  > $SNANA_DIR/src/snlc_fit.F90   # light curve fitting
  > $SNANA_DIR/src/psnid.F90      # psnid classifier
  > $SNANA_DIR/src/SALT2mu.c      # implement BBC to create hubble diagram

  > $SNANA_DIR/util/create_covariance.py # create stat+syst cov matrix
  > $SNANA_DIR/util/submit_batch.sh      # submit batch jobs
```

The environment variable $SNDATA_ROOT points to a directory that contains publicly released information from survey teams, and is downloadable from zenodo.[2] $SNDATA_ROOT includes public data, filter transmissions, CalSpec SEDs, SN models, cadence libraries (SIMLIB), host libraries (HOSTLIB), Milky Way extinction map, etc.

A 200+ page manual is part of github ($SNANA_DIR/doc/snana_manual.pdf). For beginners, however, it is recommended to get help from an expert.

2.1 Dr. Evil-ABORT-Face

The SNANA programs include intensive error checking throughout the execution. If anything looks fishy, the program aborts with an easy-to-identify message looking like

```
  .|.......|.
  <| o\ /o |>
   | ' ; ' |
   |  ___  |     ABORT program on Fatal Error.
   | |' '| |
   | '---' |
   \_______/

FATAL ERROR ABORT called by RDSNNML
  Could not open namelist file:
  dummy.nml
```

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.

[1] https://github.com/RickKessler/SNANA
[2] https://zenodo.org/record/4728252

2.2 Data Files

Data files can be written in TEXT (§4.37.2) or FITS (§4.37.1) format. Large data & sim samples should use FITS format because it is much more compact and much faster to read. With TEXT format, each event is written to a separate ASCII file. With FITS format, there are two FITS files: a summary HEAD file with a one-row summary per event (Fig. 1), and a PHOT file with the light curves (Fig. 2). For fast reading, the HEAD file includes pointers to the PHOT table (`PTROBS_MIN, PTROBS_MAX`), so do not catenate FITS files without updating these pointers.

2.3 Citations

While an `SNANA` citation is always appreciated ([8]), please make sure to reference 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.

```
HEAD file:
  variable name              description
# ------------------------------------------------------------------
* SNID                       integer or character ID
* FAKE                       0=real data, 1=fake overlaid on image, 2=SNANA sim,
* RA,DEC                     sky coordinates (degrees)
* SNTYPE                     integer type assigned by survey (e.g., SPEC type)
* NOBS                       number of obs (all bands)
* PTROBS_MIN                 pointer to 1st light curve obs in PHOT file
* PTROBS_MAX                 pointer to last light curve obs
* MWEBV[_ERR]                E(B-V) for Milky way
* REDSHIFT_HELIO[_ERR]       best heliocentric redshift;
                             zSpec if available; else zPhot; else -9
* REDSHIFT_FINAL[_ERR]       best CMB redshift (no VPEC correction)
* VPEC_[ERR]                 pec velocity (km/sec) and error on correction

* PEAKMJD                    approx PEAKMJD; useful to init light curve fit
* MJD_TRIGGER                MJD when survey trigger is satisfied
* MJD_DETECT_FIRST           MJD of first detection
* MJD_DETECT_LAST            MJD of last detection

* HOSTGAL_SB_FLUXCAL_[band] Surface brightness mag at SN location
* HOSTGAL_NMATCH             number of host matches with DDLR < 4
* HOSTGAL_NMATCH2            number of host matches with DDLR < 7
* HOSTGAL_OBJID              integer id of host (long long int)
* HOSTGAL_RA                 R.A. (deg) for center of host
* HOSTGAL_DEC                declination (deg) for center of host
* HOSTGAL_SNSEP              transient-host separation, arcsec
* HOSTGAL_DDLR               SN-host sep in distance-light-radii (d_DLR)
* HOSTGAL_SPECZ[_ERR]        zSpec of host  (-9 -> not available)
* HOSTGAL_PHOTOZ[_ERR]       zphot (mean of PDF); -9 -> not available
* HOSTGAL_ZPHOT_Q[PPP]       redshift containing PPP percent of zPDF prob
* HOSTGAL_LOGMASS[_ERR]      logmass and error
* HOSTGAL_LOGSFR[_ERR]       log(star formation rate) and error
* HOSTGAL_MAG_[band]         host mags for band=u,g,r,i,z,Y
* HOSTGAL_MAGERR_[band]      uncertainty on above mags
    [Each hostgal_xxx has hostgal2_xxx for 2nd host-match;
     -999 indicates no 2nd host; else values are filled.
     First listed host has smallest DDLR and may not be the true host]

     For simulated data:
* SIM_TYPE_INDEX             true integer transient type (e.g., 1 for SNIa)
* SIM_[SSS]                  true sim property SSS
```

Figure 1: FITS data structure for header file (*HEAD.FITS).

```
PHOT file:
* MJD                       modified Julian date (-777 -> end of light curve)
* BAND                      e.g.,  'u', 'g', 'r' etc ...
* CCDNUM                    CCD or detector number
* IMGNUM                    image number; e.g. exposure number, visit id ...
* FIELD                     name of field; e.g., SHALLOW, DEEP, etc ...
* PHOTFLAG                  bit-mask of information (check data README)
* PHOTPROB                  float metric; e.g, RealBogus score, chi2, ...
* FLUXCAL                   calibrated flux : mag = 27.5 - 2.5*log10(FLUXCAL)
* FLUXCALERR                Poisson uncertainty on FLUXCAL; sky+galaxy+source

     image properties
* PSF_SIG1                  PSF Gauss sigma, pixels
* SKY_SIG                   sky noise (ADU/pixel)
* SKY_SIG_T                 template sky noise (for DIFFIMG, not for SMP)
* ZEROPT                    image zero point
* GAIN                      N_photoelectron/ADU
* XPIX                      x-locaton on CCD (pixels)
* YPIX                      y-locaton on CCD (pixels)

    For simulated data:
* SIM_MAGOBS                true model mag
```

Figure 2: FITS data structure for photometry file (*PHOT.FITS)

2.4 Community Code Contributions

First and most important, avoid the temptation to make quick-and-dirty fixes for your private use. Anything you need is likely to be useful for somebody else, so please work with SNANA developers to integrate your updates into the public github repository.

Before making code modifications, it is good practice to inform the primary SNANA developers, but permission is not needed, especially for urgent fixes. The SNANA code is stored on github[3], and you need to become a collaborator to submit a pull request (PR). If it is not practical to become a code collaborator (e.g., one-line code fix), an update may be sent to SNANA developers via e-mail.

Before submitting a PR, it is important to perform a regression test using the command

```
SNANA_code_tests.py --snana_dir <my_snana_dir>
```

This regression test has been setup on clusters at Fermilab (des*.fnal.gov), U.Chicago (midway2 at RCC), and NERSC (Cori). Please contact SNANA developers for help on setting up regression tests on other clusters. If you create a code update on a private laptop or cluster, an SNANA developer can run the regression test for you.

A few other requests for coding habits:

- post github issue.
- clone or update SNANA code before making changes.
- Verify that update/fix actaully works; successful compilation is not adequate.
- Do not fall into this dangerous logic trap: "*I only changed one line of code, so no need to test it.*" As a post-doc working on the KTeV experiment, I accepted this logic and installed a one-line code fix that crashed the entire experiment.
- annotate history section at top of file and/or function so that future code editors can easily notice who/when previous changes were made. A very brief description is adequate.
- annotate end of doc/README_UPDATES in the appropriate category of IMPORTANT (e.g., major bug-fix or new feature), USEFUL (new feature with limited scope), or MISCELLANEOUS (little interest to anyone except the person posting issue). While this document is redundant with github comments, searching key-words in this file has been very useful to track histories.
- for a significant new feature, consider updating SNANA manual (pdf you are reading now).
- To avoid conflicts that are difficult to resolve, avoid editing code for long time periods without merging. Rather than waiting several weeks for all coding to finish, send multiple PR's, each with small changes. Make sure that your development does not interfere with normal code usage; e.g., use DEBUG_FLAG or new input key to invoke your new code.
- after submitting PR, contact Kessler or Narayan to merge it (free github usage does not include automated e-mail alert).
- for major bug-fixes or improvements, inform your collaborators.

[3] https://github.com/RickKessler/SNANA

3 The Calibration + K-Correction file

Before running the simulation 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/sample_input_files/kcor`
and the command to create a K-corr file is

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

The output file is specified by the kcor-input file key

```
OUTFILE:  mySurvey.fits
```

3.1 Changing the Mean Filter Wavelength

The mean filter wavelength can be adjusted via the command-line argument

```
kcor.exe  myKcor.Input  FILTER_LAMSHIFT  r 2.1  i 3.2
kcor.exe  myKcor.Input  FILTER_LAMSHIFT  r 2.1  i 3.2  OUTFILE kcor_lamshift.fits
```

which shifts the mean r- and i-band wavelengths by 2.1 and 3.2 Å, respectively. These shifts can be entered only via command-line arguments, and it is therefore recommended to also include a unique `OUTFILE` name as well, as shown in the 2nd example above. The LAMSHIFT info is written into the output header. Note that to implement more complex wavelength variations requires a new set of transmission-vs-λ curves.

§5.25 shows how to define a duplicate set of filters with a common wavelength shift, and how to select the λ-shifted band(s) in the fitting program.

3.2 Defining a SPECTROGRAPH

A SPECTROGRAPH instrument is defined as a list of SNR-vs-wavelength, SNR(λ), for two distinct magnitudes. SNR(λ) is defined for two (and only two) mag values so that in each λ-bin an effective zero point and skyNoise is computed analytically, allowing SNR(λ) to computed for any mag. For each spectroscopic bin-center, the zeropoint (ZP) and skyNoise ($\sigma_{\rm sky}$) are given by

$$\mathrm{ZP} = 2.5\log_{10}\left[\frac{10^{-0.4m_1}-10^{-0.4m_2}}{(10^{-0.4m_1}/\mathrm{SNR}_1)^2-(10^{-0.4m_2}/\mathrm{SNR}_2)^2}\right] \quad (1)$$

$$\sigma^2_{\rm sky} = (F_1/\mathrm{SNR}_1)^2 - F_1 \;\;;\;\; F_1 \equiv 10^{-0.4(m_1-\mathrm{ZP})} \quad (2)$$

where $m_{1,2}$ are the two AB mag-reference values, and $\mathrm{SNR}_{1,2}$ are the corresponding SNR. Beware that there is no solution if $m_2 - m_1 < 2.5\log_{10}(\mathrm{SNR}_1/\mathrm{SNR}_2)$. To account for exposure-time ($T_{\rm expose}$) dependence, SNR(λ) can be defined for multiple $T_{\rm expose}$ values. The SPECTROGRAPH is defined in a custom table file as shown in Fig. 3. The required keys include name of instrument, two ref mag

```
# required keys:
INSTRUMENT: MYSPECDEVICE
MAGREF_LIST:  20 28           # defines SNR1 and SNR2 (ony 2 allowed)
TEXPOSE_LIST: 300 1000 2000   # seconds (arbitrary list size)

# optional keys to check for bad SNR inputs
SNR_POISSON_RATIO_ABORT_vsMAGREF:  5     # default
SNR_POISSON_RATIO_ABORT_vsTEXPOSE: 1.2   # default

#         LAM   LAM   LAM
#         MIN   MAX   RES    SNR1 SNR2      SNR1 SNR2      SNR1 SNR2
SPECBIN:  4200  4210  12.4  11.39 0.017    24.78 0.063    50.72 0.168
SPECBIN:  4210  4222  14.5  12.39 0.018    24.94 0.066    50.82 0.172
etc ...
```

Figure 3: Spectrograph table format for kcor program and simulation.

values (m_1, m_2 in Eq. 2), and a list of exposure times. The optional keys to check for bad SNR inputs are described below.

Each λ-bin (SPECBIN key) includes $\lambda_{\rm min}$ and $\lambda_{\rm max}$ (Å) to avoid confusion with non-uniform λ bins. The LAMRES column specifies the wavelength resolution as Gaussian σ in Å. Since there are three $T_{\rm expose}$ values in the example above (see TEXPOSE_LIST key), three SNR pairs are given, where each SNR pair corresponds to the two mag values following the MAGREF_LIST key. An abritrary number of $T_{\rm expose}$ values can be defined, with the corresponding number of SNR pairs. For the simulation, $T_{\rm expose}$ can be defined in the SIMLIB file (§4.7.2) or in the sim-input file (§4.26). SNR($T_{\rm expose}$) is interpolated based on the $T_{\rm expose}$ grid.

The above SPECTROGRAPH file is not read directly by the simulation, but instead it is ingested into a kcor file along with the filters and calibration references. If the above file is named MYSPECDEVICE.DAT, the following keys can be added to a kcor-input file, after a FILTPATH key:

```
SPECTROGRAPH:  MYSPECDEVICE.DAT
#            name    minLam  maxLam  ABoff
SYN_FILTER:  IFU-0    4200    4500    0.0
SYN_FILTER:  IFU-1    6000    6600    0.0
SYN_FILTER:  IFU-2    7200    7600    0.0
```

The SPECTROGRAPH key defines the file containing the noise properties. Optional SYN_FILTER keys define IFU-like synthetic filters from the SPECTROGRAPH.

The SPECTROGRAPH λ-bins can be re-binned in the kcor file as follows,

```
SPECTROGRAPH:  MYSPECDEVICE.DAT(rebin=3)
```

In this example, every three consecutive λ-bins are combined into one, and the SNR values are added in quadrature. With 200 λ-bins in MYSPECDEVICE.DAT, the rebin=3 option results in keeping 66 combined bins. Since $3 \times 66 = 198 < 200$, the last two λ-bins are ignored.

SNR_POISSON_RATIO_ABORT_vsMAGREF ($\mathcal{R}_{\rm Abort}$) is used to check each SNR pair: SNR1 and SNR2. The MAGREF difference is used to predict the SNR-ratio ($R_{\rm SNR}$ =SNR2/SNR1) from Poisson statistics,

$$R_{\rm SNR-Poiss} = 10^{0.2*({\rm MAG1-MAG2})} \ . \tag{3}$$

The kcor program aborts if

$$|\log_{10}[R_{\rm SNR}/R_{\rm SNR-Poiss}]| > \log_{10}(\mathcal{R}_{\rm Abort}) \tag{4}$$

The cut on absolute-value of log conveniently cuts on both $R_{\rm SNR}/R_{\rm SNR-Poiss}$ and $R_{\rm SNR-Poiss}/R_{\rm SNR}$. Note that "R" refers to SNR ratios, while user-input $\mathcal{R}_{\rm Abort}$ is a ratio of R values.

SNR_POISSON_RATIO_ABORT_vsTEXPOSE ($\mathcal{R}_{\rm Abort}$) performs a similar test for SNR vs. exposure time. If the exposure time doubles, the naive Poisson noise expectation is that SNR increases by $\sqrt{2}$. With default $\mathcal{R}_{\rm Abort} = 1.2$, the kcor program aborts if the SNR-ratio differs by more than 20% from the $\sqrt{2}$ expectation.

3.2.1 make_spectrograph_table.py: Convert ETC output into SPECTROGRAPH Table

Spectroscopic instruments typically provide an exposure time calculator (ETC) to estimate SNR vs. wavelength for an arbitrary magnitude. To convert the ETC output into a SPECTROGRAPH table, first translate the ETC output into a simple csv format with 3 columns: "wave flux fluxerr." The flux and fluxerr units don't matter here. Run the ETC for 2 reference mags that roughly correspond to brightest and faintest source, and also for several $T_{\rm expose}$ value in roughly logarithmic bins. If there are separate spectra for blue and red wavelength regions, create separate blue and red csv files. As an example, with 2 reference mags, four $T_{\rm expose}$ values and blue & red components, the total number of ETC-generated files should be $2 \times 4 \times 2 = 16$. To combine these 16 files into a single SPECTROGRAPH table file for SNANA, run the following commands

```
make_spectrograph_table.py -H
 [follow directions to construct config_file that
  includes all ETC-generated csv files ]

make_spectrograph_table.py <config_file>
```

4 The SNANA Simulation: snlc_sim.exe

4.1 Overview of Model Magnitudes and Noise Calculation

The available lightcurve models are described in §9. 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 shape parameter (e.g., Δ, DM15) 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,

$$\sigma^2_{\rm SIM} = \Big[F + (A \cdot b) + (F \cdot \hat{\sigma}_{\rm ZPT})^2 + (\hat{\sigma}_0 \cdot 10^{0.4 \cdot {\rm ZPT_{pe}}})^2 + \sigma^2_{\rm host}\Big]\, \hat{S}^2_{\rm SNR} \qquad (5)$$

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).
- $\hat{\sigma}_{\rm ZPT}$ is the zeropoint uncertainty.
- $\hat{\sigma}_0$ is a constant FLUXCAL uncertainty, and $\rm ZPT_{pe}$ transforms $\hat{\sigma}_0$ into an uncertainty in p.e.
- $\hat{S}_{\rm SNR}$ is an empirically determined scale that depends on the signal-to-nose ratio (SNR)
- $\sigma_{\rm host}$ is from the underlying host galaxy.

The terms with hats ($\hat{\sigma}_0$, $\hat{\sigma}_{\rm ZPT}$, $\hat{S}_{\rm SNR}$) 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 $A, b, \hat{\sigma}_{\rm ZPT}$ terms are discussed in §4.7. The $\hat{\sigma}_0$, $\hat{S}_{\rm SNR}$ terms are discussed in §4.14. The host-galaxy noise ($\sigma_{\rm host}$) is discussed in §4.23.

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 $SNDATA_ROOT/sample_input_files/SALT2/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 generated 100 lightcurves using the SALT2 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 requesting help from other people familiar with `SNANA`.

The simulated lightcurves are located in $`SNDATA_ROOT/SIM`. Do NOT try 'ls' !!! Instead, try '`ls -d *`' to see all versions. To avoid sifting through hundreds of versions from multiple users, use your initials (or project acronym) as a prefix for your versions. Each simulated version has an auto-generated README file. If your version is called 'MYFIRSTSIM', then do

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

which contains a list of all your simulation options. The default sim-data 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`" (§4.37). 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 Search Hierarchy for Supplemental Input Files

The simulation has input options for more than a dozen different maps that are read in as supplemental files. These maps include: cadence, host galaxy, detection efficiency vs. SNR, spectroscopic selection efficiency, etc ... To see all maps in your simulation job, see dashboard utility in §12.3.1. The simulation will search for each supplemental file with the following priority

1. current diretory (i.e., no path appended to file name)
2. path specified by sim-input `PATH_USER_INPUT: <path>`
3. default path under $SNDATA_ROOT.

Priority (1) allows over-riding defaults by simply copying a map file to your local directory. Priority (2) is useful in higher-level pipelines that copy the top-level sim-input file to a spearate directory, but doesn't copy the supplemental files. Priority (3) checks for default files from previous analyses or publications. For example, a SIMLIB file is checked under `$SNDATA_ROOT/simlib`, a k-correction file is checked under `$SNDATA_ROOT/kcor`, trigger files are checked under `$SNDATA_ROOT/models/searcheff`, etc ...

Note that the most robust method is to specify file names with a full path that has an environment (ENV) variable; the ENV ensures that the sim job will run on other computer platforms.

4.4 Cosmological Parameters

The default cosmological parameters are specified in `sntools.h`; as of March 2020,

```
// cosmo params from Planck 2018 (https://arxiv.org/abs/1807.06209)
#define OMEGA_MATTER_LCDM   0.315
#define OMEGA_LAMBDA_LCDM   0.685
#define w0_LCDM            -1.0
#define wa_LCDM             0.0
#define H0_SALT2            70    // km/s/Mpc : tied to SALT2 training
#define H0_MLCS             65    // km/s/Mpc : tied to MLCS training
#define H0_Planck          67.4   // 1807.06209 (Planck 2018)
#define H0_SH0ES           74.03  // 1903.07603 (Riess 2019)
```

The default H0 value is `H0_SALT2`; if an MLCS model is requested, then `H0_MLCS` is used. The `Planck` and `SH0ES` values of H0 are not used in the simulation. For specific analyses requiring different cosmo parameters, use the following sim-input keys:

```
OMEGA_MATTER:  <value>  # default = OMEGA_MATTER_LCDM
OMEGA_LAMBDA:  <value>  # default = OMEGA_LAMBDA_LCDM
w0_LAMBDA:     <value>  # default = -1.0  [legacy W0_LAMBDA accepted too]
wa_LAMBDA:     <value>  # default =  0.0
H0:            <value>  # default = 70(SALT2), 65(MLCS)
  or
HzFUN_FILE:  <fileName>  # non-wCDM:  2 columns with  z  H(z)

MUSHIFT: <value>       # mag shift for all distances (default=0)
MUSHIFT: dmu0:dmu1     # random mag shift between dmu0 and dmu1
```

It is recommended to *NOT* specify cosmological parameters and thus use the defaults; global default updates are easily propagated thru Github rather than modifying numerous sim-input files. The curvature is computed internally as $\Omega_k = 1 - \Omega_M - \Omega_\Lambda$, and the radiation term is ignored.

For non-standard $H(z)$, `HzFUN_FILE` specifies an aribtrary function with 2-columns: z and $H(z)$. The z-bins need not be uniform; i.e., $\log(z)$-binning. Since some of the cosmology functions integrate from $z = 0$, the first map bin must have $z = 0$ (sim aborts if not satisfied).

Finally, MUSHIFT is not a physical model, but may be useful to generate bias-corrections with different MUSHIFT values in order to interpolate distances in a cosmology fitting program. SIM_MUSHIFT is written to the data files, and for non-zero values a SIM_MUSHIFT column is included in the SNANA and FITRES output tables from light curve fitting. SIM_DLMAG is the computed distance modulus, and it is *not* corrected for MUSHIFT or LENSDMU.

4.5 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, HOSTLIB, 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) in each simulation job, and verify that the following output to the README file is identical:

```
Random Number Sync:
 RANDOM SEED: 128473   (RANLIST_START_GENSMEAR: 1)
 FIRST/LAST Random Number (List=1): 0.816247 0.216962  AVG(wrap) = 0.7 +_ 0.2
 FIRST/LAST Random Number (List=2): 0.622194 0.875059  AVG(wrap) = 0.6 +_ 0.1
```

The first list is for the nominal generation, the second list is for the intrinsic scatter models, The optional sim-input key RANLIST_START_GENSMEAR can be used to pick different random numbers for the intrinsic scatter without affecting the main random sequence. Note that RANLIST_START_GENSMEAR is an offset in a list (not a SEED). The AVG(wrap) numbers is the average number of times that each finite list of randoms is re-used each event. Ideally, AVG(wrap)<1 so that randoms are not re-used.

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 (§4.38.2) to verify the sync.

For simulating spectra (§4.26), the number of randoms can be too large for practical storage; e.g., events with several spectra, each with $\sim 10^4$ wavelength bins. A list of randoms that is too small results in obvious spectral ringing. To avoid this problem, two independent streams are used: 1st stream for generation and intrinsic scatter (List=1,2), and 2nd stream for unlimited spectral noise. The number of streams is controlled by sim-input "NSTREAM_RAN: 2" and the default is 2. Only values of 1 or 2 are allowed.

The native C functions needed for two streams result in failed compilation on Macbook, although it works on many other platforms including Fermilab, NERSC, and U.Chicago RCC. For Mac, enable the pre-processor flag

```
#define  ONE_RANDOM_STREAM   // enable for Mac only, in sntools.h
```

in sntools.h, which will force one and only one random stream, and avoid the native functions that don't work on Mac. The default Github release has this flag commented out. Enabling ONE_RANDOM_STREAM should work fine, but the random-sync feature won't work if spectra are generated.

4.6 Simulated TYPE

The simulation produces an SNTYPE value for the data header, allowing specific sub-types to be analyzed.[4] There are two basic SNTYPEs assigned: SNTYPE for spec-confirmed SNe (§4.18), and a different SNTYPE for unconfirmed SNe; the latter correspond to a photometrically identified sample. By default, the integer SNTYPE for unconfirmed SNe is 100 + the SNTYPE of spec-confirmed SNe.

The SNIa-SNTYPE is determined with the sim-input key "SNTYPE_Ia" or "SNTYPES_Ia". For example, the SDSS–II code for type Ia is

```
SNTYPE_Ia:  120       # spec Ia -> type 120; photo-Ia -> type 220
   or
SNTYPES_Ia: 120  106  # spec-Ia -> type 120; photo-Ia -> type 106
```

where the 2nd key allows specifying the photometric-id type to be different than 100 + spec-confirmed type. This integer code appears in the header of each output data file after the "SNTYPE:" key. For spec-confirmed SNIa, the default SNTYPE_Ia value is 1.

The SNTYPE values for NON1ASED and NON1AGRID models are given in the sim-input file as described in §9.7. To specify TYPE for non-SN models, or over-ride the above, use GENTYPE as follows:

```
GENTYPE:  80       # specType=80, photoType=180
   or
GENTYPES: 80 81    # specType=80, photoType=81
```

Beware of SNTYPE collisions ! The user must beware of SNTYPE collisions between SNIa and Non-Ia. For example, consider the example above with "SNTYPE_Ia: 120" in the SNIa sim-input file, and a separate Non-Ia input file in which SNTYPE=20 for one of the species. The unconfirmed Non-Ia SNTYPE value will be 20+100 = 120, which conflicts with the SNIa-SNTYPE value. There is no problem if the SNIa and Non-Ia samples are analyzed separately, but there could be a problem if the samples are combined. There is no way for the simulation code to identify these conflicts because the Ia and Non-Ia are generated separately; hence the user must check.

[4] See &SNLCINP namelist variable SNTYPE_LIST in the snana.exe and snlc_fit.exe programs.

4.7 The 'SIMLIB' Cadence Library File

A user-generated 'SIMLIB' file[5] is needed to define the cadence (MJD, ZP, PSF, SKY), translate magnitudes into CCD counts, and compute the uncertainty as described in Eq. 5. As an example, the start of the SIMLIB file for the SDSS-II 2005 survey is shown in Fig. 4. 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
```

A SIMLIB file is created by someone with knowledge of the telescope and observation conditions. There is a C-code utility, SNANA_DIR/src/simlib_tools.c, that can be used to create the SIMLIB file in the correct format. This utility has a lot of error checking to avoid accidentally writing absurd values such as 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) re-makes a SIMLIB file with all exposures per filter combined into a single effective exposure per night.

The SIMLIB file begins with a required DOCUMENTATION block (§12.3.5). Next is a global header with the following keys:

```
# Required keys:
  SURVEY:  <SURVEY>     # must find match in $SNDATA_ROOT/SURVEY.DEF
  FILTERS: <filtList>   # must find match in kcor/calib file

# Optional keys:
  SKYSIG_UNIT: ADU_PER_SQARCSEC # change SKYSIG unit from default per-pixel
  PSF_UNIT:    ARCSEC_FWHM      # change PSF unit from default pixels
  PSF_UNIT:    NEA_PIXEL        # give Noise-Equiv-Area instead of PSF1,PSF2,PSFRATIO
  NPE_PIXEL_SATURATE: <VAL>     # pixel-flux Saturation, photo-electrons
  PHOTFLAG_SATURATE:  <MASK>    # saturation mask in output PHOTFLAG column
  NLIBID:             <VAL>     # Number of obs-sequences (LIBIDs) in file
  PIXSIZE:            <VAL>     # size of pixel, arcSec
  SOLID_ANGLE:        <VAL>     # sr, used if SOLID_ANGLE=0 in input file
```

For batch jobs submitted with submit_batch_jobs.sh script, NLIBID is used to assign a different starting LIBID (see below) for each batch core, ensuring uniform library sampling.

[5] "SIMLIB" is an abbreviation for SIMulation LIBrary, although "cadence library" is a more appropriate name.

Following the global header is a list of observation sequences. Each sequence starts with a header as follows:

```
# Required
  LIBID: <ID>    # integer id (does not have to be sequential)
  NOBS:  <VAL>   # number of observations
  RA:    <VAL>   # Right ascension, degrees
  DEC:   <VAL>   # Declination, degrees

# Optional
  MWEBV:   <VAL>    # MW E(B-V). If zero, use software options.
  PIXSIZE: <VAL>    # pixel size, arcSec
  FIELD:   <NAME>   # name of FIELDl e.g., DEEP, WIDE, etc
  CCDNUM:  <VAL>    # CCD number (to locate on focal plane)
```

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. 4. Different FIELD names can be repeatedly toggled withing a LIBID, 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.

Following the LIBID header, below is a brief explanation for each column in the observation table, along with references to terms in Eq. 5,

1. **S:** key starts a line with search-image info.

2. **IDEXPT:** arbitrary integer identifier for exposure/visit ID. Related options include:

   - **IDEXPT*NEXPOSE:** Exposure/Visit ID & number of expsoures used to determine saturation (§4.7.4). Default **NEXPOSE**=1.
   - **IDEXPT(DETNUM):** Exposure/Visit and detector/ccd number in ().
     Default **DETNUM**= -9.
   - **IDEXPT*NEXPOSE(DETNUM):** Exposure/Visit & Nexpose & detector/ccd number.

3. **BAND:** 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.

6. **SKYSIG:** Standard deviation of sky (including dark current), in ADU (or DN) per pixel. See §4.15 for noise calculation. You can optionally enter the skysig values per arcsec^2 by specifying the simlib global 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 σ-widths (pixels) of σ_1 = PSF1 and σ_2 = PSF2. **RATIO** refers to the ratio at the origin, PSF2(origin)/PSF1(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. 5 and Eq. 12). If you have computed A from the measured PSF, then you can simply define PSF1 = $\sqrt{A/4\pi}$ and set **PSF2**=**RATIO**=0.

8. **NEA:** Rather than providing 3 PSF parameters, the noise-equiv-area (NEA) can be provided using header key `PSF_UNIT: NEA_PIXEL` (see Fig. 5).

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

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

   For example, if F_{20} is the flux (in ADU) for a point source with 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 §4.20. Note that the zeropoint in photoelectrons is given by $\mathrm{ZPT_{pe}} = \mathrm{ZPTAVG} + 2.5\log_{10}(\mathrm{GAIN})$.

10. **ZPTSIG:** See $\hat{\sigma}_{\mathrm{ZPT}}$ term in Eq. 5.

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.

```
DOCUMENTATION:
    PURPOSE:  SDSS 1st-year cadence for SNANA simulation
    [provide doc in YAML format]
DOCUMENATION_END:
# =======================
SURVEY: SDSS
FILTERS: gri
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 4: Header and part of first entry of the SIMLIB file used for the SDSS-II survey.

```
DOCUMENTATION:
    PURPOSE:  SDSS 1st-year cadence for SNANA simulation
    [provide doc in YAML format]
DOCUMENATION_END:
# =======================
SURVEY: SDSS
FILTERS: gri
PSF_UNIT: NEA_PIXEL  # <== describe PSF with noise-equiv-area (NEA) in pixels
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
#     MJD      IDEXPT  FLT GAIN NOISE SKYSIG  NEA    ZPTAVG ZPTSIG
S: 53616.383  556600405 g  4.05  4.25  4.04  80.38   28.36  0.020
S: 53616.383  556600405 r  4.72  4.25  5.28  61.64   28.17  0.022
S: 53616.383  556600405 i  4.64 12.99  6.95  57.23   27.84  0.017
FIELD: 82S
S: 53622.395  558200552 g  4.03  5.45  4.09  48.43  28.45  0.018
S: 53622.395  558200552 r  4.89  4.65  5.00  41.15  28.15  0.028
S: 53622.395  558200552 i  4.76 10.71  6.43  46.53  27.85  0.029
etc ...
```

Figure 5: Same as Fig. 4, but using NEA column instead of three PSF params.

4.7.1 SIMLIB Options in the Sim-Input File

```
SIMLIB_MSKOPT:  nnn     # bit-mask of options (see below)
SIMLIB_SURVEY:  xyz     # override survey name in SIMLIB file; e.g., rename survey to be
SIMLIB_MINOBS:  nnn     # require at least this many obs per LIBID
SIMLIB_MAXOBS:  nnn     # require fewer than this many obs per LIBID
SIMLIB_IDSTART: nnn     # start at LIBID nnn
SIMLIB_IDLOCK:  nnn     # use only LIBID nnn ; skip all others
SIMLIB_MAXRANSTART: nnn # start at random LIBID among first nnn entries
SIMLIB_NSKIPMJD:  n     # use every 'n+1'th observation
SIMLIB_NSKIPMJD:  2(50),4(100)  # PS=1 for Trest>50, PS=4 for Trest>100 days.
SIMLIB_IDSKIP:  nn1     # ignore LIBID nn1
SIMLIB_IDSKIP:  nn2     # ignore LIBID nn2 (add as many as you want)
SIMLIB_DUMP:      1     # dump summary of simlib
SIMLIB_NREPEAT:  nn     # repeat each LIBID nn times (for speed)
SIMLIB_FIELDLIST xyz    # plus-separated list of field-substrings
SIMLIB_MINSEASON nn     # remove seasons less than nn day duration
USE_SIMLIB_PEAKMJD:  1  # use peakMJD in header (if there)
USE_SIMLIB_REDSHIFT: 1  # use redshift in header (if there)
USE_SIMLIB_DISTANCE: 1  # use lumi-distance in header (invert to get zCMB)
USE_SIMLIB_TAKE_SPECTRUM: 1  # use TAKE_SPECTRUM keys in header
USE_SIMLIB_SPECTROGRAPH:  1  # use SPECTROGRAPH keys under each LIBID
USE_SIMLIB_SALT2:    1  # use SALT2c & SALT2x1 keys in header

SMEARFLAG_FLUX:    1  # apply Poisson noise to fluxes
SMEARFLAG_ZEROPT:  1  # apply ZPTSIG to true flux, but not reported flux
SMEARFLAG_ZEROPT:  3  # apply ZPTSIG to true and reported flux
```

More information about the SIMLIB_DUMP option is in §4.38.1. The SIMLIB_MSKOPT options are

- MSKOPT += 2 : for each LIBID in the simlib, keep generating until an event is accepted. The number generated for each LIBID is stored in the output tables(s) and SIMGEN-DUMP file as NGEN_LIBID. To avoid infinite loop, generation for a given LIBID stops when NGEN_LIBID > MXGEN_LIBID. To preserve NGEN_LIBID information for no-accept LIBIDs, these events are forced to be accepted with NOBS = NEPOCH = 0 to ensure that forced events fails all analysis cuts.

- MSKOPT += 4 : stop generating after one pass through SIMLIB file. Set NGENTOT_LC to large value to avoid stopping before end of SIMLIB. Beware: if APPLY_CUTWIN_OPT is set, the number of events written to data file may be fewer than number of SIMLIB entries.

- MSKOPT += 8 : debug option to replace correlated template noise (TEMPLATE_XXX keys below) with random (uncorrelated) noise.

- MSKOPT += 16 : ignore template noise in SIMLIB file.

- MSKOPT += 32 : ignore FLUXERR_COR map in SIMLIB file.

- MSKOPT += 128 : if any part of $T_{\rm rest}$ range overlaps a season, include entire season in light curve. Season defined by gap ≥ 90 days.

- MSKOPT += 256 : include every MJD in survey, regardless of $T_{\rm rest}$-range.
- MSKOPT += 512 : see§4.7.7

When the trigger efficiency is low, such as for NON1A models at high redshift, the simulation speed is limited by reading the ascii SIMLIB file. The simulation speed can be improved by a factor of few using 'SIMLIB_NREPEAT: 10', where 10 is a suggested value. This option generates 10 SNe with each SIMLIB entry before reading the next SIMLIB entry, and thus reduces the amount of reading by a factor of 10. Be careful to make sure that your SIMLIB is fully sampled. For example, if a SIMLIB has 1000 entries and 1000 SNe are generated with SIMLIB_NREPEAT=10, then the first 100 SIMLIB entries are used and the remaining 900 are ignored. In this situation at least 10,000 SNe should be generated to sample each SIMLIB entry.

SIMLIB_NREPEAT **BEWARE**: this option results in non-uniform sampling of the SIMLIB, and it can be corrected with NGEN_LIBID in the output table. However, if you are not sure how to correct with NGEN_LIBID then you should not be using this option.

For SIMLIB_FIELDLIST, see §4.9.2.

SIMLIB_MAXRANSTART is useful when using submit_batch_jobs.sh (§12.2.1) and each job generates fewer events than the number of SIMLIB entries. For example, suppose a SIMLIB has 1000 entries, and a sim-job generating 2000 events is distributed among 50 cores using submit_batch_jobs.sh. The problem is that each of the 50 jobs generates 2000/50 = 40 events, and each job samples only the first 40 SIMLIB entries, which does not properly sample the SIMLIB. Using

```
SIMLIB_MAXRANSTART: 1000
```

each job will start at a random entry among the first 1000 SIMLIB entries, thereby assuring a proper sampling of the full SIMLIB. If each of the 50 jobs has enough statistics to sample the full SIMLIB, there is no need to use this feature. However, be careful when using SIMLIB_NREPEAT because the number of sampled SIMLIB entries is reduced by this NREPEAT factor.

4.7.2 SIMLIB SPECTROGRAPH

A SPECTROGRAPH instrument is defined in the kcor-input file as described in §3.2. The SIMLIB Search-epoch key for a broadband filter is "S:" and the corresponding SIMLIB key for a spectrum is

```
SPECTROGRAPH:  54997.3  1840   # MJD  & Texpose (seconds)
```

An arbitrary number of SPECTROGRAPH keys are allowed in a SIMLIB file. In addition to computing a spectrum at the listed MJD, the optional synthetic magnitudes (SYN_FILTERS key in kcor-input filers) are evaluated in the same way as any other broadband filter.

Correlated template noise in both the spectra and synthetic filters can be included by specifying the template exposure time in the LIBID header as follows:

```
TEMPLATE_TEXPOSE_SPECTROGRAPH:  5010   # seconds
```

4.7.3 SIMLIB Options for each LIBID

The following header options can follow each LIBID key in a SIMLIB file.

```
   RA:       xxx   # right ascension, degrees
   DECL:     xxx   # declination, degrees
   NOBS:     nnn   # number of obs to follow (i.e., number of 'S:' rows)
   PIXSIZE:  xxx   # size of each pixel, arcsec

#      optional
   MWEBV:     xxx   # E(B-V) from Galactic extinction
   FIELD:     sss   # name of field (optional)
   REDSHIFT:  xxx   # force this redshift
   PEAKMJD:   xxx   # force this peak-MJD

   CUTWIN_REDSHIFT: xxx xxx  # reject if zCMB is not in this range
   REDSHIFT_RANGE:  xxx xxx  # idem with legacy key

   GENRANGE_REDSHIFT: xxx xxx # regenerate zCMB in this range

   GENRANGE_PEAKMJD:  xxx xxx # regenerate PEAKMJD in this range
   GENSIGMA_PEAKMJD:  xxx     # pick from Gaussian profile; otherwise flat

   GENRANGE_SALT2x1:  xxx xxx # regenerate SALT2x1 in this range
   GENSIGMA_SALT2x1:  xxx     # pick from Gaussian profile; otherwise flat

   GENRANGE_SALT2c:   xxx xxx # regenerate SALT2c in this range
   GENSIGMA_SALT2c:   xxx     # pick from Gaussian profile; otherwise flat

#    correlated template noise
   TEMPLATE_ZPT:    <value for each filter>   # ADU
   TEMPLATE_SKYSIG: <value for each filter>   # ADU/pix
   TEMPLATE_CCDSIG: <value for each filter>   # e-/pix
```

If GENSIGMA_XXX is not specified, a flat distribution is generated within GENRANGE_XXX. If the GENSIGMA_XXX are specified, each XXX value is generated from a Gaussian distribution whose peak value is the center of the GENRANGE_XXX window.

For more info on the TEMPLATE_XXX keys, see §4.10. The USE_SIMLIB_XXX keys are specified in the sim-input file (§4.7.1).

4.7.4 Saturation Option

Saturation is specified by two SIMLIB keys in the global header before the "BEGIN LIBGEN" key:

```
NPE_PIXEL_SATURATE:  65000  # Npe to saturate central pixel, per exposure
PHOTFLAG_SATURATE:   2048   # add this to PHOTFLAG for each epoch
```

The central PSF-pixel fluxes are summed for source, galaxy and sky. If the resulting number of photoelectrons exceeds NPE_PIXEL_SATURATE, then this epoch is flagged as saturated as follows:

- PHOTFLAG = PHOTFLAG_SATURATE
- FLUXCAL $= 0 \pm 10^8$

Saturated epochs are not used to form trigger logic, and SNANA programs (snana.exe, snlc_fit.exe) ignore saturated epochs.

For co-added observations, the exposure-averaged flux is compared with NPE_PIXEL_SATURATE. The number of exposures in the co-add (NEXPOSE, default=1) can be optionally included with the IDEXPT column as follows:

```
#                                                PSF1 PSF1 PSF
#     MJD    ID*NEXPOSE  FLT GAIN NOISE SKYSIG (pixels) RATIO  ZPTAVG ZPTERR  MAG
S: 59770.366  13819*6   r  1.00  1.12 138.55 1.45 0.00 0.000  34.89  0.005    99
etc ..
```

which corresponds to 6 exposures in the co-add. Note that ID*NEXPOSE is a single string with no blank spaces between the characters. In this example, a saturated epoch is flagged if

```
FLUX(central pixel)/6 > NPE_PIXEL_SATURATE
```

The fraction of flux in the central pixel (f_A) is computed analytically using a Talyor expansion of a Gaussian:

$$f_A = \frac{P^2}{2\pi\sigma^2}\left[1 - \frac{P^2}{4\pi\sigma^2}\right]$$

where P is the pixel size and σ refers to the PSF. Beware that when $P > \sigma$, the analytic approximation degrades. However, the computation of f_A only affects the saturation flag, and is not used in determining the broadband fluxes.

Finally, the total number of saturated/unsaturated observations can be included in the SIMGEN_DUMP file (§4.38.2): NOBS_SATURATE and NOBS_NOSATURATE. CUTWIN options can be used to select based on the number of [un]saturated observations per filter (§4.19).

4.7.5 APPEND Option

The SIMLIB-APPEND feature prevents epochs from being MJD-sorted. This feature allows appending SIMLIB observations without changing the original light curve fluxes and their Poisson fluctuations. The SIMLIB syntax is shown below. Epochs after the APPEND key are not MJD-sorted, and thus won't change the random sync for epochs before the APPEND key. The integer argument after the APPEND key is a mask that is added to the PHOTMASK column in the data file.

```
...
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
APPEND: 4
S: 53612.395  558200552 g  4.03  5.45  4.09  1.58 3.31 0.107  28.45  0.018
S: 53612.396  558200552 r  4.89  4.65  5.00  1.46 3.55 0.065  28.15  0.028
S: 53612.397  558200552 i  4.76 10.71  6.43  1.53 3.65 0.075  27.85  0.029
...
```

4.7.6 SNR Monitor

To monitor the quality of observing conditions at each epoch, the SNR for a fixed magnitude can be computed with the following sim-input:

```
MAGMONITOR_SNR:  20  # only integer mag allowed
```

which results in computing SNR for mag= 20 at each MJD in the SIMLIB. The results are stored in the data files under a photometry column labelled SIM_SNRMAG20. Beware that SIM_SNRMAG20 depends on the SIMLIB quantities (ZP,PSF,SKY) and does not depend on the light curve model. The argument of MAGMONITOR_SNR must be an integer in order to construct the variable name of the output photometry column.

If SIM_SNRMAG## exists, it is automatically read by the analysis programs and included in output tables with epoch columns: in particular, the tables SNANA+EPOCHS and FITRES+RESIDUALS.

4.7.7 Ideal Cadence with fixed MJD Grid

This option (SIMLIB_MSKOPT | 512) is designed to reduce the necessary size of an ideal SIMLIB cadence that i) has fixed MJD grid, ii) has no seasonal gaps, and iii) has the same SKYSIG/ZP/PSF for every epoch. An ideal cadence is useful to generate model magnitides (and SEDs) without instrumental signatures.

To illustrate the potential problem, consider an ideal 0.1 day cadence for a fast transient such as Kilonova, 10 passbands, and a 5 year (1825 day) survey duration in which we want to randomly generate model light curves. The size of the ideal SIMLIB cadence is $10 \times 1825/0.1 = 182{,}500$ rows, which vastly exceeds the default MXOBS_SIMLIB = 15,000.

A practical solution is to define an ideal SIMLIB with an MJD range that covers the min-to-max $T_{\rm obs}$ range of the model, and shift the resulting light curve to an MJD range outside of the SIMLIB range. This strategy is implemented by settig the 512-bit of SIMLIB_MSKOPT. More specifically, this option does the following:

- defines required MJD range to be
  `( GENRANGE_TREST[1] - GENRANGE_TREST[0] ) * (1+GENRANGE_REDSHIFT[1])`
  The sim aborts if the `SIMLIB` MJD-range is smaller than the required MJD range.

- after picking a random peak-MJD (`PEAKMJD_SURVEY`) within `GENRANGE_PEAKMJD`, a temporary `PEAKMJD_TMP` is selected inside the `SIMLIB` cadence such that (i) `TRESTMIN(1+z)` to `TRESTMAX*(1+z)` is contained, and (ii) the MJD-difference,
  `MJD_SHIFT` = `PEAKMJD_SURVEY - PEAKMJD_TMP`,
  is an integer multiple of the MJD grid size.

- After reading and storing the cadence, all MJDs are shifted by `MJD_SHIFT`.

4.8 Simulating Coordinates and Random Shifts

The RA and DEC coordinates are read from each `SIMLIB` entry (§4.7), which has the advantage of exactly correlating observation properties (ZP,PSF,SKY,MWEBV) with each coordinate. The host galaxy coordinate option (see `MSKOPT+=8` in §4.23.1) enables generating SN light curves to overlay on real images. An artifact of these strategies is that coordinates are repeated for large samples that read through the `SIMLIB` (or `HOSTLIB`) mulitple times. This artifact can be problematic, for example, when using/testing codes that spatially match observations.

To significantly reduce spatial overlaps, there is an option to shift the coordinates to a random location in a circle centered on the originally chosen coordinate,

```
MXRADIUS_RANDOM_SHIFT: 0.01  # circle radius, degrees
```

The assumption is that shifts are $\ll$ 1 deg, and thus `MWEBV` and CMB-redshift are *not* updated. The RA,DEC shifts are not stored in the data files, but the shifts can be monitored via the `SIMGEN_DUMP` utility (§4.38.2), e.g.,

```
SIMGEN_DUMP: CID,ZCMB,DLMAG,RA,DEC,shift_RA,shift_DEC,shift_RADIUS,shift_PHI
```

4.9 Simulating Fields

4.9.1 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 4 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 (§5.20) that work for both data and simulations. First you can pick specific fields with

```
SNFIELD_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.

4.9.2 Field Subset and Field Prescales

There are two distinct methods to simulate a subset of fields, and the difference is important for getting the right normalization of generated events. The FIELD-subset options are illustrated here with a hypothetical survey with 10 fields: 8 of them are called SHALLOW and 2 of them are called DEEP. Assume that each field has the same area of 10 deg^2, or 100 deg^2 total. Finally, consider a SIMLIB file with 200 DEEP-field entries and 800 SHALLOW-field entries. If we select only DEEP fields, there are two different ways in which the simulation can work: 1) count every SHALLOW and DEEP field as part of NGENTOT_LC, or 2) ignore SHALLOW fields as if they were not in the SIMLIB. For the first option, only 20% of the events land on a DEEP field, and thus the efficiency will be less then 20%. For the 2nd option, all events are processed with a DEEP field SIMLIB entry, and thus the efficiency can be as high as 100%. For the 2nd option, one should reduce the solid angle by a factor of 5 in order to get the same number of events as in the first option.

These two SIMLIB-counting options are implemented in the sim-input file as follows:

```
SOLID_ANGLE:        0.03    # 100 deg^2
SIMLIB_FIELDLIST:   DEEP1+DEEP2    # NGENTOT counts both SHALLOW and DEEP
    or
SOLID_ANGLE(DEEP1+DEEP2):  0.006  # 20 deg^2 for DEEP only
```

Used with submit_batch_jobs.sh, both options result in the same number of output events (within statistical errors), but 2nd efficiency will be $\times 5$ larger than the first to compensate the first job generating $\times 5$ more events.

Be careful with SIMLIB_FIELDLIST because it checks for substring matches rather than an exact match. For the example above,

```
SIMLIB_FIELDLIST:  DEEP    # select all fields with DEEP in name
```

is equivalent to explicitly selecting DEEP1+DEEP2. This sub-string select feature is convenient when there are many fields, but it can also lead to undesirable behavior as illustrated in the following example. Consider ten SIMLIB fields: F1,F2,...F10. Naively, the first field is selected with "SIMLIB_FIELDLIST: F1," but it actually selects both F1 and F10 since F1 is a sub-string of both. To avoid such mistakes, define ten fields as F01,F02, ... F10, and select "SIMLIB_FIELDLIST: F01."

To simulate large bias-correction (biasCor) samples for BBC,[6] it may be useful to *prescale* some fields to reduce processing time for simulation and light curve fitting. In the above DEEP+SHALLOW example, suppose that BBC needs few$\times 10^5$ events in SHALLOW and DEEP separately, but the physical distribution results in $\times 4$ more SHALLLOW events. To generate enough DEEP-field events requires over-generating SHALLOW field by a factor of 4. To balance the biasCor load, a prescale can be used as follows,

```
SIMLIB_FIELDLIST:  SHALLOW/4+DEEP
   or
SIMLIB_FIELDLIST:  DEEP+SHALLOW/4
```

[6] BBC = "Beams with Bias Corrections; see §10 and §9.3.1

A few warnings: 1) DEEP fields have a larger redshift range and thus it may be prudent to generate a larger DEEP sample by prescaling shallow by 5 or 6. 2) data/sim overlays using the biasCor will give good agreement only if a FIELD-select cut is applied: will have very poor agreement if all fields are combined. 3) Rather than matching biasCor statistics in BBC, another approach is to match BBC loss per FIELD (see IDSAMPLE losses printed at top of BBC output FITRES files).

Finally, there is a subtle issue using "`SOLID_ANGLE(DEEP)`" as a command line argument. Unix tries to parse the parentheses, resulting in bad arguments passed to the simulation. To avoid this problem, use single quotes or back-slashes:

```
snlc_sim.exe mySim.input  'SOLID_ANGLE(DEEP)' 0.006
   or
snlc_sim.exe mySim.input  SOLID_ANGLE\(DEEP\) 0.006
```

The `submit_batch_jobs.sh` script automatically adds the backslashes, so there is no need to modify the `GENOPT` argument.

4.10 Correlated Template Noise

Coherent template noise is simulated with the following SIMLIB keys:

```
TEMPLATE_ZPT:    <value for each filter>   # ADU      [required]
TEMPLATE_SKYSIG: <value for each filter>   # ADU/pix  [optional]
TEMPLATE_CCDSIG: <value for each filter>   # e-/pix   [optional]
```

where the list of 'NFILT' values corresponds to the list of NFILT filters following the FILTERS key in the header. For example, if 6 filters are defined by "FILTERS: ugrizY" then each TEMPLATE_XXX key must be followed by 6 values, even for LIBIDs that use a subset of filters. The TEMPLATE_XXX keys can appear before the first LIBID to specify a global set of values for every generated SN. Alternatively, these TEMPLATE_XXX keys can appear after each LIBID (along with RA, DECL, etc ...) to specify an independent template noise for each LIBID entry. Note that TEMPLATE_ZPT is required if one or both of the noise keys (SKYSIG or CCDSIG) is specified. For each generated SN epoch, a filter-dependent random template fluctuation is normalized to the search image using the ZPT information, and the normalized fluctuation added to the flux.

Internally the simulation adds two sets of fluctuations: all errors excluding the template are used to pick one fluctuation and the template error is used for the other fluctuation. The first fluctuation is independent for all epochs; the 2nd fluctuation is coherent among epochs in the same filter. The reported FLUXCAL_ERR combines these two sources in quadrature.

```
Off-diagonal errors coming soon ...
```

4.11 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 instruments. The only caveat is the that FIELD name and pixel size must be re-defined as illustrated in Fig. 6.

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 (§4.7), 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.

FIELD: V8    PIXSIZE:  0.339  # <== for different instrument
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 6: Excerpt from SIMLIB with two instruments: DES optical (*griz*) and VIDEO infrared (*YJ*).

4.12 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.

4.13 Simulating a Filter as a Sum of Components

OBSOLETE as of SNANA version v10_50m

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

$$Z_0 = 2.5 \times \log_{10}\left[\sum_i 10^{-0.4(m_i - M_0 - Z_i)}\right] \quad (6)$$

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.

4.14 Noise Corrections: $\hat{S}_{\rm SNR}$ and $\hat{\sigma}_0$

From the noise calculation in Eq. 5 there are two quantities which the simuation cannot determine from the SIMLIB file: 1) global scale $\hat{S}_{\rm SNR}$, and 2) offset $\hat{\sigma}_0$. §4.14.1 explains how a survey team provides this information via tables, and §4.14.3 suggests a general strategy on how to determine $\hat{S}_{\rm SNR}$.

4.14.1 FLUXERRMODEL Tables

$\hat{S}_{\rm SNR}$ and $\hat{\sigma}_0$ are each defined as an aribtrary multi-dimensional function of the following 8 variables:

```
MJD:      Modified Julian date
PSF:      FWHM, arcsec
SKYSIG:   ADU/pixel
ZP:       zero point, ADU
LOGSNR:   log10(SNR)  # calculated SNR before error fudge
SBMAG:    surface-brightness mag, per arcsecond^2
GALAMG:   total galaxy mag
SNSEP:    SN-host separation, arcsec
```

For each observation, the set of 8 values is called $\vec{O}_\sigma$. Several 1D maps can be defined, or a single multi-dimensional map can be used to capture correlations. The 1D maps are illustrated for the 3-year SDSS sample in

```
$SNDATA_ROOT/simlib/SDSS/SDSS_fluxErrModel.DAT
```

The first NVAR-1 columns are input variables among the 8 variables above. The last column is either ERRSCALE ($= \hat{S}_{\rm SNR}$) or ERRADD ($= \hat{\sigma}_0$). The BAND key specifies that each map corresponds only to that band. There is also an optional FIELD key, but since the FIELD key is not used in this example, the map is applied to all fields (82-N and 82-S). Similarly, leaving out the BAND key would apply the map to all filters.

The MAPNAME is chosen by the user, and only one of each MAPNAME can be used per observation. If the simulation finds two valid maps with the sampe MAPNAME, the code will abort. Muliple maps can be applied per observation, as long as each map has a different MAPNAME. In the SDSS example, two maps are applied for each observation: FLUXERR_ADD and FLUXERR_SCALE. For a given $\vec{O}_\sigma$, $\hat{S}_{\rm SNR}$ and $\hat{\sigma}_0$ are computed from linear interpolation of the map. The map binning must be uniform, and the maps are not extrapolated which means that the map range must cover all possible $\vec{O}_\sigma$ values from the survey: violating either criteria results in an abort.

While the example file above (SDSS_fluxErrModel.DAT) illustrates 1D maps, Fig. 7 shows an explicit example of a 2D map for the Dark Energy Survey (DES). This map has 3 ZP bins and 5 PSF bins; the total map size is $3\times5 = 15$ bins. Fig. 7 illustrates an optional feature, DEFINE_FIELDGROUP, to associate a key with a group of fields. The map corresponds to the DEEP fields, which is internally translated to C3 and X3 fields. The map applies to all four (*griz*) passbands.

The maps can be applied to simulations and to data. For simulations, the following sim-input command will use the SDSS set of maps:

```
  FLUXERRMODEL_FILE:  SDSS_fluxErrModel.DAT

# option: suppress map(s) in reported errors for systematic tests:
  FLUXERRMAP_IGNORE_DATAERR:  FLUXERR_ADD
    or
  FLUXERRMAP_IGNORE_DATAERR:  FLUXERR_SCALE
    or
  FLUXERRMAP_IGNORE_DATAERR:  FLUXERR_ADD,FLUXERR_SCALE

# Option: re-write maps in FITRES format (for plotting,analysis, etc...)
# One output text file per MAPNAME
  FLUXERRMODEL_OPTMASK: 256
```

The maps are always applied to the generated fluxes to modify the true scatter on $F - F_{\rm true}$. By default the maps also modify the reported errors, but `FLUXERRMAP_IGNORE_DATAERR` allows suppressing some maps from the reported errors in the data file to study the impact of underestimated errors. The third `IGNORE` key suppresses both maps, which is equivalent to removing the `FLUXERRMODEL_FILE` key.

To modify data uncertainties in the analysis programs (`snana,exe,snlc_fit.exe,psnid.exe`),

```
&SNLCINP
   FLUXERRMODEL_FILE     = 'SDSS_fluxErrModel.DAT'  ! data only
   FLUXERRMODEL_OPTMASK  = 256  ! optional text dump of maps
   SIM_FLUXERRMODEL_FILE = 'SDSS_fluxErrModel.DAT'  ! force on SIMs
```

By default, the `FLUXERRMODEL_FILE` key operates only on real data by adjusting the reported errors. `FLUXERRMODEL_FILE` is ignored for simulated samples because the simulated scatter and errors can be altered during generation. To force simulated errors to be altered in the analysis stage, use the optional key `SIM_FLUXERRMODEL_FILE`.

Finally, the `FLUXERRMODEL_OPTMASK=256` option (for simulation and analysis) writes out the maps in the same TEXT format as the `FITRES` files, and writes one TEXT file per `MAPNAME`. This format may be more useful for making plots and debugging.

```
DEFINE_FIELDGROUP:  SHALLOW   E1+E2+S1+S2+C1+C2+X1+X2
DEFINE_FIELDGROUP:  DEEP      C3+X3

REDCOV: g:0.3,r:0.5  # reduced corr = 0.3 & 0.5 for g & r, respectively
REDCOV: iz:0.4       # reduced corr = 0.4 for i+z combined

MAPNAME:  ERRSCALE_2D_ZPPSF
BAND: griz    FIELD:  DEEP
SCALE_FLUXERR_DATA: 1.2   # scale reported error, but not true err
SCALE_FLUXERR_TRUE: 1.1   # scale true error, but not reported err
VARNAMES:  ZP  PSF  ERRSCALE
ROW:       29  0.5    1.02
ROW:       29  1.5    1.25
ROW:       29  2.5    1.72
ROW:       29  3.5    2.06
ROW:       29  4.5    2.20
ROW:       32  0.5    1.0
ROW:       32  1.5    1.05
ROW:       32  2.5    1.12
ROW:       32  3.5    1.19
ROW:       32  4.5    1.30
ROW:       35  0.5    1.0
ROW:       35  1.5    1.05
ROW:       35  2.5    1.12
ROW:       35  3.5    1.19
ROW:       35  4.5    1.30
ENDMAP:
```

Figure 7: Illustration of `FLUXERRMODEL_FILE` with 2D map to define $\hat{S}_{\rm SNR}$ for DEEP fields in DES. While the bands and fields correspond to DES, this map is made up for illustration and should not be used in any analysis.

4.14.2 Modeling Flux Correlations

The anomolous flux uncertainty may be from a process that is correlated among each epoch: e.g, image-subtraction artifacts. Optional flux correlations in the *anomalous* scatter can be defined with reduced covariances among bands and fields. Since the Poisson noise components are independent, the correlation is introduced only in the anomalous part of the uncertainty. Defining $\sigma_{\rm final} = \hat{S}_{\rm SNR}\sigma_{\rm calc}$, where $\hat{S}_{\rm SNR}$ is the error scale from the map, the simulation breaks the final uncertainty into two components: $\sigma^2_{\rm final} = \sigma^2_{\rm calc} + \sigma^2_{\rm fudge}$. Correlations are introduced among the $\sigma_{\rm fudge}$ components, while the Poisson-$\sigma_{\rm calc}$ components are independent.

Reduced covariances are specified with `REDCOV` keys (Fig. 7). The first key (`REDCOV: g:0.3,r:0.5`) specifies a 30% reduced correlation among g-band epochs, and 50% correlation among r-band epochs; there are no correlations between the g & r bands. This one-line specifier can also be broken into two lines: "`REDCOV: g:0.3`" and "`REDCOV: r:0.5`." The second key (`REDCOV: iz:0.4`) specifies a 40% reduced correlation among the i & z bands. For correlations among all bands, `REDCOV: griz:0.4`.

A field dependence can be specified in parentheses as follows:

```
REDCOV(DEEP): griz:0.6
REDCOV(SHALLOW): g:0.1,r_0.2,i_0.3,z:0.4
  or
REDCOV(C3+X3): griz:0.6
REDCOV(E1+E2+S1+S2+C1+C2+X1+X2): g:0.1,r_0.2,i_0.3,z:0.4
```

where `DEEP` and `SHALLOW` are convenient `FIELDGROUP` definitions in Fig. 7. Note that the `FIELDGROUP`s are defined only in the `FLUXERRMODEL_FILE`, and not in the sim-input file. However, the `FIELDGROUP` names can be used in the sim-input file and command-line overrides.

To vary the correlations without modifying `FLUXERRMODEL_FILE` (e.g., systematic studies using `submit_batch_jobs.sh`), use the following sim-input keys;

```
FLUXERRMODEL_REDCOV:  g:0.3,r:0.5,iz:0.5
  or
FLUXERRMODEL_REDCOV(DEEP): griz:0.6
  or
FLUXERRMODEL_REDCOV: NONE  # disable flux covariances.

FLUXERRMODEL_SNRMIN_REDCOV:  2.0  # default to avoid slow Cholesky decomp
```

For command line overrides and `GENOPT` options for `submit_batch_jobs.sh`, remove the colon. The parenetheses are properly handled by `submit_batch_jobs.sh`, but for command-line overrides quotes are needed, e.g.,

```
snlc_sim.exe <inpFile> 'FLUXERRMODEL_REDCOV(DEEP) griz:0.6'
```

Just as multiple `REDCOV` keys are allowed in the `FLUXERRMODEL_FILE`, multiple `FLUXERRMODEL_REDCOV` keys are allowed in the sim-input file, or as command line overrides.

With very dense sampling over a long survey time, the Cholesky decomposition for the resulting large covariance matrix can be very slow because the decomposition time scales as $N^3_{\rm obs}$. To avoid this potentional CPU bottleneck, and 1) covariance is applied among subset with SNR$>$ 2 (change default with FLUXERRMODEL_SNRMIN_REDCOV), and 2) the `REDCOV` feature is disabled for `LCLIB` (Galactic) and AGN models.

4.14.3 Suggested Strategy for Determining FLUXERRMODEL Tables

For analyses requiring simulated bias corrections, here is a general strategy for determiming the FLUXERRMODEL maps from fake SN-like objects overlaid on images and processed through the same photometry pipeline as the data. We do not make proposals for surveys without fakes.

Two FLUXERRMODEL maps are needed. The first map corrects the simulated scatter to match the scatter in the fake data. Correcting $\sigma_{\rm SIM}$ with $\hat{S}_{\rm SNR}$ (Eq. 5) will correct both the true and reported uncertainties in the simulations. The second map corrects the reported uncertainty in the fake data to reflect the true scatter. This 2nd map is applied in the analysis stage via namelist parameter,

```
&SNLCINP
    FLUXERRMODEL_FILE = 'DataCorMap.dat'
```

This map is applied to the fake data and the real data. It is not applied in the simulation because the simulated true scatter corresponds to the reported uncertainty.

There are two reasons for using fakes: 1) the true flux is known, and 2) the true model and rate are known. The first $\hat{S}_{\rm SNR}$ map is created from both the fakes and the SNANA simulation:

$$\hat{S}_{\rm SNR}(\vec{O}_\sigma) = \frac{\rm RMS[(F_{true} - F)/\sigma_{SIM}]_{fake}}{\rm RMS[(F_{true} - F)/\sigma_{SIM}]_{sim}} \quad (7)$$

where the $\vec{O}_\sigma$-dependence is also on the right-hand side, and RMS indicates root-mean-square in a bin of $\vec{O}_\sigma$. The fakes and SNANA simulation must be generated with exactly the same light curve model and redshift-dependent rate. Any bugs or features used to generate fakes should be reproduced in the simulation. An example of such a feature is interpolating the fake flux on a pre-computed grid in few-day bins; this same interpolation should be used in the SNANA simulation. While Eq. 7 shows a general strategy, each survey team must decide the appropriate subset of $\vec{O}_\sigma$ variables to use, if the maps depend on band and field, and if there are sufficient correlations to motivate multi-dimensional maps. Note that dividing by $\sigma_{\rm SIM}$ (Eq. 5) is optional, but may be useful for debugging since each term should be near unity.

The second map is created soleley from the fakes,

$$\hat{S}_{\rm SNR}(\vec{O}_\sigma) = \frac{\rm RMS[(F_{true} - F)/\sigma_{SIM}]_{fake}}{\langle\ \sigma_{\rm PhotPipe}/\sigma_{\rm SIM}\ \rangle_{\rm fake}} \quad (8)$$

where $\sigma_{\rm PhotPipe}$ is the uncertainty from the photometry pipeline, and $\langle\ \rangle$ indicates an average in the $\vec{O}_\sigma$ bin. If there are flux-outliers, it may be prudent to replace RMS with $1.48 \times$ median.

4.14.4 Extracting Information for FLUXERRMODEL

Here are some suggestions for extracting the 8 variables at the top of §4.14.1. The first suggestion is to simply parse the data file with your own script. The second method is to use the FITRES+RESIDUALS table feature (§12.1) in the light curve fitting program (snlc_fitexe). This feature produces information for each observation in ROOT or HBOOK format. The epoch information can be extracted into TEXT format using the sntable_dump.py utulity (§12.1.3):

```
sntable_dump.py  myFile.root  FITRES  obs
```

A few of the output variables must be converted for the FLUXERRMAP:

```
SBMAG       = 27.5 - 2.5*log10(SBFLUXCAL)  [and protect SBFLUXCAL<0]
sigma_SIM   = ERRTEST * FLUXCAL_DATA_ERR
PSF(arcSec) = PSF(sig,pixels) * 2.35 * PIXSIZE
```

BEWARE that full epoch information is available *only* with FITS format (not with TEXT format).

4.14.5 Legacy Noise Corrections

Here is a legacy correction based on a table near the top of a SIMLIB file. These corrections are still supported, but it is recommended to use the newer and more general flux-error maps in §4.14. Note that if the flux-error maps are used, the SIMLIB noise terms below are ignored.

Global noise-terms $\hat{\sigma}_0$ and $\hat{S}_{\rm SNR}$ (Eq. 5) can be specified in the simlib header. The $\hat{\sigma}_0$ term can be used to account for additional noise from the SN-photometry. $\hat{S}_{\rm SNR}$ is a global correction as a function of the log of the signal-to-noise ratio, $\log_{10}$(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, SNR > 10 results in corrections corresponding to the last FLUXERR_COR row.

To determine $\hat{\sigma}_0$, plot the calibrated flux-uncertainty "FLUXCAL_ERRTOT" 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 the output table for SNTABLE_LIST='SNANA+EPOCH'.

To determine $\hat{S}_{\rm SNR}$, plot ERRTEST vs. $\log_{10}$(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.

4.15 Example Noise Calculation

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

$$\mathcal{N}_{\rm ADU} = N_{\rm pe} \times G^{-1} \times \mathcal{S}_{ZP} \tag{9}$$

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

$$\sigma^2(\mathcal{N}_{\rm ADU}) = \mathcal{N}_{\rm ADU} \times G^{-1} \times \mathcal{S}_{ZP} \tag{10}$$

The sky-background error is computed from

$$\sigma_{skytot} = \mathcal{S}_{ZP} \times \sqrt{A \times (\sigma^2_{skypix} + \bar{\sigma}^2_{skypix})} \tag{11}$$

where σ_{skypix} is the search-run skynoise per pixel, $\bar{\sigma}_{skypix}$ 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[7], the noise-equivalent area is

$$A = \frac{1}{\int PSF^2(r,\theta) r dr d\theta} = \frac{4\pi(\sigma_1^2 + \sigma_2^2)}{1 + 4\pi^2\sigma_1^2\sigma_2^2(h_1+h_2)^2} = 4\pi\sigma_1^2 \left[\frac{(1+R_\sigma^2)(1+R_\sigma^2 r_h)^2}{R_\sigma^2(1+r_h)^2 + (1+R_\sigma^2 r_h)^2} \right] , \tag{12}$$

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, $r_h \to 0$ for any value of R_σ, 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:

[7] 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	(x10^11)
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 if the magnitude (m_i) and zeropoint (Z_i),

$$\texttt{FLUX} = 10^{-0.4(m_i - Z_i)} = 10^{-0.4(23.3402 - 34.56)} = 30755 \text{ ADU} \quad (13)$$

$$\texttt{FLUXCAL} = 10^{-0.4 m_i} \times 10^{11} = 46.13 \quad (14)$$

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 $\sigma_{sig} = \sqrt{N_{pe}} = 175.4$ $p.e.$.

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

4.16 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 [9, 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/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[8] determines a K-correction for arbitrary z,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.F90. 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.

[8] a double precision version of CERNLIB's FINT is used for multi-dimensional linear interpolations.

4.17 Intrinsic Brightness Variations

4.17.1 Supernova 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         # or C11
GENMAG_SMEAR_SCALE:       1.2         # scale all magSmear values
GENMAG_SMEAR_SCALE(c):    0.95,0.3    # scale magSmear by 0.95 + 0.3*c
GENMAG_SMEARPAR_OVERRIDE: BLA  1.234  # optional param override
GENMAG_SMEARPAR_OVERRIDE: BLALIST[3] 1.8  2.2  3.4  # idem
GENMAG_SMEAR_ADDPHASECOR: 0.03 7.0    # magSmear  expTau(days)

# method 4: vary RV with asymmetric Gaussian distribution
GENPEAK_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 wavelength (SALT2 only); see text for details
GENMAG_SMEAR_USRFUN:  <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 with an arbitrarily complex function to describe the smearing as a function of wavelength and/or phase. These functions are in a separately compiled module (`$SNANA_DIR/src/sntools_genSmear.c`), and these functions can in principle be used in non-SNANA applications. The top of `sntools_genSmear.c` provides a list of model-name options. 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. `GENMAG_SMEARPAR_OVERRIDE` allows overriding default smear-model parameters; a list of allowed keys can be seen by grepping the source code,

```
grep  exec_genSmear_override  $SNANA_DIR/src/sntools_genSmear.c
```

Finally, `GENMAG_SMEAR_ADDPHASECOR` is a phase dependent brightness variation that can be combined with `GENMAG_SMEAR_MODELNAME`. The two `ADDPHASECOR` arguments are 1) Gaussian sigma, and 2) coherence parameter (τ) such that for epoch separation ΔT days, the reduced correlation is $\rho = \exp(-\Delta T/\tau)$. Note that $\rho = 1$ for $\Delta T = 0$.

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 $\bar{\lambda}_{\rm obs}/(1+z)$ is applied to the magnitude of the entire passband, where $\bar{\lambda}_{\rm obs}$ 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 λ-nodes separated by `LAMSEP` Å with an initial phase of `LAMPHASE` Å. The 1σ scatter magnitude ($\sigma_{\rm mag}$) at each λ-node (λ_n) is

$$\sigma_{\rm mag} = A_{5500} \times \exp[-(\lambda_n - 5500)/\tau_\lambda] \times \exp[T_{\rm rest}/\tau_{\rm day}] \tag{15}$$

where τ_λ = `TAU_LAM` and $\tau_{\rm day}$ = `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 λ is made by connecting the nodes with cosine functions that ensure that the derivative is zero at each node.

4.17.2 Weak Gravitational Lensing

The effect of weak lensing is simulated using a pre-computed map of lensing probability as a function of redshift and $\Delta\mu \equiv -2.5\log(\text{magnification})$. Maps are stored in \$`SNDATA_ROOT`/`models/lensing`, and the lensing effect is implemented with the following sim-input keys:

```
WEAKLENS_PROBMAP_FILE:  LENSING_PROBMAP_LogNormal+MICE.DAT
WEAKLENS_DMUSCALE:      2.1     # default is 1.0
   or
LENSING_PROBMAP_FILE:  LENSING_PROBMAP_LogNormal+MICE.DAT # legacy key
LENSING_DMUSCALE:      2.1                                # legacy key
```

A MICE simulation [10] is used for $z < 1.4$, and a log-normal approximation [11] is used for $z > 1.4$.[9] The magnification grid extends to about 4 ($\Delta\mu \simeq -1.5$), and thus does not include strong lensing, nor multiple lenses. The distance-modulus (μ) scatter is about $0.04 \times z$, about a factor of 2 smaller than the largest scatter values found in the literature. Input key `LENSING_DMUSCALE` is used to increase or decrease the scatter.

The randomly chosen `SIM_LENSDMU` is stored in the simulated data files, and also stored in the output tables created by the analysis programs. Users should examine and validate the distribution of `SIM_LENSDMU` vs. redshift.

4.17.3 Strong Gravitational Lensing

Strong lensing is simulated by including a library of lenses with sim-input

```
STRONGLENS_FILE:  <fileName>
```

and the library contents are as follows:

```
RESOLVED: 1
VARNAMES: LENSID NIMG ZLENS ZSRC     XIMG          YIMG        MAG       DELAY
LENS:     101    2    0.286 0.68   0.71,+0.02  -0.15,-0.02  0.1,2.2  1.8,5.2
LENS:     102    2    0.328 0.87  -2.85,-0.12  -1.01,+0.08  0.2,2.8  2.2,6.4
LENS:     103    3    0.966 3.39  -.2,-.3,.4   -0.1,-.2,.7  .8,.3,2.  4,2,3
```

where ...

Simulated data files include `SIM_STRONGLENS_XXX` variables, where XXX includes

```
ID z DELAY XIMG YIMG MAGSHIFT NIMG IMGNUM
```

For resolved light curves, each light curve has a unique CID. `SIM_STRONGLENS_ID` has the same integer value for lensed light curves from the same event, and the redshifts are also the same. For unresolved light curves ... TBD ...

The `SIMGEN_DUMP` list can include `STRONGLENS_MAGSHIFT` (or `SL_MAGSHIFT`).

If the number of light curves from a lens exceeds `NGENTOT_LC`, then additional light curves are generated so that all lensed events are included. For example, suppose `NGENTOT_LC` is 100, and on the 99th event there are 4 lensed events. The simulation will generate 102 light curves, not 100, in order to include all of the lensed events for the last SN.

Beware of `GENRANGE_PEAKMJD` clipping events. For example, a common debugging feature is to set `GENRANGE_PEAKMJD` to a delta function (e.g., `MJD0`). For strong lenses, however, the time delays will move each generated `PEAKMJD` away from the generated `MJD0` value. Therefore, zero events would be generated unless some of the time delays are zero. With finite range for `GENRANGE_PEAKMJD`, keep in mind that the time delays will push events outside the generation range.

[9] We thanks Jacobo Asorey for creating this lensing library.

4.18 Search Efficiency

The simulation includes options to model the search efficiency of the survey. The search efficiency is broken into three parts: (i) image subtraction pipelines characterized by efficiency vs. S/N or mag, (ii) spectroscopic efficiency that describes visual scanning and spectroscopic targeting and selection, and (iii) for spectroscopically-unconfirmed SNe, the "zHOST" efficiency for obtaining an accurate host-galaxy redshift. 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 *essential* to always check that the simulated redshift distribution matches that of the data, and to check separately for the spectroscopically-confirmed and unconfirmed sub-samples.

Default efficiency files are stored in the directory $SNDATA_ROOT/models/searcheff, and the file-name includes the name of the survey. However, you can specify an efficiency file in your private directory, or ignore the efficiency file by specifying a file-name as NULL or NONE.

4.18.1 Software-Pipeline Efficiency

The image subtraction pipeline efficiency, $\epsilon_{\rm subtr}$, is based on software algorithms and therefore in principle this part of the efficiency can be rigorously determined. The simplest $\epsilon_{\rm subtr}$ requirement is a minimum number of observations with the sim-input keyword "MINOBS_SEARCH: <MINOBS>" (default is 2). The simulation also parametrizes $\epsilon_{\rm subtr}$ 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

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

and one can add more files corresponding to different surveys. If no SEARCHEFF_PIPELINE file is found, then $\epsilon_{\rm subtr} = 1$. 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_EFF_FILE:  MY_PIPELINE.DAT
   or
SEARCHEFF_PIPELINE_FILE:      MY_PIPELINE.DAT
   or
SEARCHEFF_PIPELINE_EFF_FILE:  NONE   # disable feature
```

overrides the default file or disables this feature. The simulation always checks first if a file exists in your private directory: if not there then the public directory above is checked. The SEARCHEFF_PIPELINE_FILE allows for two kinds of maps as shown in Fig. 8. These maps are very course for illustration, but much finer-grid maps can be used. The first and more common map is the detection efficiency vs. SNR, or MAG or ABS(SNR). The ABS(SNR) option is useful for galactic transients, which can have positive or negative signals in a subtracted image. The map must be contained betweem the keys "MAPNAME_DETECT: <name>" and "ENDMAP:". The example below shows one map valid for *griz* bands, but a separate map for each band can be included.

The PHOTFLAG_DETECT key (Fig. 8) is a mask that is added to the PHOTFLAG column in the data file for each detection. The PHOTFLAG_TRIGGER key specifies the PHOTFLAG mask to add on the first epoch satisfying the trigger logic (see LOGIC file below). In the analysis, setting &SNLCINP input

`PHOTFLAG_DETECT=nnn` results in the following variables added to the SNANA & FITRES tables: NEPOCH_DETECT (number of detections) and TLIVE_DETECT (time between first and last detection).

The second type of optional map is used to specify a PHOTPROB value between zero and one. This quantity is not used in the simulation, but is passed to the output data files so that cuts can be applied. For example, PHOTPROB can represent a machine-learning score from difference-imaging, or from cuts that reject flux-outliers. The map consists of a PHOTPROB distribution in bins of SNR (or LOGSNR) and/or SBMAG (surface brightness), and/or REDSHIFT. The PHOTPROB_WGTLIST is a histogram of events in each bin. The simulation internally determines the cumulative distribution function (CDF) for each SNR-SBMAG bin, and interpolates for each simulated epoch based on the generated SNR, SBMAG, and REDSHIFT. The interpolated CDF is used to generate a random PHOTPROB value for each observation. Finally, this example applies to all fields and bands, but separate maps can be read for each band and/or each field.

The optional REQUIRE_DETECTION flag results in PHOTPROB=0 when there is no detection, and PHOTPROB>0 for detections. By default the PHOTPROB are independent for each observation. Optional PHOTPROB correlations are introduced with the key "REDUCED_CORR: <rho>," where -1 < rho < +1. The example below introduces 50% correlations. If there is a separate map for each band, the correlation between epochs in different bands is the geometric mean of each map-correlation, $\texttt{CORR}_{12} = \sqrt{\texttt{CORR1} \cdot \texttt{CORR2}}$, where CORR1,CORR2 are the reduced correlations. Note that if either correlation is zero, $\texttt{CORR}_{12} = 0$. The correlation method for $N_{\rm PhotProb}$ PHOTPROB values is as follows. First, the Cholesky decomposition technique is used to determined $N_{\rm PhotProb}$ correlated randoms from Gaussians with $\sigma = 1$. Call these correlated Gaussian randoms $G_i,\ i = 1, N_{\rm PhotProb}$. Each G_i is converted to a uniform random between zero and one, $U[0,1]$, from the Gaussian distribution:

$$U[0,1]_i = (2\pi)^{-1/2} \int_{-\infty}^{G_i} dx\ \exp(-x^2/2) \tag{16}$$

where the explicit σ terms are dropped since $\sigma = 1$. The correlations between the G_i are transferred to the $U[0,1]_i$. Finally, $U[0,1]_i$ and the PHOTPROB CDF are used to compute $\texttt{PHOTPROB}_i$:

$$U[0,1]_i = \int_0^{\texttt{PHOTPROB}_i} d\texttt{P}\ \texttt{CDF(P)} \tag{17}$$

Beware that while generating correlated G_i is rigrous and robust, the PHOTPROB correlations are ad-hoc and not rigorous. It is therefore recommended to check the correlations using the NDUMP option, which produces a one-line per event dump to standard-out as follows:

```
777777  [SNID]  [PHOTROB-1]  [PHOTROB-2] . . . [PHOTROB-NDUMP]
```

The "777777" string allows using grep to extract the dump rows. The first NDUMP PHOTPROB>0 values per event are included. If there are not enough PHOTPROB values for the dump, the dump is skipped for the event.

Figure 8: Illustration of SEARCHEFF_PIPELINE_FILE

```
# optional keys to set bit(s) in PHOTFLAG column of data files.
PHOTFLAG_DETECT:  4096  # set this bit for each detection
PHOTFLAG_TRIGGER: 2048  # set this bit on epoch when trigger forms

MAPNAME_DETECT:  EFFDETECT_SNR-g
FILTER: griz
SNR:   0.0  0.0002
SNR:   2.0  0.22
SNR:   4.0  0.45
SNR:   6.0  0.75
SNR:   8.0  0.87
SNR:  10.0  0.98
SNR:  12.0  1.00
ENDMAP:


MAPNAME_PHOTPROB:  MLSCORE_FAKES
BAND: griz   FIELD: ALL  # map applies to all fields and bands
REQUIRE_DETECTION: 1   # generate PHOTPROB only for detections, 0 otherwise
REDUCED_CORR:  0.6     # induce correlation among all PHOTROB per event
CUTVAL:        0.5     # require PHOTPROB>0.5 for detection
NDUMP: 3               # DEBUG: one-line dump of 3 PHOTPROB per event
NVAR:  6               # LOGSNR, SBMAG + 4 PHOTPRPB_WGTLIST bins
VARNAMES:   LOGSNR  SBMAG  PHOTPROB_WGTLIST
ROW:         0.0      20   W11 W12 W13 N14
ROW:         1.0      20   W21 W22 W23 W24
ROW:         2.0      20   W31 W32 W33 W34
ROW:         0.0      22   W12 W12 W13 N14
ROW:         1.0      22   W22 W22 W23 W24
ROW:         2.0      22   W32 W32 W33 W34
ROW:         0.0      24   W13 W12 W13 N14
ROW:         1.0      24   W23 W22 W23 W24
ROW:         2.0      24   W33 W32 W33 W34
ENDMAP:
```

The above SEARCHEFF information gives the probability of a single-epoch detection in a particular filter, but does not specify if the supernova would have been discovered. The *discovery* or *trigger* logic is defined for each survey in the file:

```
more  $SNDATA_ROOT/models/searcheff/SEARCHEFF_PIPELINE_LOGIC.DAT
SDSS: 3 gr+ri+gi  # require 3 epochs, each with detection in two bands.
DES   2 g+r+i+z   # require 2 epochs, any band
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 string that follows defines the logic for a 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 DES logic requires two single detections in any band. The HST discovery logic requires a single detection in filter '6'. As described above, a single detection is defined by the efficiency curves in SEARCHEFF_PIPELINE_[SURVEY].DAT. The final caveat is the epoch time-window used to count a detection. The sim-input key

```
NEWMJD_DIF:  0.4  # days
```

specifies combining all observations within 0.4 days (i.e., one night) into a single detection. For the DES logic, observations of all four *griz* bands in one night would count as one epoch and fail the trigger; an additional observation on a different night is required to pass the trigger. If NEWMJD_DIF is reduced to 0.001 (1.4 minutes), then observations with any two of the four DES filters in one night would pass the trigger.

The pipeline efficiency can be applied in the simulation, or the results can be stored in the data files for future analysis: the user control flag APPLY_SEARCHEFF_OPT is discussed below after the spectroscopic efficiency is discussed.

The LOGIC file can be specified explicitly with sim-input key

```
SEARCHEFF_PIPELINE_LOGIC_FILE:  $SNDATA_ROOT/models/searcheff/SEARCHEFF_PIPELINE_LOGIC.DAT
```

4.18.2 Spectroscopic-Confirmation Efficiency

The human/spectroscopic efficiency ($\epsilon_{\rm spec}$) cannot be rigorously computed since it involves human decision making during the survey. The SNANA simulation parametrizes $\epsilon_{\rm spec}$ as an arbitrary function of redshift, peak-mags, peak colors, and closest time to peak. §4.18.5 suggests how to determine this function from the data and simulations. Rather than using an analytical form for $\epsilon_{\rm spec}$, the simulated efficiency is defined on a multi-dimensional grid of redshift, peak-magnitudes and peak-colors. An example is shown here,

```
SEARCHEFF_SPEC_FILE:  $SNDATA_ROOT/models/searcheff/SEARCHEFF_SPEC_SDSS.DAT
SEARCHEFF_SPEC_SCALE: 0.8  # default=1.0
     or
SEARCHEFF_SPEC_FILE:  ZERO  # force EFF_SPEC=0 without map
     or
SEARCHEFF_SPEC_FILE:  ONE   # force EFF_SPEC=1 without map
     or
SEARCHEFF_SPEC_FILE:  NONE  # force EFF_SPEC=1 without map
```

If this file is not specified, then $\epsilon_{\rm spec} = 1$. For each simulated SN, $\epsilon_{\rm spec}$ 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 $\epsilon_{\rm spec} = 0$ 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 $\epsilon_{\rm spec} = 0$ 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. 9.

Fig. 9 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 "`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. 10.

Figure 9: Illustration of spectroscopic efficiency format for the `SNANA` simulation.

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

# TABLE 2 -- 4m telescope
  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
  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
```

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, there is an overall magnitude offset between different SN models. Second, 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 $\epsilon_{\rm spec} = 0$ from the grid-map. The second problem can be fixed only by using distributions with appropriate tails.

For multiple maps in one SEARCHEFF_SPEC_FILE file, the default logic is "OR", meaning that random selection from any map results in spectroscopic confirmation. This logic accomodates multiple telescopes covering different mag ranges, but is not appropriate for selection *requirements* such as PEAKMJD, DTPEAK, or DTSEASON_PEAK. To force selection from a particular map, add a REQUIRE key before the VARNAMES key as shown in this example:

```
REQUIRE:   1
VARNAMES:  DTSEASON_PEAK  SPECEFF
SPECEFF:   0    1.0   # 100% eff for in-season peakMJD
SPECEFF:  999   1.0
```

which forces spec-confirmed events to have PEAKMJD in-season; i.e., between the first and last observation of a season, where a season is defined as a set of observations with gaps less than 90 days. This map alone would select every in-season event as spec-confirmed, but addition peakmag-dependent maps can be added to the same file.

For each simulated SN, 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, defined as follows:

```
SIM_SEARCHEFF_MASK += 1   # detected by image-subtraction
SIM_SEARCHEFF_MASK += 2   # spectroscopically confirmed
SIM_SEARCHEFF_MASK += 4   # unconfirmed with host redshift (next section)
```

And here are some command examples,

```
SIM_SEARCHEFF_MASK = 1   # detected by image-subtr; spec-unconfirmed
SIM_SEARCHEFF_MASK = 2   # failed image-subtr, but spec-confirmed
SIM_SEARCHEFF_MASK = 3   # detected by image-subtr & spec confirmed
SIM_SEARCHEFF_MASK = 5   # detected by image-subtr, unconfirmed with zHOST
```

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
APPLY_SEARCHEFF_OPT: 5  # keep SN if software trigger and zHOST
```

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.

Finally, SIM_SEARCHEFF_MASK can be specified as a SIMGEN_DUMP variable (§4.38.2) and it appears in the analysis tables (§12.1).

Figure 10: Valid VARNAMES to describe $\epsilon_{\rm spec}$ in Fig. 9

```
* g r i             # peak mag in any band
* PEAKMAG_[band]    # idem for any [band]
* g-r               # peak color
* g-i               # another peak color
* REDSHIFT          # redshift
* PEAKMJD           # MJD at time of peak brightness
* DTPEAK            # closest T-Tpeak; can be pos or negative
* DTSEASON_PEAK     # positive if PEAK is in-season; else negative
* SALT2mB           # SALT2 mB
* SALT2x1           # SALT2 stretch parameter
* SALT2c            # SALT2 color parameter
* LOGMASS           # hsot galaxy LOGMASS
* HOSTMAG_[band]    # mag of host galaxy in [band]
* SBMAG_[band]      # surface brightness mag/arcsec^2 in [band]
```

4.18.3 Unconfirmed Efficiency for Host-Galaxy Redshift

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

```
SEARCHEFF_zHOST_FILE:  <fileName>
    or
SEARCHEFF_zHOST_FILE:  ONE       # Eff(zHOST) = 1.0
    or
SEARCHEFF_zHOST_FILE:  ZERO      # Eff(zHOST) = 0.0
    or
SEARCHEFF_zHOST_FILE:  NONE      # Eff(zHOST) = 1.0
```

where the file contains an efficiency map. There are two kinds of maps: 1) legacy map which depends only on true redshift, and 2) nominal map (starting with v10_70) defining an arbitrary function of HOSTLIB parameters (see "VARNAMES:" key in Fig. 11).

Here we show an example of the redshift-dependent legacy map:

```
# LEGACY SEARCHEFF_zHOST map:
CUTWIN_SNRMAX:  5 1E8  # optional SNRMAX cut on source (not on host)

FIELDLIST:  F1+F2       # optional
HOSTEFF:  0.0     1.0   # required: args are redshift & effic.
HOSTEFF:  0.50    0.8
HOSTEFF:  1.00    0.6
HOSTEFF:  1.50    0.2

FIELDLIST:  F3+F4+F5
HOSTEFF:  0.0     1.0
HOSTEFF:  0.50    0.9
HOSTEFF:  1.00    0.8
HOSTEFF:  1.50    0.7
 etc ...
```

The FIELDLIST arguments are optional, as leaving this out will apply the map to all fields. HOSTEFF keys are required. In analogy with the spectroscopic efficiency, the default survey-dependent file is

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

and similary with 'SDSS' replaced by an arbitray survey name. This feature can be used even if there is no host-galaxy simulation (§4.23).

Next, we given an example of a map with more complex function of HOSTLIB properties:

```
OPT_EXTRAP: 1               # default is 0=>abort; 1=>use value at grid edge
PEAKMJD_RANGE: 55600 58900  # default is 10000 90000
FIELDLIST:  C3+X3           # default is ALL
VARNAMES: g_obs i_obs  HOSTEFF
HOSTEFF:  18  20    1.
HOSTEFF:  18  22    0.95
HOSTEFF:  18  24    0.55
HOSTEFF:  20  20    1.
HOSTEFF:  20  22    0.9
HOSTEFF:  20  24    0.5
HOSTEFF:  22  20    1.
HOSTEFF:  22  22    0.8
HOSTEFF:  22  24    0.4
HOSTEFF:  24  20    0.9
HOSTEFF:  24  22    0.7
HOSTEFF:  24  24    0.2

FIELDLIST:  C1+C2+S1+S2+X1+X2+E1+E2
PEAKMJD_RANGE: 55600 58900
VARNAMES: g_obs  i_obs  HOSTEFF
HOSTEFF:  18  20    1.
HOSTEFF:  18  22    0.98
HOSTEFF:  18  24    0.58
HOSTEFF:  20  20    0.95
HOSTEFF:  20  22    0.85
HOSTEFF:  20  24    0.45
HOSTEFF:  22  20    0.95
HOSTEFF:  22  22    0.75
HOSTEFF:  22  24    0.34
HOSTEFF:  24  20    0.85
HOSTEFF:  24  22    0.65
HOSTEFF:  24  24    0.15
```

OPT_EXTRAP is zero by default, which means that any parameter outside the map range results in an abort. OPT_EXTRAP=1 forces the map-edge value for parameters outside the map range. FIELDLIST is optional for FIELD-dependent maps; C3+X3 means that the map applies to both C3 and X3 fields. Sub-string matching is used, and therefore "FIELDLIST: X" would apply to X1, X2, X3. Also note that FIELDLIST must come before the VARNAMES key. PEAKMJD_RANGE allows for season-dependent maps. Finally, the first $N-1$ VARNAMES specifies arbitrary HOSTLIB propertes from the VARNAMES key in Fig. 11. The last (Nth) VARNAME must be HOSTEFF.

For unconfirmed SNe that have no redshift information, the redshift and its error are set to −9; this value flags the photo-z fitting option in `snlc_fit.exe` to ignore redshift-prior information (`OPT_PHOTOZ=2`, (§5.12). When an accurate host-galaxy redshift is used, the 4-bit is set in `SIM_SEARCHEFF_MASK`. There is a valid (i.e, positive) redshift when `SIM_SEARCHEFF_MASK` has either the 2-bit or 4-bit set. `SIM_SEARCHEFF_MASK=1` means that the SN is detected by the pipeline (§4.18.1) but that there is no valid redshift (`REDSHIFT_FINAL = -9`). Finally, recall from §4.6 that the unconfirmed SNTYPE in the data header is 100 plus the spec-confirmed SNTYPE.

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 PIPELINE+SPEC efficiencies
SEARCHEFF_PIPELINE_EFF_FILE: TESTEFF_PIPELINE.DAT  # private file
SEARCHEFF_SPEC_FILE:         TESTEFF_SPEC.DAT      # private file
```

To simulate a mix of confirmed and unconfirmed SN Ia:

```
APPLY_SEARCHEFF_OPT: 1        # apply only the software trigger
```

Events with `SIM_SEARCHEFF_MASK=3` or 5 have a spectroscopic redshift with uncertainty given by the `GENSIGMA_REDSHIFT` key. Events with `SIM_SEARCHEFF_MASK=1` (unconfirmed and no host spec-z) have a host-galaxy photo-z if such information is availabe in the `HOSTLIB`; otherwise the redshift is set to -9.

To simulate and select events with spectroscopic host redshift,

```
APPLY_SEARCHEFF_OPT: 5        # apply software trigger & zHOST
SEARCHEFF_zHOST_FILE:  SEARCHEFF_zHOST.DAT
```

zHOST VALIDATION: to validate the zHOST efficiency, include `SIM_EFFMASK` in the variable list for the `SIMGEN_DUMP` option (§4.38.2), and use command-line override "`APPLY_SEARCHEFF_OPT 1`." The simulated zHOST efficiency is defined to be

$$\mathrm{EFF(zHOST)} = \frac{\mathrm{N[SIM\_EFFMASK=5]}}{\mathrm{N[SIM\_EFFMASK=1] + N[SIM\_EFFMASK=5]}} ,$$

and note that spectroscopically-confirmed events (`SIM_EFFMASK=3`) are ignored here. `EFF(zHOST)` can be defined as a function of the `HOSTLIB` property(s) and compared against the expected efficiency defined in the file argument of `SEARCHEFF_zHOST_FILE`.

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

4.18.4 Simulating and Fitting Mix of $z_{\rm spec}$ and $z_{\rm phot}$ From Host Galaxy

For future surveys, it may be interesting to consider a mix of redshift sources: spectroscopic host redshifts ($z_{\rm spec}$) measured at lower redshifts, and photometric host redshifts ($z_{\rm phot}$) measured at higher redshifts. This mix-z simulation requires two inputs: 1) `SEARCHEFF_zHOST_FILE` specifying efficiency vs. redshift of acquiring $z_{\rm spec}$ of the host, and 2) `ZPHOT` and `ZPHOTERR` columns in the `HOSTLIB`, or use the `HOSTLIB_GENZPHOT_FUDGEPAR` input (§4.23.1). The relevant sim-inputs are:

```
APPLY_SEARCHEFF_OPT:   1        # apply software trigger only
SEARCHEFF_zHOST_FILE:    SEARCHEFF_zHOST.DAT
```

The `zHOST` efficiency map is used first to determine if the event has an accurate $z_{\rm spec}$. If there is no $z_{\rm spec}$, then the redshift is taken from `ZPHOT` and `ZPHOTERR`. Note that `REDSHIFT_FINAL[_ERR]` in the data file is either $z_{\rm spec}$ or `ZPHOT`, so there is only one redshift variable to access in the analysis. Supplemental data variables `HOSTGAL_ZSPEC` and `HOSTGAL_ZPHOT` are also available, for example, to compare $z_{\rm phot}$ against $z_{\rm spec}$ in the subset for which both are available.

In the light curve fitting stage, use `&FITINP` namelist option `OPT_PHOTOZ=2` to perform a 5-parameter $z_{\rm phot}$ fit with the host redshift as a prior (§5.12). The z-prior is a Gaussian with mean and σ taken from `REDSHIFT_FINAL[_ERR]`. The value of `REDSHIFT_FINAL_ERR` is much smaller for $z_{\rm spec}$ events compared with $z_{\rm phot}$ events. For $z_{\rm spec}$ events, the small σ value results in such a strong z-prior that the fit is essentially a 4-parameter fit with fixed redshift. To validate this process, plot the following from the output `FITRES` table,

```
zHD - SIM_ZCMB vs. SIM_ZCMB
zHDERR vs. SIM_ZCMB
```

Example input files for WFIRST are provided in

```
$SNDATA_ROOT/sample_input_files/WFIRST_zspec+zphot
```

4.18.5 Determining $\epsilon_{\rm spec}$

Since there is no rigorous method to determine $\epsilon_{\rm spec}$, 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 $\epsilon_{\rm spec}(z, \vec{m}, \vec{E})$, 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

$$\chi^2 = \sum_i [({N_{\rm DATA}}^i - {N_{\rm SIM}}^i \times E_0 \epsilon_{\rm spec}(z, \vec{m}, \vec{E}))/\sigma_i^2] \quad (18)$$

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 σ_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 of 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.

4.18.6 Time Above Detection and Number of Detections

In some analyses, it is useful to cut on the time above detection (e.g., looking for fast or long-live transients), and also the number of detections. This section explains how to access this information in the SNANA and FITRES tables via the following variables:

```
TLIVE_DETECT    # MJD_DETECT(last) - MJD_DETECT(first)
NEPOCH_DETECT   # Nobs with detection
```

The detection information must be stored as one of the bits in the PHOTFLAG column of the data files. For real data, the survey team is responsible for setting this bit. For simulations, set the PHOTFLAG_DETECT mask as shown Fig. 8. To get TLIVE_DETECT and NEPOCH_DETECT in the analysis-output tables, set the following analysis input:

```
&SNLCINP
  PHOTFLAG_DETECT = nnn      ! defines detection bt in PHOTFLAG
  CUTWIN_PHOTPROB = xxx, yyy ! optional cut on PHOTPROB, each epoch
```

where nnn is the mask value (e.g., 4096). If PHOTPROB values are included in the data files, CUTWIN_PHOTPROB can be used to reject poor-quality observations. For simulations, default PHOTPROB=1, or a PHOTPROB map can be defined (Fig. 8).

4.18.7 Trigger on Single-Detections instead of Coadd

SIMLIB cadences are typically co-added within a night to achieve optimal search depth. If the ID column of the SIMLIB includes the number of exposures that were co-added (see NEXPOSE in §4.7.4), there is a sim-input flag to trigger on single detections instead of the coadd,

```
APPLY_DETECT_SINGLE: 1
```

This flag results in the following changes in the pipeline-detection:

- SNR $\rightarrow$ SNR/$\sqrt{\mathrm{NEXPOSE}}$ is used to determine the detection efficiency ($\epsilon_{\mathrm{subtr}}$) from the efficiency-vs-SNR map (§4.18.1)
- While SNR is reduced in each exposure, there are NEXPOSE opportunities for detection; thus the final detection efficiency is re-computed as

$$\epsilon_{\mathrm{subtr}} \rightarrow 1 - (1 - \epsilon_{\mathrm{subtr}})^{\mathrm{NEXPOSE}} \tag{19}$$

4.19 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 (§4.38.2). The last option (3) is useful for keeping track of absolute efficiencies, along with the SIMGEN_DUMPALL: key.

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
CUTWIN_PEAKMAG_ALL: 15 30           # ALL filter-peakmag between 15 and 30
CUTWIN_EPOCHS_SNRMIN: 5 20 iz    # SNR(iz) > 5 for < 20 days
CUTWIN_NOBS_SATURATE: 0 3  griz  # require <=3 saturations in each griz band
CUTWIN_NOBS_NOSATURATE: 5 500 griz  # require >=5 unsaturated epochs each band
CUTWIN_REDSHIFT_TRUE:  0 1       # cut on true cmb redshift
CUTWIN_REDSHIFT_FINAL: 0 1       # cut on best redshift (zSpec or zPhot)
CUTWIN_HOST_ZPHOT:     0 1       # cut on host photo-z
```

Each cut-command can be specified in the sim-input file, or using the command-line override (§12.2.3) 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 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.

Adding "CUTMASK to the SIMGEN_DUMP variable list shows which cuts pass/fail each generated event. The CUTMASK bit-definitions are obtained by grepping the source-code,

```
grep CUTBIT $SNANA_DIR/src/snlc_sim.h | grep define
#define CUTBIT_TRESTMAX     0   // (1)
#define CUTBIT_TRESTMIN     1   // (2)
#define CUTBIT_SNRMAX       2   // (4)   max SNR cut
#define CUTBIT_TGAPMAX      3   // (8)   max Tgap
#define CUTBIT_TOGAPMAX     4   // (16)  idem near peak
#define CUTBIT_NOBS_SNR     5   // (32)  NOBS with special SNR cut
#define CUTBIT_NOBS_MJDDIF  6   // (64)  NOBS with MJDDIF cut
#define CUTBIT_MWEBV        7   // (128) galactic extinction
#define CUTBIT_REDSHIFT     8   // (256) redshift
#define CUTBIT_PEAKMAG      9   // (512) peak mag
```

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 (§4.30).

4.20 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 multiples 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"

4.21 Simulating Galactic Extinction

The following sim-input parameters control MilkyWay (MW) Galactic extinction:

```
RV_MWCOLORLAW:        3.1    # default
OPT_MWCOLORLAW:       94     # default: CCM89+ODonnel94
PARLIST_MWCOLORLAW:   p0,p1, # optional: additional parameters
OPT_MWEBV:            1      # default: MWEBV key in SIMLIB file
GENSIGMA_MWEBV_RATIO: 0.16   # default: smear by sig(MWEBV) = .16*MWEBV

APPLYFLAG_MWEBV: 1   # correct FLUXCAL for MWEBV (e.g., for PLASTICC)

# for systematic tests:
GENSIGMA_MWEBV:       0.0      # smear by fixed sig(MWEBV)
GENSHIFT_MWEBV:       0.0      # fixed shift relative to dust map
GENSCALE_MWEBV:       1.0      # scale MWEBV
GENRANGE_MWEBV:  xxx yyy       # genrate flat distrib between xxx and yyy
```

The first five keys control the R_V value, color law, optional color law params (PARLIST), source of MWEBV= $E(B-V)$, and the fractional uncertainty on $E(B-V)$. The remaining keys are intended for additional systematic tests. In `MWgaldust.c`, the function '`GALextinct`' applies the selected color law (based on `OPT_MWCOLORLAW`) to compute the extinction at arbitrary wavelengths, and '`modify_MWEBV_SFD`' determines $E(B-V)$ according to `OPT_MWEBV`. The color laws currently implemented within the '`GALextinct`' function are mostly R_V-dependent extinction laws derived from observations in the Milky Way (§4.21.2). However, there are curves with different parameters, which must be passed as a comma-separated list via the key `PARLIST_MWCOLORLAW`. To see what these curves require, see §4.21.3, or search for `PARLIST` in the '`GALextinct`' function.

To turn off Galactic extinction, set either `OPT_MWCOLORLAW` or `OPT_MWEBV` to zero. Analogous control keys exist in the fitting programs (§5.6), and thus systematic tests can be performed using both simulations and fitting.

Since options may change over time, a robust way to see the current options is to grep the definition keys,

```
grep  OPT_MWCOLORLAW  $SNANA_DIR/src/MWgaldust.h
#define OPT_MWCOLORLAW_OFF              0 // No Extinction applied.
#define OPT_MWCOLORLAW_CCM89           89 // Cardelli, Clayton, & Mathis (1989)
#define OPT_MWCOLORLAW_ODON94          94 // O'Donnel (1994) update
#define OPT_MWCOLORLAW_FITZ99_APPROX  -99 // approx Fitzpatrick (1999)
#define OPT_MWCOLORLAW_FITZ99_EXACT    99 // exact Fitzpatrick (1999)
#define OPT_MWCOLORLAW_GORD03         203 // Gordon et al. (2003)
#define OPT_MWCOLORLAW_FITZ04         204 // Fitzpatrick (2004)
#define OPT_MWCOLORLAW_GOOB08         208 // Goobar (2008) power law
#define OPT_MWCOLORLAW_MAIZ14         214 // Maiz Apellaniz et al. (2014)
#define OPT_MWCOLORLAW_GORD16         216 // Gordon et al. (2016)
#define OPT_MWCOLORLAW_FITZ19_LINEAR -219 // approx Fitzpatrick et al. (2019)
#define OPT_MWCOLORLAW_FITZ19_CUBIC   219 // Fitzpatrick et al. (2019)
#define OPT_MWCOLORLAW_GORD23         223 // Gordon et al. (2023)
#define OPT_MWCOLORLAW_SOMM25         225 // Sommovigo et al. (2025)
```

```
grep  OPT_MWEBV  $SNANA_DIR/src/MWgaldust.h
#define OPT_MWEBV_OFF            0  // no extinction
#define OPT_MWEBV_FILE           1  // FILE value (simlib or data header)
#define OPT_MWEBV_SFD98          2  // use SFD98 value
#define OPT_MWEBV_Sch11_PS2013   3  // PS1-2013 implementation of Schlafly 2011
```

An auxilary table of extinction mag vs. R_V, λ for a few color law options is created with sim-input command:

```
SIMGEN_DUMP_MWCL: 1
```

which creates `[GENVERSION].MWCL` table file in the same directory as the sim data files.

The default keys are indicated above, and these defaults are trivially implemented by running the simulation without specifying any of the MWEBV keys in the sim-input file. With the default `OPT_MWEBV=1`, 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 [12], which is equivalent to `OPT_MWEBV=2`. `OPT_MWEBV>2` are intended for SFD98 recalibrations such as in Schlafly & Finkbeiner [13]. For each simulated SN, the nominal map-value of $E(B-V)$ and its uncertainty are written to the header, and this value is used in the fitting programs (§5.6). `GENSIGMA_MWEBV_RATIO` controls the $E(B-V)$ scatter relative to the nominal map value, and the smeared values are used to redden the true magnitudes in the simulation. If the smearing for a given SN results in negative extinction, the extinction is set to zero.

Finally, the reported `FLUXCAL` are corrected for `MWEBV` if `APPLYFLAG_MWEBV` is set, as was done for the PLASTICC classification challenge. Use caution with this flag to avoid double-correcting in both the simulation and in the analysis.

4.21.1 Some Details on Galactic Extinction Computations

The implementation of Galactic extinction is quite different for observer-frame and rest-frame models, although the sim-input keys work the same for both. The discussion here is relevant for both the simulation and fitting programs.

For observer-frame models (e.g., SALT2, BAYESN, S11DM15) the extinction is computed exactly for each filter-wavelength bin in the flux integrals. For rest-frame models that require K-corrections (e.g., mlcs2k2, snoopy) the Galactic extinctions are stored in the kcor/calib file; the resulting lookup improves CPU speed. In the kcor/calib file, the extinction is computed and stored for $E(B-V) = 0$ and $E(B-V) = 0.1$ mag, and interpolation is used to compute the extinction for arbitrary $E(B-V)$. The treatment of `RV` and `OPT_MWCOLORLAW` is more subtle. These two keys in the sim-input file are also used in the kcor-input file to define the Galactic extinction values. If `RV` and/or `OPT_MWCOLORLAW` are varied in the sim-input file or in the fit-input namelist (§5.6), i.e., are different than what was used to generate the kcor/calib file, an extinction correction is computed at the central wavelength of the observer-frame filter.[10] The correction is the extinction difference between using the sim-input (or fit-input) parameters and those used to generate the kcor/calib file. With this strategy, a new kcor/calib file is *not* required to use a different `RV` value or color law. However, for large variations it would be prudent to make an explicit crosscheck by re-generating another kcor/calib file.

[10] See function `GET_MWXT8` in `snana.F90`.

4.21.2 Milky-Way Extinction Curves

There are a number of extinction laws now present and accessible via the '`GALextinct`' function. The form used is selected by the `OPT_MWCOLORLAW` key. The default option is 99 (`FITZ99_EXACT`(. Also included are variants of the Fitzpatrick law [14, 15], the CCM-like law by Maíz Apellániz et al. [16], the more recent Fitzpatrick et al. curve [17], and a new UV–optical–NIR–MIR curve from Gordon et al. [18]. All of these are R_V-dependent. The '`GALextinct`' function will abort if called outside the valid wavelength range of these models. Some curves with alternative (beyond R_V) parameterizations are included in `SNANA` and described in §4.21.3. Any R_V restrictions are not enforced, but they are defined via the variables `RVMIN_*` and `RVMAX_*` in `$SNANA_DIR/src/MWgaldust.h`.

A Note About F99-like Dust Laws: Extinction laws of the form introduced by Fitzpatrick [14] have a troubled history in `SNANA`. The current implementation of F99 is set by the '`GALextinct_Fitz99_exact`' function, and is based on a "natural" cubic spline that mirrors Fitzpatrick's `FM_UNRED.pro` IDL routine. This is accessed via '`OPT_MWCOLORLAW : 99`', and also forms the basis of several other extinction laws [15, 19, 20]. Prior to September 2024, `SNANA`'s F99 implementation was based on a polynomial fit to the ratio of F99 and O94. This is only reliable for $R_V = 3.1$. For legacy reasons, this remains available via the '`OPT_MWCOLORLAW : -99`' key. Another caveat to keep in mind is that the F99-like curves currently have ∼mmag disagreements with the public `dust_extinction` package [21] due to differences in the spline boundary conditions.

Similar caveats apply to the F19 [17] curve; our implementation uses either a natural cubic spline, or a faster linear interpolation of the tabulated values from table 3 of ref. [17]. The reference implementation in `dust_extinction` uses a B-spline. Our cubic spline implementation agrees with this excellently below 10000 Å, and well (within ~ 2 mmag) below 20000 Å.

4.21.3 Different Extinction Curve Parameterizations

There are several extinction curves that are not neatly parameterized by R_V. For these, the `RV_MWCOLORLAW` simulation input key will be ignored, and a list of parameters will be read from a comma seperated list passed via the `PARLIST_MWCOLORLAW` key. The '`GALextinct`' function should abort if it cannot find valid instances of these parameters, for whatever color law has been requested.

`SNANA` now includes the two-parameter (R_V^A, f_A)-dependent curve from Gordon et al. [19]. This gives a mixture of A-type MW-like [14] and B-type SMC-bar-like [20] dust. The fraction of A-type dust with $R_V = R_V^A$ is set by f_A. The B-type component has $R_V = R_V^B = 2.74$ and no UV bump. `SNANA` will try to read R_V^A and f_A from, respectively, the first and second entries in the list given to `PARLIST_MWCOLORLAW`. The curve generated will have an effective R_V given by $R_V^{\rm eff} = [f_A/R_V^A + (1-f_A)/R_V^B]^{-1}$, where $R_V^B = 2.74$ [19]. This is only valid for $0.0 \le f_A \le 1.0$ and $2.0 \le R_V^A \le 6.0$, and will abort outside these ranges. The space of $R_V^{\rm eff}$ accessible can be seen in fig. 7 of ref. [19].

The SMC bar curve [20] can also be accessed via its own key, although it will abort for $R_V \neq 2.74$.

The two-parameter (p, a)-dependent G08 power-law from Goobar [22] is also available, with power-law index p, and amplitude a being read from the first and second entries in the list given to `PARLIST_MWCOLORLAW`. This form of extinction law is motivated by circumstellar dust, and is defined by $A(\lambda) = A_V \times [1 - a + a \times (\lambda/5495)^p]$. Requires $0.0 < a \le 1.0$ to ensure the resulting $A(\lambda)$ is positive, and $-2.5 \le p \le -0.5$ based on the range of literature values used [22, 23]. Has $R_V^{\rm eff} = [a \times (0.8^p - 1)]^{-1}$, based on eq. 4 in ref. [22]. Ref. [22] quote parameter values of $(p, a) = (-2.5, 0.8)$ and $(-1.5, 0.9)$ for fits to the scattering properties of LMC-like [24] and MW-like [25] dust.

The one-parameter A_V-dependent attenuation curve from the Learning the Universe (LtU)

collaboration [26] is available. This is based on a four-parameter (c_1, c_2, c_3, c_4) fitting function [27, 28], with the four shape parameters being tied to A_V using some scaling relations from the Sommovigo et al. LtU paper [26]. Specifically, $\log_{10}(c_1) = -0.37 \times \log_{10}(A_V) + 0.75$, $c_2 = 1.88$, $c_3 = 1.21 \times \log_{10}(c_1) - 1.33$, and $\log_{10}(c_4) = -0.59 \times \log_{10}(A_V) - 1.42$. The '`GALextinct`' function does not need any extra parameters from `PARLIST_MWCOLORLAW` here, and will ignore the input R_V; the shape of the curve and the amount of extinction are completely set by A_V.

4.22 Simulating Intergalactic Extinction

...

4.23 Simulating the Host Galaxy

The simulation includes a host-galaxy package designed to model the following effects:

- select host galaxy based on arbitrary user-function of host properties; see `WGTMAP` (§4.23.4).
- Generate SN propertes (e.g., stretch & color) correlated with host properties; see `VARNAMES list`.
- two methods to shift SN magnitudes (coherent for all epochs and wavelengths) based on host properties; add `SNMAGSHIFT` to `VARNAMES` list or use `WGTMAP`.
- host-galaxy photo-z to use as photo-z fitting prior (§ 5.12); include `ZPHOT` & `ZPHOT_ERR` in `VARNAMES` list.
- select random SN location (near host galaxy) weighted by surface brightness; see Sersic params and set `HOSTLIB_MSKOPT += 8`.
- host-galaxy contribution to [SN] Poisson noise at each epoch; set `HOSTLIB_MSKOPT += 2`.

The host-galaxy information is stored in a library, hereafter denoted "HOSTLIB". An example HOSTLIB is shown in Fig. 11. This self-documented file contains a list of "`VARNAMES`," and there are two mandatory variables: `GALID` and true heliocentric redshift `ZTRUE` (or `ZTRUE_CMB`;' see §4.23.14) ; 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. 11 are optional, and enable different effects based on `HOSTLIB_MSKOPT`. The `ZPHOT` and `ZPHOTERR` specify an externally-computed photometric redshift, and "[filt]_obs" are the observer-frame galaxy magnitudes needed to compute the Poisson noise. The coordinate variable names can be `RA, RA_GAL or RA_HOST`, and similarly for `DEC`; one of the latter two is recommended to avoid potential conflict with the RA,DEC of the SN in one of the output tables.

The galaxy shape is described by an arbitrary sum of Sersic profiles using the following `VARNAMES`:

```
a0_Sersic - a9_Sersic   # half-light radius, major axis
b0_Sersic - b9_Sersic   # half-light radius, minor axis
n0_Sersic - n9_Sersic   # Sersic index; 0.5, 1, 4 -> Gauss, exp, deVauc
w0_Sersic - w9_Sersic   # weights (if 2 or more profiles per galaxy)
```

The a and b 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`: 1) include in `VARNAMES` list (e.g., see `n0_Sersic` in Fig. 11) to allow a different Sersic index for each host galaxy, and 2) define a fixed Sersic index for all host galaxies using a key after the `VARNAMES` list as shown in Fig. 11. The first Sersic-index method is recommended because the 2nd method may break python readers. Commonly used Sersic indices are $n = 0.5, 1, 4$ for Gaussian, exponential and deVaucouleurs, respectively. If there are 2 or more Sersic profiles per galaxy, weights (`w0_Sersic`-w9_Sersic) are needed; the sum of Sersic weights is 1.

Finally, "`a_rot`" is the rotation angle (in degrees) of the major axis w.r.t. the +RA coordinate. If North is up and East is to the right, `a_rot` is measured clockwise, from the East toward the South.

Figure 11: Example HOSTLIB for the SNANA simulation.

```
VARNAMES: GALID RA_GAL DEC_GAL ZTRUE ZERR  u_obs g_obs r_obs i_obs z_obs
          r_ABS n0_Sersic  a0_Sersic  b0_Sersic  a_rot  ZPHOTFLAG

n0_Sersic: 0.5    # optional: fix sersic index for each host
                  # Note that n0_Sersic can be defined only once;
                  # either among the VARNAMES or here.

GAL:   2940448741  52.67432  -27.17716   0.3964    0.0828    99
       22.06580    20.99990  20.65600  20.33090  -21.43 0.50  0.99290
       0.73811  154.40625  1
GAL:   2940535300  52.67437   -28.42170   0.7860    0.0580    99
       25.78030    23.14940    22.0693   21.4904  -22.98 0.50   1.14660
       0.51003     77.85583  1
etc ...
```

Figure 12: Example HOSTLIB_WGTMAP_FILE for the SNANA simulation.

```
VARNAMES_WGTMAP: LOGMASS   LOG_SFR   WGT  SNMAGSHIFT
WGT:  4.50 -12.5  9.119e-04  0.
WGT:  4.50 -12.3  1.114e-03  0.
WGT:  4.50 -12.1  1.360e-03  0.
etc ...
```

The host-galaxy selection starts by finding a "near-z" subset of host galaxies, where sim-input `HOSTLIB_DZTOL` (§4.23.1) defines the zSN-zGAL tolerance. 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.

4.23.1 HOSTLIB Options

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

```
   HOSTLIB_FILE:         DES.HOSTLIB  # required
# optional
   HOSTLIB_MSKOPT:       8            # default=0
   HOSTLIB_WGTMAP_FILE:   xxxx        # default=none
   HOSTLIB_WGTMAP_EXTRAP: 1           # decault=0
   HOSTLIB_ZPHOTEFF_FILE: xxxx        # default=none
   HOSTLIB_SPECBASIS_FILE: xxxx       # default=none
   HOSTLIB_SPECDATA_FILE:  xxxx       # default=none
   HOSTLIB_MAXREAD:      10000        # default=billion
   HOSTLIB_MNINTFLUX_SNPOS: .20       # default=0.0
   HOSTLIB_MXINTFLUX_SNPOS: .97       # default=0.99
   HOSTLIB_SNR_DETECT:   xxx          # default=0  SNR>xxx for detection (see below)
   HOSTLIB_MAG_DETECT:   xxx          # default=99 MAG>xxx for detection
   HOSTLIB_MAXDDLR:      xxx          # default=4 (for host matches)
   HOSTLIB_MAXDDLR2:     xxx          # default=7 (for HOST_CONFUSION calculation)
   HOSTLIB_SBRADIUS:     xxx          # default=1.2 arcsec
   HOSTLIB_GENRANGE_RA:    xxx yyy    # default= -999 to +999
   HOSTLIB_GENRANGE_DEC:   xxx yyy    # default= -999 to +999)
   HOSTLIB_STOREPAR:  var1,var2,var3  # default = ''
   HOSTLIB_MINDAYSEP_SAMEGAL: xxx     # default=9999999
   HOSTLIB_DZTOL:  a0,a1,a2,,,aN      # default=0.002,0.040
   HOSTLIB_GALID_PRIORITY: xxx yyy    # default= 0 0
   HOSTLIB_GALID_UNIQUE: 0            # default=0; 1-> assign unique GALID
   HOSTLIB_VARNAME_ZPHOT:  ZPHOT_xxx  # use this column for ZPHOT[_ERR]
   HOSTLIB_GENZPHOT_FUDGEMAP: 'z0:rms0,z1:rms1,z2:rms2,etc..' #no pad spaces
   HOSTLIB_GENZPHOT_FUDGEPAR: a0 a1 a2 bprob b1 # default = all -9
   HOSTLIB_GENZPHOT_BIAS:     b0 b1 b2 b3  # 3rd order poly(ztrue)
   HOSTLIB_SCALE_SERSIC_SIZE:  xxx         # default= 1.0
   HOSTLIB_SCALE_LOGMASS_ERR:  xxx         # default= 1.0 (Legacy key)
   HOSTLIB_SCALE_PROPERTY_ERR: 0.8(LOGMASS),0.9(LOGSFR),0.1(LOGsSFR)
                                  # default= 1.0 for all properties
# debug options
   HOSTLIB_GALID_FORCE:    ####       # forced GALID
   HOSTLIB_FIXRAN_RADIUS:  #.##       # fix radial random number (0-1)
   HOSTLIB_FIXRAN_PHI:     #.##       # fix phi random number (0-1)
   HOSTLIB_FIXSERSIC:  a b n a_rot    # a,b in arcsec, a_rot in deg
   HOSTLIB_ABMAG_FORCE:   #.##        # force mag for galmag and host spectrum
   HOSTLIB_ABMAG_OFFSET:  #.##        # add this mag offset to all galmags
```

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. HOSTLIB_ZPHOTEFF_FILE specifies a 2-column file with true redshift and efficiency of finding a host photo-z. For very large HOSTLIB

files, the MAXREAD key may be useful to reduce the initialization time or limit memory usage. The next options (MNINTFLUX_SNOPOS,MXINTFLUX_SNPOS) sets the fraction range of total galaxy flux used to generate the SN position; this option truncates extreme galaxy-SN separations, and can be used to keep events away from the core. HOSTLIB_STOREPAR is a comma-separated list of parameters (case-insensitive, but no blank spaces before or after each comma) to store in the data files; these "STOREPAR" paramaters are automatically propagated into the SNTABLEs and ascii fitres file. HOSTLIB variables defined in SEARCHEFF_zHOST_FILE, GENPDF_FILE, and WGTMAP are automatically included in the HOSTLIB_STOREPAR list.

HOSTLIB_SNR_DETECT works only if both mags and mag-uncertainties are defined in the HOSTLIB as [band]_obs and [band]_obs_err. Setting "HOSTLIB_SNR_DETECT: 5" requires at least one band with SNR>5; an argument of '5,5' requires two bands to satsify SNR>5; '5,5,3' requires at least two bands with SNR>5 and a 3rd band with SNR>3. If the true host fails the SNR cut, the event will either be host-less (no matches), or matched to a different nearby galaxy that passes SNR cut.

HOSTLIB_MINDAYSEP_SAMEGAL specifies the minimum PEAKMJD separation (in days) for re-using a galaxy. This option can be useful, for example, in multi-year surveys with long seasonal gaps, and in very low-z simulations that have a limited number of galaxies.

A redshift tolerance, $dz_{\rm tol} \equiv \max|z_{\rm SN} - z_{\rm GAL}|$, sets a limit on the max difference between the SN redshift and the redshift of the host galaxy picked from the HOSTLIB. This tolerance is needed because for each generated SN redshift there is no host with the exact same redshift. Requiring stricter z-tolerance (and allowing only 1 SN per host) generally requires a larger HOSTLIB to avoid running out of hosts and aborting. The input key HOSTLIB_DZTOL defines $dz_{\rm tol}$ as a polynomial function of redshift: $a_0 + a_1 z + a_2 z^2 ... + a_N z^N$, where $a_{0,1,2,,,N}$ are $N+1$ comma-separated parameters following the HOSTLIB_DZTOL key. The default is $0.002, 0.040$ so that $dz_{\rm tol} = 0.002$ at $z = 0$ and increases to $dz_{\rm tol} = 0.042$ at $z = 1$. To require a fixed 0.01 z-dif tolerance at all redshifts set "HOSTLIB_DZTOL: 0.01." To monitor $z_{\rm SN} - z_{\rm GAL}$, include GALZDIF in the SIMGEN_DUMP list (§4.38.2). The legacy syntax of space-separated a_i values is allowed, but only for a 2nd-order polynomial: HOSTLIB_DZTOL: a0 a1 a2.

HOSTLIB_GALID_PRIORITY allows giving priority to a subset specified by a GALID range. For example, if there only a few real host galaxies with GALID > 0, and the majority are fake galaxies on random sky with GALID < 0, then setting "HOSTLIB_GALID_PRIORITY: 0 99999999" will preferentially select the real galaxies before picking the fake galaxies.

HOSTLIB_GALID_UNIQUE sets a unique GALID for every event, even if the same HOSTLIB galaxy is re-used many times. This feature is useful, for example, in simulated data challenges. The algorithm uses the already unique CID[11] and computes

```
GALID = 57 * CID + int[ ran(0:1) x 57 ]
```

If there are multiple hosts per event and a duplicate GALID, another random GALID is selected. To avoid posible confusion with duplicate host-galaxy mags (e.g., r_obs column), each mag is shifted by a Gaussian-random fluctuation based on the mag-uncertainty (e.g., r_obs_err column).

HOSTLIB_GENZPHOT_FUDGEPAR specifies an analytic description of the host photo-z profile and its error. This FUDGEPAR implemenation does not require ZPHOT and ZPHOTERR keys in the HOSTLIB; if these keys exist in the HOSTLIB, their values are overwritten by the FUDGEPAR description. The ZPHOT-ZTRUE profile is described by a sum of two Gaussian profiles. The first Gaussian (G1) has a width described by the first 3 FUDGEPAR parameters: $\sigma_1 = a_0 + a_1(1+z) + a_2(1+z)^2$. The second Gaussian (G2) is described by the remaining two parameters: 1) bprob is the probability of selecting from G2, and 1-bprob is the prob of selecting from G1, and 2) $\sigma_2 = b_1(1+z)$ is the

[11] To ensure unique CIDs, set "RESET_CID: 2" in the sim-master-input file for submit_batch_jobs.

width of G2. G2 is intended to be broader than G1, and to account for outliers and wrong-host matches. The reported ZPHOTERR is σ_1, and thus G2 introduces non-Gaussian outliers which are not described by the reported uncertainty.

To avoid negative photo-z with FUDGEPAR, the Gaussian distribution is internally modified to an asymmetric Gaussian if the true redshift is less than 3σ away from 0.01. Defining σ_- and σ_+ to be bifurcated Gaussian sigmas, and $z_{\rm peak}$ to be the redshift with max probability, these 3 quantities are constrained by

$$z_{\rm peak} = 0.01 + 3\sigma_- \quad (20)$$

$$\texttt{ZTRUE} = z_{\rm peak} + \left[(\sigma_- - \sigma_+)\sqrt{2/\pi}\right] \quad (21)$$

$$\texttt{ZPHOTERR}^2 = \left[(\sigma_+ + \sigma_-)^2 + (\sigma_+ - \sigma_-)^2(1 - 2/\pi)\right]/4 \quad \text{(variance)} \quad (22)$$

Although an exact solution exists for this quadratic equation, the solution for $z_{\rm peak}, \sigma_-, \sigma_+$ is found numerically. If ZPHOTERR/ZTRUE ratio is too large, a solution cannot be found. In this case of no solution, the mean ZTRUE is preserved while the variance will be smaller than the requested ZPHOTERR^2. The reported ZPHOTERR reflects the calculated variance on the r.h.s of Eq. 22.

If $b_1 \geq 10$, the G2 distribution is flat over the redshift range of the HOSTLIB. To specify a different outlier photo-z range (e.g., $0.01 < z_{\rm phot} < 2.5$), set the b1 (5th) element to the following string: 'FLAT(.01:2.5)'

HOSTLIB_GENZPHOT_BIAS introduces a z-dependent bias on ZPHOT. The parmaeters b0,b1,b2,b3 describe a 3rd order polynomial function of true redshift. This bias applies to HOSTLIB_GENZPHOT_FUDGEPAR or to the ZPHOT[ERR] columns in the HOSTLIB.

HOSTLIB_SCALE_SERSIC_SIZE scales the a0 and b0 half-light radii to make the galaxy sizes bigger or smaller.

HOSTLIB_SCALE_LOGMASS_ERR scales the LOGMASS uncertainty.

HOSTLIB_SCALE_PROPERTY_ERR scales the error of the property in parenthesis as illustrated above.

The HOSTLIB_MSKOPT options can be viewed with the grep-command

```
grep MSKOPT $SNANA_DIR/src/sntools_host.h
```

- MSKOPT += 2 : compute the host-galaxy Poisson noise within an aperture of radius $2\sigma_{\rm PSF}$ (added to total error per epoch), and also compute the local surface brightness.[12] This option requires defining HOSTLIB header keys "f_obs" for each filter (see VARNAMES in Fig. 11), where 'f' is the one-character filter representation. These observer-frame HOSTLIB mags should be corrected for MW Galactic extinction and correspond to the entire galaxy flux.
- MSKOPT += 4 : apply SN mag shift from WGTMAP. Does not control SNMAGSHIFT in VARNAMES list.
- 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. SN-generated $z_{\rm cmb}$ is preserved; $z_{\rm hel}$ is updated to account for new coordinates and generated $v_{\rm pec}$. Generated distance (μ) is also updated.
- MSKOPT += 16 : adjust the SN redshift, SN mags, and μ to galaxy redshift, ZTRUE (useful for image simulations). $z_{\rm hel}$ = ZTRUE; $z_{\rm cmb}$ and μ are updated to account for new $z_{\rm hel}$ and generated $v_{\rm pec}$.

[12] see "HOSTGAL_SB_FLUXCAL" key in data header: units are FLUXCAL per arcsec^2.

- `MSKOPT` += 32 : allow only one SN per host galaxy (results in abort if library is too small)
- `MSKOPT` += 64 : use SN properties in hostlib to override default generation: c, x1, Delta, AV, RV. Case-insensitive so that x1 and X1 both work. Sim aborts if no SN parameters are found.
- `MSKOPT` += 128 : set `ZTRUE = ZPHOT` (e.g., study impact of `HOSTLIB` based on photo-z).
- `MSKOPT` += 256 : increase init verbosity to screen.
- `MSKOPT` += 512 : use peculiar velocity columns in `HOSTLIB` (`VPEC,VPEC_ERR`). Overrides sim-input keys `GENSIGMA_VPEC` and `VPEC_ERR`.
- `MSKOPT` += 1024 : detailed screen dump for each event (for small tests only)
- `MSKOPT` += 32768 : force Gaussian PDF (debug test) for quantiles with mean,sigma = `ZPHOT,ZPHOTERR`. Overwrite quantile values in `HOSTLIB`, or force 10 quantile redshifts.

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.

4.23.2 Poisson Noise from Host

If the 2-bit of `HOSTLIB_MSKOPT` is set, the noise covariance from the host ($C_{\rm host}$) is computed as the PSF-convolved galaxy flux contained within the noise-equivalent-area (NEA) centered on the Supernova,as illustrated in Fig. 13;

$$C_{\rm host} = \int_{\rm host} \left[\int_{\rm NEA} F_{\rm host}(\rho_g, \theta_g) PSF(x - x_g, y - y_g) dx dy \right] \rho_g d\rho_g d\theta_g \qquad (23)$$

where ρ_g is the reduced radius[13] θ_g is the azimuthal angle, and $F_{\rm host}$ is the host Sersic profile flux at coordinate $\{\rho_g, \theta_g\}$ (w.r.t. center of host). The inner PSF integral is over the circular NEA region; for computational speed, the Gaussian-PSF integrals are pre-computed and stored on a grid as a function of $R = \sqrt{(x - x_g)^2 + (y - y_g)^2}$, where $\{x_g, y_g\}$ are Cartesion coordinates converted from polar coordinates. The outer integral is over the entire host galaxy with 100 ρ_g bins and 36 θ_g bins. To avoid computing this integral for each observation per event, $C_{\rm host}$ is pre-computed on a grid of about 10 `PSFSIG` values at the start of each generated event, and interpolated at the `PSFSIG` value of each observation.

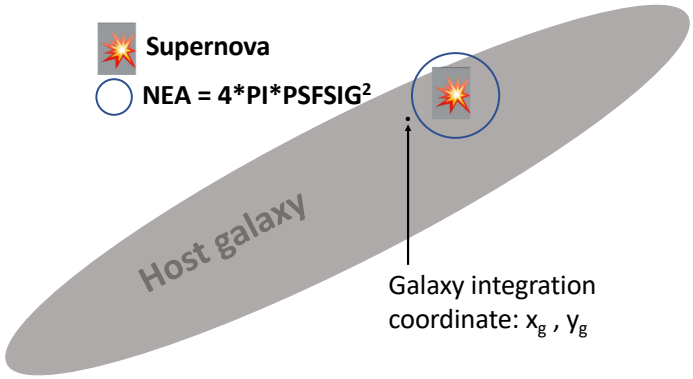

Figure 13: Diagram of host galaxy, SN, NEA and coordinates used to compute Poisson noise.

[13] ($\rho_g \equiv R/R_{1/2}$, where $R_{1/2}$ is the half-light radius.

4.23.3 Hostgal properties

In the HOSTLIB columns you can define true, observed, and error for up to four properties;

```
LOGMASS  LOGMASS_TRUE  LOGMASS_OBS  LOGMASS_ERR
LOGSFR   LOGSFR_TRUE   LOGSFR_OBS   LOGSFR_ERR
LOGsSFR  LOGsSFR_TRUE  LOGsSFR_OBS  LOGsSFR_ERR
COLOR    COLOR_TRUE    COLOR_OBS    COLOR_ERR
```

Item in the left column is the "BASENAME," and defining a HOSTLIB column as BASENAME is interpreted as BASENAME_TRUE. If BASENAME_ERR is specified, BASENAME_OBS is computed as Gauss(BASENAME_TRUE,BASENAME_ERR). Non gaussian errors can be simulated with BASENAME_OBS instead of BASENAME_ERR. BASENAME_ERR can be rescaled using the sim-input key

```
HOSTLIB_SCALE_PROPERTY_ERR: 0.0(LOGMASS),0.0(LOGSFR) # set errors to zero
    or
HOSTLIB_SCALE_PROPERTY_ERR: 0.8(LOGMASS),0.9(LOGSFR),0.5(LOGsSFR)
```

where the first example sets errors to zero so that observed property equals the true value, and second example scales the error on 3 properties. These host-property values are automatically propagated to the simulated data files, and after light-curve fitting they are propagated to output tables (e.g. SNANA, FITRES).

4.23.4 HOSTLIB Weight Map (WGTMAP)

Ideally the simulation would select a random galaxy in the universe, and then select SN properties based on the host galaxy properties. In practice, however, the reverse is more practical: SN properties are picked from a measured population, and after the SN is selected a host galaxy is assigned. Using the HOSTLIB properties, an optional "weight map" (WGTMAP) defines a relative probability for assigning a host galaxy. The redshift is matched based on a tolerance (see HOSTLIB_DZTOL key), and the WGTMAP is applied among all galaxies within the redshift tolerance. An example HOSTLIB_WGTMAP_FILE is shown in Fig. 12. Any HOSTLIB parameter may appear before the WGT column, and the map dimension can be arbitrarily high provided that your computer has sufficient memory to hold the entire weight map. An important constraint is that binning in each dimension must be uniform.

Following the VARNAMES_WGTMAP key, the first "$N-2$" variables must be from the VARNAMES list in Fig. 11, which describe an arbitrary dependence on host properties. The last two variables must be WGT and SNMAGSHIFT, where the latter is applied to the generated magnitudes if the 4-bit is set in HOSTLIB_MSKOPT. All WGTMAP variables are automatically included in the data files, equivalent to internally appending these variables to HOSTLIB_STOREPAR. While there is no need to specify the same variable in both the WGTMAP and HOSTLIB_STOREPAR list, there is no harm in doing so.

If no weight map is provided, the simulation will assign a weight of unity to each host galaxy, with zero mag-shift. Starting with SNANA version v10_70, the "NVAR:" and "NVAR_WGTMAP:" keys are obsolete, although leaving them in the HOSTLIB file will not cause any harm.

To include WGTMAP correlations with SALT2 streth (x1) and color (c), simply include x1 and/or c in the WGTMAP. These are the only WGTMAP variables that are not required to be in the HOSTLIB VARANMES list. If x1 and c are not in the HOSTLIB, the generated value is snapped to the nearest WGTMAP grid point so that the simulation runs more efficiently. For example, consider a WGTMAP color grid from -0.3 to $+0.5$ in bins of 0.1; for a generated color of $c = 0.12$, the weight is the pre-computed value at the grid point $c = 0.1$.

If a HOSTLIB value is beyond the defined range of the WGTMAP, the simulation aborts by default. To avoid the abort and fix out-of-range values to the WGTMAP edge,

```
HOSTLIB_WGTMAP_EXTRAP: 1
```

MEMORY WARNING: To maintain simulation speed, the WGT and SNMAGSHIFT are precomputed and stored in memory for each galaxy and each x1,c grid point; beware that this memory storage can be large. For example, consider a HOSTLIB with a million galaxies and a WGTMAP that includes a 20×20 "x1 $\times$ c" grid. The WGT and SNMAGSHIFT are stored for $20 \times 20 \times 10^6 = 400$ million values, consuming 4 Gb of memory.[14] Check memory usage in the stdout from this output:

```
Interpolate WGTMAP for each galaxy (68.47 MB)
```

[14] While WGT is stored as 8-byte double, SNMAGSHIFT*20000 is stored as 2-byte short int to reduce memory.

4.23.5 Generating Host Spectra

Host spectra are generated with the SPECTROGRAPH feature, and with several inputs as follows:

- HOSTLIB_SPECBASIS_FILE points to a file with spectral basis such as PCA. The header keys must include
  VARNAMES: ROW WAVELENGTH SPECBASIS00 SPECBASIS01 SPECBASIS02 ... etc,
  where each vertical SPECBASIS column is an SED basis. Following VARNAMES, a ROW: key specifies λ and $dF/d\lambda$ for each wavelength bin.
  Optional keys before the header are:
  ```
  FLAM_SCALE: 1.55E-30    # global flux norm.
  FLAM_SCALE_POWZ1: -2    # z-norm: 1/(1+z)**2
  ```
  If using EAZY SED templates, reformatting to SNANA-table format can be avoided with key "EAZY_TEMPLATES_LIST_FILE: <list_file>" that points to the template-list file in the EAZY repository. With this EAZY-direct read, leave out the VARNAMES and ROW keys.
- HOSTLIB_SPECDATA_FILE points to a file with spectral data. The header keys must include VARNAMES: ROW WAVELENGTH SPECDATA00 SPECDATA01 ... etc, where each vertical column is an SED. Optional keys before the header are the same as for SPECBASIS.
- HOSTLIB_FILE that includes columns COEFF_SPECBASIS00, COEFF_SPECBASIS01, etc.. or for specdata there is a single IDSPECDATA column to pick a specific specdata column from the SPECDATA_FILE.
- TAKE_SPECTRUM key with HOST argument as shown in §4.26.2.

For a given hostlib entry, each basis spectrum "specbasis[nn]" is multiplied by coeff_specbasis[nn], and all of the spec-basis terms are summed. The sum is multipled by FLAM_SCALE and the z-norm term.

4.23.6 Computation of DDLR (SNSEP/DLR)

The Sersic parameters in the HOSTLIB are assumed to represent the true galaxy profile, without PSF smearing, and this profile is used to overlay a true SN location near the host galaxy. To compute DDLR=SNSEP/DLR, a default DLR is computed from the true Sersic profile, which does not account for PSF smearing. There are two approximate methods to account for PSF smearing in the DLR computation:

1. define HOSTLIB columns a_DLR and b_DLR based on measured half-light radii. These quanitieis are used only to compute DLR, but are not used to overlay a true SN location.
2. Use sim-input "HOSTLIB_SMEAR_SERSIC: 1.0" which is loosely interpreted as FWMH (arcsec) for average PSF. This parameter does not impact placement of SN near galaxy, but it affects DLR; defining $ab_{\rm smear} \equiv$HOSTLIB_SMEAR_SERSIC$/2$,

   $$a \to \sqrt{a^2 + ab_{\rm smear}^2} \qquad b \to \sqrt{b^2 + ab_{\rm smear}^2} \tag{24}$$

   where a, b are the semimajor/minor half-light radii.

4.23.7 Generating HOST-less Events

The following sim-input keys control the fraction of simulated events reported by HOSTGAL_NMATCH=0; i.e., "host-less."

```
  # keys that require Sersic profile in HOSTLIB
   HOSTLIB_MAXDDLR:          4.0       # SEP/DLR
   HOSTLIB_MXINTFLUX_SNPOS: 0.94       # default=0.99

  # keys that require [band]_obs and [band]_obs_err in HOSTLIB
   HOSTLIB_SNR_DETECT:  5    # or 5,5 -> 2 bands with SNR>5
   HOSTLIB_SNR_SCALE:   0.5  # reduce by 2 -> impacts SNR_DETECT
       or
 # PEAKMJD-dependent SNR_SCALE
   HOSTLIB_SNR_SCALE(53000:53365): 0.3
   HOSTLIB_SNR_SCALE(53365:53720): 0.5
   HOSTLIB_SNR_SCALE(53720:54050): 0.7
```

The first key (MAXDDLR) defines host-less events based on the separation from the host. Very broad profiles (e.g., de Vaucouleur with $n = 4$) can lead to excessive host-less fractions from large DDLR. To truncate the galaxy profile, MXINTFLUX_SNPOS removes the high-probability region at large radii.

The last two keys simulate host-less events by excluding nearby galaxies which fail a user-defined SNR cut. Galaxy mags and uncertainties are required in the HOSTLIB for this feature. In the example above, at least one band must satisfy SNR> 5.

A HOSTLIB is often based on the full depth of a survey, and is thus not appropriate for simulating samples before the end of the survey. For example, consider a HOSTLIB based on a 4-year co-add depth; to simulate a 1 year co-add depth, set SNR_SCALE: 0.5 to reduce the galaxy SNR by a factor of $\sqrt{1/4} = 1/2$. An optional PEAKMJD-dependent SNR_SCALE can be given by providing PEAKMJD-ranges as parenthetical arguments.

4.23.8 Generating Host Galaxy Photometric Redshifts

Photometric redshifts are read from the HOSTLIB. The $z_{\rm phot}$ values are not used in the simulation for calculations, but are simply written to the data files for analysis. There are two ways to provide photo-z information into the HOSTLIB. First, provide ZPHOT and ZPHOTERR columns. The second method is to provide 11 quantile redshift columns corresponding to integrated (CDF) probabilities of 0, 10%, 20%, ... 100%:

```
ZPHOT_Q000  ZPHOT_Q010   ZPHOT_Q020   ZPHOT_Q030 ... ZPHOT_Q100
```

One or both methods may be provided in the same HOSTLIB. A more general framework for arbitrary quantiles may be developed in the future. The corresponding output data variables are

```
HOSTGAL_ZPHOT_Q000  HOSTGAL_ZPHOT_Q010
HOSTGAL_ZPHOT_Q020  HOSTGAL_ZPHOT_Q030 ... HOSTGAL_ZPHOT_Q100
NZPHOT_Q
```

where NZPHOT_Q=11.

To select photo-z in the light curve fitting program, set &FITINP namelist variable OPT_PHOTOZ=4 (§5.12). To override photo-z information in the data files (real or sim), see (§5.28.1). Note that quantile overrides are unique because they can be appended even if the original data files do not have quantile data.

4.23.9 Compute Synthetic HOSTLIB Magnitudes (+HOSTMAGS)

Using HOSTLIB_SPECBASIS_FILE, synthetic galaxy mag is computed for each band in "GENFILTERS:" key using the command-line option

```
snlc_sim.exe <iputFile> +HOSTMAGS
```

A new HOSTLIB file is created with [band]_obs columns appended, and then the simulation stops execution. Note that +HOSTMAGS is not read from the sim-input file, and thus can be specified only as a command-line argument. This new HOSTLIB can be used to generate with galaxy mags, compute additional Poisson noise, and model analmolous flux-scatter as a function of local surface brightness.

4.23.10 Generate List of HOSTLIB Neighbors (+HOSTNBR)

To simulate the effect of matching SN to the wrong galaxy with the DLR method, a list of true galaxy neighbors is needed for each HOSTLIB entry. For a HOSTLIB that includes sky coordinates, a supplemental column with a comma-separated list of neighbors can be created with

```
snlc_sim.exe <inputFile> +HOSTNBR
        or
snlc_sim.exe <inputFile> +HOSTNBR   SEPNBR_MAX 10.0  NNBR_WRITE_MAX 10
```

The sim-input file must include a HOSTLIB_FILE key, but no other sim-input keys are required. The optional command-line arguments are 1) SEPNBR_MAX, the maximum angular separation between galaxies (default: 10″), and 2) NNBR_WRITE_MAX, the max number of neighbors to include (default: 10). These commands work only on the command line, and do not work as sim-input keys. A new HOSTLIB is created with +HOSTNBR extension, and the simulation quits without generating events. If Sersic profiles are included, running the simulation again with the newly-created HOSTLIB results in computing DLR for the true galaxy and each neighbor galaxy, and ordering the galaxy matches by DDLR = DLR/Angsep, even if the smallest DDLR is the wrong match.

To find the galaxy neighbors quickly when using the output HOSTLIB+NBR in a simulation, the galaxy identifiers are HOSTLIB row numbers (start at row 1) rather than GALID. To allow for easy verification, the +HOSTNBR option includes a screen-dump for several entries, where the dump includes both row number and GALID.

SIM_HOSTLIB_GALID is the true galaxy ID, and is in output data files and output tables created by analysis codes. An incorrect host-match is flagged by SIM_HOSTLIB_GALID≠HOSTGAL_OBJID; in this case, it is likely that SIM_HOSTLIB_GALID = HOSTGAL2_OBJID. To reject host-matches with a DDLR cut (§4.23.1), set sim-input key "HOSTLIB_MAXDDLR: <cutval>." For analysis, host information can be added to TEXT-formatted tables (§12.1) using

```
SNTABLE_LIST = 'SNANA(text:host) FITRES(text:host)'
```

4.23.11 Append Arbitray Columns to HOSTLIB (+HOSTAPPEND)

Contents from a supplemental HOSTLIB file can be appended to a primary HOSTLIB with

```
snlc_sim.exe <inputFile>  +HOSTAPPEND  <hostlib_append_file>
```

The sim-input file must include a HOSTLIB_FILE key, but no other sim-input keys are required. The supplemental hostlib_append_file must include a VARNAMES key that begins with GALID, and the GALID values should match those in the primary HOSTLIB. Each row should begin with "GAL:" key. The sim program ends after writing a new HOSTLIB, and warnings are printed for the number of missing GALIDs.

4.23.12 Treatment of LOGMASS

To propagate effects of host-galaxy mass uncertainty, the simulation treats the following HOSTLIB columns:

```
LOGMASS_TRUE    # true log10(M/Msolar)
LOGMASS         # same as above
LOGMASS_ERR     # Gaussian uncertainty
LOGMASS_OBS     # measured log10(M/Msolar)
```

If LOGMASS_OBS is not included, then the measured mass is determined from a random Gaussian fluctuation using LOGMASS_ERR. LOGMASS_OBS is propagated to the output and used in cosmology fitting. The sim-input parameter HOSTLIB_SCALE_LOGMASS_ERR scales the uncertainty (see Table near top of §4.23.1).

4.23.13 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

$$\mathcal{S}_n(\rho) = 2\pi \int_0^\rho \exp\left[-B_n(x^{1/n} - 1)\right]\, x dx \tag{25}$$

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 $x_{\rm gal}$ and $y_{\rm gal}$ is the sum over Sersic components,

$$F = \sum_n w_n \frac{\exp\left[-B_n(\rho^{1/n} - 1)\right]}{a_n b_n \mathcal{S}_n(\infty)} \tag{26}$$

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 $\rho^2 = (x_{\rm gal}/a_n)^2 + (y_{\rm gal}/b_n)^2$ is the reduced radius.

4.23.14 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≥ 1. If there is no host-galaxy photo-z, GENSIGMA_REDSHIFT≥ 1 results in undefined REDSHIFT_FINAL$= -9 \pm -9$.

4.23.15 ZTRUE_CMB Column instead of Heliocentric ZTRUE

The HOSTLIB redshifts can be specified in the CMB frame using a ZTRUE_CMB column instead of the default ZTRUE column. For this ZTRUE_CMB feature, the HOSTLIB coordinates (RA_GAL, DEC_GAL) are used to transform ZTRUE_CMB back to heliocentric ZTRUE during initialization. Therefore, this CMB-frame feature works only with the HOSTLIB_MSKOPT += 8 option to use HOSTLIB coordinates intead of SIMLIB coordinates. If ZTRUE_CMB is specified without also using the HOSTLIB-coordinate option, the simulation aborts.

4.23.16 Connecting SIMGEN_DUMP to HOSTLIB

Consider this example in the sim-input file:

```
GENVERSION: MYTEST
SIMGEN_DUMP: CID,ZCMB,ZHELIO,GALID,GALZTRUE,GALZPHOT,GALZPHOTERR,GALZDIF
etc ...
```

The simgen-dump file can be examined with command

```
get_fitres_values.py \
  -f $SNDATA_ROOT/SIM/MYTEST/MYTEST.DUMP \
  -v ZCMB,ZHELIO,GALID,GALZTRUE,GALZPHOT,GALZPHOTERR,GALZDIF \
  --nrow 1
```

```
CID      ZCMB    ZHELIO   GALID  GALZTRUE  GALZPHOT  GALZPHOTERR   GALZDIF
2    0.732063  0.733324  328788    0.7359  0.736324       0.0155 -0.002576
```

The original HOSTLIB information is extracted using the GALID:

```
get_fitres_values.py \
 -f <HOSTLIB_FILE_NAME> \
 -g 328788 -v ZTRUE,ZPHOT,ZPHOTERR
```

```
GALID    ZTRUE   ZPHOT  ZPHOTERR
328788  0.7359  0.7389    0.0155
```

Here a few comments on how these two sets of variables are related to each other:

- ZCMB is randomly selected SN redshift (cmb frame) from rate model.
- ZHELIO is the SN helio-redshift, transformed from ZCMB.
- GALZTRUE is the original true helio-redshift of the selected host with GALID=328788.
- GALZTRUE(SIMGEN_DUMP) = ZTRUE(HOSTLIB) = 0.7359
- ZHELIO-GALZTRUE $= 0.733324 - 0.7359 = -0.002576$ is the difference between the SN redshift and randomly selected host based on HOSTLIB_DZTOL; this difference is GALZDIF.
- GALZPHOT(SIMGEN_DUMP) - ZPHOT(HOSTLIB) $= 0.736324 - 0.7389 = -0.002576 =$ GALZDIF.
- The original ZPHOT(HOSTLIB) can be computed as
  ZPHOT(HOSTLIB) = GALZPHOT(SIMGEN_DUMP) - GALZDIF(SIMGEN_DUMP)

4.23.17 Miscellaneous

HOSTLIB Variables for SIMGEN_DUMP file:
The SIMGEN_DUMP option is described in §4.38.2. Here is an example showing the HOSTLIB variables:

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

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

HOSTGAL vs. SIM_HOSTLIB variables in Data Files:
There are two sets of HOST-related parameters in the output. The first set, HOSTGAL_XXX, corresponds to observables which can appear in real data files. These obervables include OBJID, ZPHOT, ZPHOT_ERR, LOGMASS_TRUE. Additional obervables may be added later. The second set, SIM_HOSTLIB_XXX, corresponds to the user-selected parameters from the sim-input key HOSTLIB_STOREPAR. If this user-list includes an observable such as ZPHOT, then HOSTGAL_ZPHOT and SIM_HOSTLIB_ZPHOT will both appear in the data files and in the analysis output (SNTABLEs and ascii fitres file).

HOSTLIB_ZPHOTEFF_FILE vs. **SEARCHEFF_zHOST_FILE:**
The former determines the probability (vs. redshift) for finding a host galaxy photo-z, and it has no impact on setting SIM_SEARCHEFF_MASK. This option also requres ZPHOT in the HOSTLIB file. The latter determines the probability (vs. redshift) for finding a *spectroscopic* galaxy redshift, and it sets the 4-bit of SIM_SEARCHEFF_MASK. This option does not depend on the HOSTLIB.

Visual Testing with HOSTLIB_FIXRAN :
To verify the "a_rot" convention, it is useful to fix the radius and angle to known values and visually inspect the location on a real image with the HOSTLIB galaxies. A relative galaxy position and Sersic profile can be selected with

- HOSTLIB_FIXRAN_PHI is a random number ($0 < r < 1$) that fixes the azimuthal angle to $r \times 360$: $r = 0,\ 0.5,\ 1$ correspond to the major axis $(0°, 180°, 360°)$; $r = 0.25$ and 0.75 correspond to the minor axis $(90°, 270°)$.

- HOSTLIB_FIXRAN_RADIUS is a random number ($0 < r < 1$) where the reduced radius contains a fraction 'r' of the total flux.

- HOSTLIB_FIXSERSIC fixes the Sersic parameters a, b, n, and also the galaxy major axis angle (a_rot) w.r.t. Right Ascension.

4.23.18 Anomalous Flux-Scatter on Bright Galaxies

This is a near-obsolete utility, and it is recommended to use FLUXERRMAPs in §4.14. With this legacy model, the true flux uncertainty is increased as a function of surface brightness, while the nominal uncertainty is reported in the data file. The sim-input key word is

```
HOSTNOISE_FILE:  <fileName>
```

and the file syntax is

```
NOISEMODEL_NAME: SB_ERRSCALE

# SBMAG    = mag/arcsec^2 in template at SN location
#          = 27.5 - 2.5*log10(FLUXCAL_SB)
# ERRSCALE: fluxerr -> fluxerr x ERRSCALE

LIBID: 1
BAND: g    FIELD: E1+E2+S1+S2+C1+C2+X1+X2
#          SBMAG  ERRSCALE
HOSTNOISE:  20.50  4.60
HOSTNOISE:  21.50  2.70
HOSTNOISE:  22.50  1.78
HOSTNOISE:  23.50  1.40
HOSTNOISE:  24.50  1.09
HOSTNOISE:  25.50  1.02
HOSTNOISE:  26.50  0.99
HOSTNOISE:  27.50  1.00
HOSTNOISE:  28.50  0.99

and repeat for each band and group of fields.
```

In the analysis, the data-file errors can be increased with the same model using

```
&SNLCINP
   FUDGE_HOSTNOISE_FILE = '<fileName>'
```

4.23.19 Simulating Mis-Matched Host Galaxy

WARNING: the recommended mis-match method is to use HOSTLIB neighbors (§4.23.10) and DDLR matching. The WRONGHOST method here pre-dates the +HOSTNBR feature, and should only be used for rapid testing.

For surveys that do not have SN spectroscopic confirmation and instead rely on a host galaxy spectroscopic redshift, there is the issue of matching each SN candidate to the correct host galaxy. This effect can be simulated by specifying a "WRONGHOST" model with sim-input key

```
WRONGHOST_FILE:  <file>
```

and the WRONGHOST model is defined in the file with

```
#
# PROB_WRONGHOST_POLY: 0.065  0.015  -0.066  0.0544  [hard-code 3rd order]
#         or
# PROB_WRONGHOST_POLY: 0.065,0.015,-0.066          [arbitrary order]
#
# ZTRUE ZMATCH
0.921  1.019
0.875  1.455
0.517  0.499
0.174  0.1674
0.995  0.7651
0.325  0.861
etc ...
```

The key PROB_WRONGHOST_POLY specifies a 3rd order polynomial function of redshift to compute the wrong-host probability; first term is a constant, 2nd term is linear in redshift, etc. For incorrectly matched host galaxies, the remainder of the file gives a list of the true SN redshift (ZTRUE) and the redshift of the incorrectly matched galaxy (ZMATCH). WRONGHOST entries are rejected if either ZTRUE or ZMATCH is outside the HOSTLIB redshift range.

For a given SN redshift ($z_{\rm SN}$), the WRONGHOST library is searched for nearby ZTRUE values within 0.01 of $z_{\rm SN}$; a random ZTRUE is selected among these nearby values. The host redshift is computed as

$$z_{\rm host} = z_{\rm SN} + ({\rm ZMATCH} - {\rm ZTRUE}).$$

Note that the WRONGHOST model is created outside of the SNANA environment. Ideally, this model is based on matching simulated SN locations to galaxies in a catalog generated from the survey.

There are two ways to identify mis-matched hosts from the data files. First is an explicit flag, SIM_ZFLAG=4 (see § 4.37.7 for details). Second, REDSHIFT_FINAL - SIM_REDSHIFT_CMB should show a tail from wrong-host matches. In the output tables (SNANA,FITRES) from snana-analysis, examine zCMB-SIM_zCMB, and extract SIM_ZFLAG using the APPEND_TABLE_TEXT feature in split_and_fit.

4.23.20 Correlating SIMLIB and HOSTLIB with GROUPID

The default use of SIMLIB and HOSTLIB libraries assumes no correlation between instrumental cadence and host properties. While this assumption is largely correct, it fails for simulating large scale structure (LSS) because the host properties (e.g., peculiar velocity) may depend on sky location and thus the cadence and host properties are correlated. To efficiently connect the SIMLIB sky location to a host at the same (within user tolerance) sky location, use the GROUPID feature.

For each SIMLIB entry, the header can include an optional GROUPID key such as

```
LIBID: 24
HOSTLIB_GROUPID: 45,46
RA: <RA>      DEC: <DEC>
etc ...
```

With sim-input key

```
USE_SIMLIB_GROUPID: 1  # enable GROUPID feature
```

the simulation selects a host with GROUPID = 45 or 46. The HOSTLIB must include a GROUPID column to use this option of forcing an {RA,DEC}-match between the SIMLIB and HOSTLIB. This GROUPID feature is compatible with WGTMAP so that host property distributions are preserved.

Beware of defining an {RA,DEC}-match tolerance that is too strict, otherwise a GROUPID match will not be found and the simulation will abort. If the sim aborts with "Unable to select GALID", try one of the following: i) create larger HOSTLIB, ii) relax {RA,DEC} match tolerance between SIMLIB and HOSTLIB, or iii) relax redshift match tolerance (see HOSTLIB_DZTOL key).

4.24 Simulating Rate vs. Redshift: Volumetric and per Season

To simulate a constant volumetric rate at all redshifts, include one of the following sim-input options:

```
DNDZ: HUBBLE
```

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

```
DNDZ: POWERLAW  2.6E-5   1.5   # rate=2.6E-5*(1+z)^1.5 /yr/Mpc^3
```

Note that setting the second `POWERLAW` argument to zero is equivalent to the `HUBBLE` option of a constant rate. Finally, to simulate multiple power laws in different redshift ranges,

```
#                   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 `R0` is the rate (/yr/Mpc^3) at $z = 0$, `Beta` gives the z-dependence (`Beta=0` for constant rate), and the last two entries give the min/max redshift range. The output `README` file includes a dump of the SN volumetric rate in redshift bins of 0.1 (grep "`MODEL-RATE`").

To read and interpolate a table of $R(z)$,

```
DNDZ_FILE: <myrate.dat>
```

containing two columns, redshift z and volumetric rate (no keys). Comment lines with # are allowed.

Highly distorted redshift distribution for special tests can be obtained with

```
DNDZ:  FLAT                    # dN/dz = constant
   or
DNDZ: ZPOLY a0,a1,,,,aN        # dN/dz = N'th-order polymon
DNDZ: ZPOLY a0 a1 a2 a3        # LEGACY dN/dz = 3rd order polynom
   or
DNDZ: CC_S15           # CC-Rate(z) from Strolger 2015 (Fig 6, green line)
   or
DNDZ: CC_S15*.3        # 30% of S15 rate
   or
DNDZ: MD14  Rate(z=0)              # Rate(z) from Madau & Dickenson 2014
   or
DNDZ_ZEXP_REWGT:  -2.0             # dN/dz *= 1/z^2
   or
DNDZ_ZPOLY_REWGT: a0,a1,,,,aN      # dN/dz *= [N'th order polynom of z]
DNDZ_Z1POLY_REWGT: a0,a1,,,,aN     # dN/dz *= [N'th order polynom of (1+z)]
```

The "`FLAT`" command results in a flat redshift distribution, and the `ZPOLY` option specifies the redshift distribution with an $N'th$-order polynomial function of redshift. `CC_S15` is the HST-measured CC rate. `MD14` is the Star-formation rate, and there is one user-input parameter to define the rate (yr^{-1}Mpc^{-3}) at $z = 0$. 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}. The last option allows the user to multiply the "`DNDZ`" redshift distribution by an arbitrary 3rd-order polynomial function of the redshift.

As a convenience, the absolute number of SN per season within your survey ($N_{\rm season}$) 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,[15] 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
```

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. $N_{\rm season}$ 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
   or
NGEN_SEASON: 3.0
```

To scale the rates,

```
DNDZ_SCALE:       xx yy  # xx= SNIa-scale; yy=scale for NON1A/SIMSED
  or
DNDZ_SCALE_NON1A: yy     # scale NON1A and SIMSED rates (scaleIa=1.0)
  or
DNDZ_ALLSCALE:  xx       # scale rate for all models
```

This option is also useful for submit_batch_jobs.sh: e.g., to add a large Ia-biasCor sample, "GENOPT: DNDZ_SCALE 10 1.0E-8" scales the Ia sample by a factor of 10 while turning off the NON1A. Beware that the NON1A scales apply specifically to NON1ASED and NON1AGRID models. SIMSED models can be either Ia or NON1A, and thus DNDZ_ALLSCALE applies to all models.

[15] 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 (§4.30).

4.25 Simulating Rate vs. Galactic Coordinates

For the LCLIB model, the relative rate is defined as a function of Galactic b-coordinates,

```
   DNDB:  COSBPOLY  b0,b1,b2,,,,bN     # Nth order polynomial in cos(b)
   DNDB:  COSBPOLY  b0 b1 b2 b3 b4 b5  # legacy input for 5th order polynom
      or
   DNDB:  BPOLY     b0,b1,b2,,,,bN     # Nth order polynomial in b
   DNDB:  BPOLY     b0 b1 b2 b3 b4 b5  # legacy input for 5th order polynom

# ------- examples -------
   DNDB:  COSBPOLY   1       # isotropic
   DNDB:  COSBPOLY   1,0.3   # 1 + 0.3*cos(b)
```

which defines the b-dependence with an $N'th$-order polynomial function of $\cos(b)$ or b. The absolute number of generated events is given by NGENTOT_LC. *Beware* of the following:

- DNDB is a re-weight function applied to the SIMLIB distribution.
- For isotropic SIMLIB and highly anisotropic DNDB reweight, set SIMLIB_NREPEAT $\gg 1$ for efficient generation.

With default "SIMLIB_NREPEAT: 1," the simulation randomly selects/rejects each SIMLIB entry based on the input DNDB function. This procedure is efficient for nearly isotropic DNDB, but for more realistic DNDB with a very large anisotropy, the simulation spends most of the time reading and skipping SIMLIB entries, and is thus very inefficient.

To reduce reading of SIMLIB entries, and thus reduce processing time, use the SIMLIB_NREPEAT feature (§4.7.1). While SIMLIB_NREPEAT is fixed for isotropic rate models that depend on redshift (§4.24), here it is multiplied by the relative rate for each SIMLIB entry i:

```
SIMLIB_NREPEAT(i) = max(SIMLIB_NREPEAT) * DNDB(b)/max[DNDB(b)]
```

The b-weight is implemented by how many times each SIMLIB entry is read. Since SIMLIB_NREPEAT(i) is a float, the floating remainder is compared with random number to determine whether to use NREPEAT=int(SIMLIB_NREPEAT) or int(SIMLIB_NREPEAT)+1. If NREPEAT=0, the SIMLIB entry is skipped.

As an example consider a $\cos^5(b)$ profile,

```
DNDB:  COSBPOLY  0,0,0,0,0,1
```

With default "SIMGEN_NREPEAT: 1" using each SIMLIB entry once, only $\int_0^1 x^5 = 1/6$ of the (isotropic) SIMLIB entries are processed. However, with "SIMGEN_NREPEAT: 100", SIMLIB entries with $|\cos(b)| > 0.4$ (60%) have SIMLIB_NREPEAT(i)> 1 and thus are processed. A caveat is to generate enough events to sample most of the SIMLIB; e.g., if NGENTOT]_LC = 100 then only a few SIMLIB entries are sampled in this example.

The DNDB-rate feature specifies a relative re-weight vs. b, but the simulation has no mechanism to normalize separate simulations of different sky regions. Here we describe how to determine the relative normalization using the SIMLIB_DUMP feature (§4.38.1). Consider three sky regions, S1, S2, S3, with solid angles $\Omega_1,\ \Omega_2,\ \Omega_3$, and each with a separate SIMLIB-cadence file. Running the simulation with an LCLIB model and the SIMLIB_DUMP option results in a calculation and screen-dump of the average weight over the SIMLIB: $\bar{w}_1 = [\sum_i \mathtt{DNDB}_i]/\mathtt{NLIBID}_1$, and similarly for $\bar{w}_2$ and $\bar{w}_3$. The sum over $i = 1, \mathtt{NLIBID}$ runs over each entry in the SIMLIB file. The NGENTOT_LC values to generate are

$$\begin{aligned} \mathtt{NGENTOT\_LC} &= N\Omega_1\bar{w}_1 \quad \text{(S1)} \\ \mathtt{NGENTOT\_LC} &= N\Omega_2\bar{w}_2 \quad \text{(S2)} \\ \mathtt{NGENTOT\_LC} &= N\Omega_3\bar{w}_3 \quad \text{(S3)} \end{aligned}$$

where N is an arbitrary number setting the global normalization.

4.26 Simulating a SPECTROGRAPH

As described in §3.2, a SPECTROGRAPH instrument can be defined in a text file and then ingested into a kcor file. While the photometric (broadband) fluxes and SNR are computed from observational information (ZP,PSF,SKY), spectral SNR-vs-λ are stored in the kcor file and read by the simulation. Thus, users must estimate spectral SNR properties with an external calculator. There are two methods for simulating spectra. First, the SPECTROGRAPH can be included in the SIMLIB file as described in §4.7.2. This method results in spectra at fixed MJD values regardless of the explosion date. Since spectroscopic programs tend to target SNe based on the time of peak brightness (t_0), there is a second method to simulate spectra within arbitrary time windows with respect to t_0 (§4.26.2).

Here are few warnings:

- works for the following models: SALT2, NON1ASED, FIXMAG, BYOSED (WARNING: does not work for SIMSED)
- if true flux is negative, it is suppressed in the output. So beware of spectroscopic wavelength holes, particularly at early epochs and the UV. To replace negative fluxes with zero, set GENMODEL_MSKOPT=16.
- there are no SNANA programs which read the simulated spectra.
- Units: $dF/d\lambda$, erg/s/cm^2/Å.

For TEXT formatted data files (FORMAT_MASK: 2), the spectra are appended to the ascii file for each SN; search for "SPECTRUM_" keys. For FITS format (FORMAT_MASK: 32), a separate *SPEC.FITS file is created with two tables as follows:

```
   TableName     |Ncol|     column names
-----------------------------------------------------------------
SPECTRO_HEADER   | 9  | SNID MJD Texpose SNR_COMPUTE LAMMIN_SNR LAMMAX_SNR
                 |    | NBIN_LAM  PTRSPEC_MIN PTRSPEC_MAX
SPECTRO_FLUX     |5-6 | LAMMIN LAMMAX FLAM FLAMERR  SIM_FLAM  SIM_WARP
```

The SPECTRO_HEADER table provides a one-row summary for each spectrum: 1) SNID is the object ID, 2) MJD is the spectrum date, 3) Texpose is the exposure time (sec), 4) SNR_COMPUTE is computed SNR, or -9 if SNR-option not used, 5,6) LAMMIN_SNR,LAMMAX_SNR is the observed λ-range whose flux-sum corresponds to SNR_COMPUTE, 7) NBIN_LAM is the number of λ bins (see comment above about suppressing negative fluxes), 8,9) PTRSPEC_MIN,PTRSPEC_MAX are pointers to extract the spectrum from the SPECTRO_FLUX table.

The SPECTRO_FLUX table contains information for every spectral bin for every SN: min and max wavelength, FLAM ($dF/d\lambda$), its uncertainty FLAMERR, and the generated (true) flux, SIM_FLAM. If a WARP_SPECTRUM key is used, there is an extra SIM_WARP column; it is multiplied by 1000 and stored as short int. After the last wavelength bin of a spectrum in the SPECTRO_FLUX table, the next row is an end-of-spectrum marker containing "777 -777 -777 0"; this marker should be checked to ensure correct parsing.

The first table is small, and should read quickly and stored in memory for quick access. The 2nd (SPECTRO_FLUX) table can be very large (few GB), so beware of memory storage. To avoid memory issues, each spectrum can be read quickly from SPECTRO_FLUX table using pointers in the SPECTRO_HEADER table.

4.26.1 Spectral MJD-Windows or Sequences

Spectra can be generated within MJD windows or with a cadence:

```
TAKE_SPECTRUM:  MJD(59400:59407)    TEXPOSE_ZPOLY(1200)   # random within week
TAKE_SPECTRUM:  MJD(59400:59600:20) TEXPOSE_ZPOLY(1200)   # every 20 days

# define different exposure time in each field
TAKE_SPECTRUM(SHALLOW):  MJD(59400:59407)  TEXPOSE_ZPOLY(800)
TAKE_SPECTRUM(DEEP):     MJD(59400:59407)  TEXPOSE_ZPOLY(1600)
```

The `MJD` argument defines either i) a pre-defined calendar window from which to select a random MJD, or ii) a pre-defined calendar sequence; e.g,, `MJD(59400:59600:20)` defines 11 spectra, one every 20 days. The `TEXPOSE_ZPOLY` key defines a 1200 sec exposure time; see §4.26.2 for more details.

The latter two keus define field-dependent exposure times for `SHALLOW` and `DEEP` fields. Beware that an invalid field name (e.g., typing mistake) results in no spectra without any warnings.

4.26.2 Spectral Time-Windows Relative to Peak Brightness

In the sim-input file, spectroscopic exposure times can be assigned with `TAKE_SPECTRUM` keys as follows:

```
TAKE_SPECTRUM:  TREST(-12:-10) TEXPOSE_ZPOLY(2000,500)
TAKE_SPECTRUM:  TOBS(-7:7)     TEXPOSE_ZPOLY(600,200,-3)
TAKE_SPECTRUM:  TREST(0:2)     TEXPOSE_ZPOLY(1000)
TAKE_SPECTRUM:  TREST(10:12)   TEXPOSE_ZPOLY(1500:2500,500)
TAKE_SPECTRUM:  HOST           TEXPOSE_ZPOLY(1200)

# field-dependent exposure time
TAKE_SPECTRUM(SHAL):  TREST(0:2)     TEXPOSE_ZPOLY(1000)
TAKE_SPECTRUM(DEEP):  TREST(0:2)     TEXPOSE_ZPOLY(2000)
```

Each `TREST` argument defines a 2-day rest-frame window from which to randomly select a spectroscopic MJD. The `TOBS` argument defines a 2-week observer-frame window from which to randomly select a spectroscopic MJD. The `TEXPOSE_ZPOLY` key defines the exposure time (seconds) as a polynomial function of redshift. The `TREST` exposure times are $2000+500z$ sec (10-12 days before peak), $600+200z-3z^2$ sec (7 days before peak), 1000 sec (near-peak), $[1500:2500]+500z$ sec (10-12 days after peak).

The values in parentheses can be float or integer: e.g., `TREST(-2.5:2.5)`. Beware that no blank spaces are allowed inside the (). The colon separates a range, while commas separate a list; if you use the wrong punctuation, the simulation will abort. A range for a polynomial coefficient (4th example above) results in a randomly selected coefficient. Arbitrary polynomial orders are specified with comma-separated values; 1st value is constant term, 2nd value is linear term, etc ... Finally, the `HOST` argument results in a host spectrum based on `HOSTLIB_FILE` and `HOSTLIB_SPECBASIS_FILE`. HOST spectra have `MJD = -9` in the data files.

The template exposure time is defined in the `SIMLIB` file with the key `TEMPLATE_TEXPOSE_SPECTROGRAPH`.

Rather than pre-defining exposure times, the exposure time can be computed from a requested SNR value. The TREST and TOBS arguments are the same as in the previous example. SNR_ZPOLY specifies the requested SNR as a polynomial function of redshift. In the first pre-peak example, SNR $= 20 - 5z$, allowing the SNR to degrade with increasing redshift in order to reduce exposure time. The third argument, SNR_LAMREST, specifies the rest-frame wavelength range for which SNR is defined: for each event, the observer wavelength range is $5000(1+z)$ Å to $6000(1+z)$ Å. The third block of arguments shows that SNR can be defined for observer-frame λ-ranges using SNR_LAMOBS.

When SNR_ZPOLY keys are defined, the SNR and $T_{\rm expose}$ information is automatically added to the SIMGEN_DUMP file (§4.38.2). This allows checking $T_{\rm expose}$ vs. redshift, or vs any other quantity allowed in the one-row-per-SN summary.

```
# 1) rest-frame epoch, rest-frame SNR def
TAKE_SPECTRUM:  TREST(-12:-10) SNR_ZPOLY(20,-5)  SNR_LAMREST(5000:6000)
TAKE_SPECTRUM:  TREST(0:2)     SNR_ZPOLY(20)     SNR_LAMREST(5000:6000)
TAKE_SPECTRUM:  TREST(10:12)   SNR_ZPOLY(20,-2)  SNR_LAMREST(5000:6000)

# 2) obs-frame window, rest-frame SNR def
TAKE_SPECTRUM:  TOBS(-7:7)      SNR_ZPOLY(20)    SNR_LAMREST(5000:6000)

# 3) rest-frame epoch, obs-frame SNR def
TAKE_SPECTRUM:  TREST(-3:3)    SNR_ZPOLY(20)   SNR_LAMOBS(8000:10000)
TAKE_SPECTRUM:  TREST(-3:3)    SNR_ZPOLY(20)   SNR_LAMOBS(13000:15000)

#4) host spectrum
TAKE_SPECTRUM:  HOST           SNR_ZPOLY(20)   SNR_LAMOBS(3000:5000)
TAKE_SPECTRUM_HOSTFRAC:   0.1  # add 10\% of FLAM(host) to each SN spectrum
TAKE_SPECTRUM_HOSTSNFRAC: 0.1  # scale FLAM(HOST) so Sum(HOST)/Sum(SNPEAK) = 0.1

TAKE_SPECTRUM:  TEMPLATE_TEXPOSE_SCALE(1.2)
```

There are two methods to introduce host contamination in SN spectra. The first method, TAKE_SPECTRUM_HOSTFRAC, adds a specific fraction of host spectrum to the SN spectrum; this option depends on correct normalization of host spectrum. The 2nd method (TAKE_SPECTRUM_HOSTSNFRAC) scales the host normalization relative to the SN at peak brightness; the correct host normalization is not necessary because $dF_{\rm host}/d\lambda$ is scaled by $S_{\rm host}$ such that

$$\texttt{HOSTSNFRAC} = \frac{S_{\rm host} \int d\lambda [dF_{\rm host}/d\lambda]}{\int d\lambda [dF_{\rm peak}/d\lambda]} \ , \tag{27}$$

where $dF_{\rm peak}/d\lambda$ is the SN spectrum at peak brightness, and the integration is over the spectrograph wavelength range in which both $F_{\rm host}$ and $F_{\rm peak}$ are defined.

There are two methods to define the template exposure time. First is to defined a fixed exposure time in the SIMLIB file with the key TEMPLATE_TEXPOSE_SPECTROGRAPH. The second option is defined in the example above with

```
TAKE_SPECTRUM:  TEMPLATE_TEXPOSE_SCALE(1.2)
```

which sets the template exposure time to be 20% longer than that used at the epoch nearest peak brightness. The template exposure time and noise are computed for each SN along with the search exposure times. While the search exposure time varies with each epoch to acquire the specified SNR, the template exposure time is the same for all epochs. This key overrides the TEMPLATE_TEXPOSE_SPECTROGRAPH key in the SIMLIB file.

4.26.3 Disable TAKE_SPECTRUM keys

Disable TAKE_SPECTRUM keys without commenting them out or removing them from the sim-input file;

```
snlc_sim.exe <inputFile>  TAKE_SPECTRUM NONE
```

WARNING: can use either the TAKE_SPECTRUM keys in sim-input file, or the SPECTROGRAPH keys in the SIMLIB file; using both results in an abort.

4.26.4 Calibration Warp vs. Wavelength

Smooth spectral mis-calibration can be defined as a polynominal function of wavelength using the sim-input WARP_SPECTRUM key as follows:

```
WARP_SPECTRUM:  LAMPOLY(1,1.0E-5)
TAKE_SPECTRUM:  TREST(-12:-10)  TEXPOSE_ZPOLY(2000,500,-20)
TAKE_SPECTRUM:  TREST(0:5)  SNR_ZPOLY(20:30,-5)  SNR_LAMREST(5000:6000)

WARP_SPECTRUM:  LAMPOLY(1,-1.0E-5:1.0E-5)
TAKE_SPECTRUM:  TREST(10:12)  TEXPOSE_ZPOLY(2000,500)
```

The first LAMPOLY argument above is a multiplicative calibration warp of $dF/d\lambda \rightarrow dF/d\lambda \times (1 + 10^{-5}\lambda)$, and it is applied to the first two TAKE_SPECTRUM keys. The 2nd LAMPOLY argument specifies a warp range with a slope (dWARP/dλ) randomly selected between -10^{-5} and $+10^{-5}$; this random warp is applied to the 3rd TAKE_SPECTRUM key. Analogous to the TEXPOSE_ZPOLY and SNR_ZPOLY arguments, the LAMPOLY argument can include an arbitrary polynomial order defined with more comma-separated terms. The warping is applied only to the measured $dF/d\lambda$ (FLAM column in output); the true $dF/d\lambda$ is not warped (SIM_GENFLAM).

4.26.5 Field-Dependent PreScale for SN Spectra

Since SN spectra are typically limited to a subset of events, field-dependent pre-scales for SN spectra can be specified in two distinct ways. First is a global prescale per field, such as:

```
# SN spectra for half of SHALLOW events
   TAKE_SPECTRUM_PRESCALE: SHALLOW/2

# SN spectra for 1/4.2 of shallow and 1/2.05 of deep
   TAKE_SPECTRUM_PRESCALE: SHALLOW/4.2+DEEP/2.05
```

The prescale is applied per event, not per spectrum. Thus if 6 SN spectra are defined per event, the prescale results in each event having either 6 SN spectra, or none. While there is no host spectrum prescale, there is a zHOST efficiency file (§4.18.3).

The second prescale method allows controling the prescale as a function of phase and exposure time (or SNR), even for the same field:

```
   TAKE_SPECTRUM(DEEP/4):   TREST(-3:3)     TEXPOSE_ZPOLY(1000)
   TAKE_SPECTRUM(DEEP/8):   TREST(-3:3)     TEXPOSE_ZPOLY(2000)
   TAKE_SPECTRUM(DEEP/10):  TREST(-10:-5)   TEXPOSE_ZPOLY(2500)
   TAKE_SPECTRUM(DEEP/15):  TREST(-10:-5)   TEXPOSE_ZPOLY(3500)

   TAKE_SPECTRUM(SHALLOW/2):  TREST(-3:3)  TEXPOSE_ZPOLY(500)
   TAKE_SPECTRUM(SHALLOW/5):  TREST(-3:3)  TEXPOSE_ZPOLY(800)
```

The TREST and TEXPOSE_ZPOLY keys can be replaced with other valid keys at that location, and the integer prescales shown here can be replaced with floats (e.g., DEEP/4.23). This option is useful to study multiple spec-follow up strategies with a single generation of the simulation, provided that the FIELD-dependent exposure times are unique for each TAKE_SPECTRUM key.

4.26.6 SPECTROGRAPH Options

The SPECTROGRAPH_OPTMASK key in the sim-input file can be used as follows:

```
  SPECTROGRAPH_OPTMASK:  1   # turn off lambda smearing
  SPECTROGRAPH_OPTMASK:  2   # double LAMSIGMA smearing
  SPECTROGRAPH_OPTMASK:  4   # flux only in center lambda bin (delta function)
  SPECTROGRAPH_OPTMASK:  8   # SNR -> SNR x 100
  SPECTROGRAPH_OPTMASK: 16   # TREF -> TREF x 100 (template expose time)
  SPECTROGRAPH_OPTMASK: 32   # only template noise
  SPECTROGRAPH_OPTMASK: 64   # for SNR_REQUEST, extrap TEXPOSE if needed
  SPECTROGRAPH_OPTMASK: 2048 # skip spectra
  SPECTROGRAPH_OPTMASK: 32768# turn off all noise in spectra
  SPECTROGRAPH_OPTMASK:  6   # flux only in center bin & LAMSIGMA*=2

  SPECTROGRAPH_SCALE_TEXPOSE: 2.4  # scale exposure times by 2.4
  SPECTROGRAPH_SCALE_SNR:     0.9,0.02,-.003  # SNR *= [0.9 + 0.02*lam - 0.003*lam^2]
```

Multiple options can be combined by adding values as illustrated by the last option above with SPECTROGRAPH_OPTMASK=6. The default is SPECTROGRAPH_OPTMASK=0, and the options above are intended for testing and debugging.

OPTMASK+=64 is a little tricky, so here is added explanation. By default, the exposure time is bounded by the min and max TEXPOSE_LIST values in the spectrograph table (Fig. 3). For example, suppose SNR_MAX=20 at the max exposure time of TEXPOSE=2000, and the user requests SNR_REQUEST=40 (via TAKE_SPECTRUM key); the default behavior is to return TEXPOSE=2000 and SNR=20 to avoid exposing for longer than what has been defined. With OPTMASK+=64, however, the exposure time is extended to give the requested SNR:

$$\texttt{TEXPOSE} = \texttt{TEXPOSE\_MAX} + [\texttt{SNR\_REQUEST}/\texttt{SNR\_MAX}]^2 \quad (28)$$

and a similar equation is used for requested SNR below the minimum-defined SNR_MIN.

The last option (SCALE_TEXPOSE) is a continuously tunable knob to scale the exposure times (for search and template).

4.26.7 Simulating a Single High-S/N Spectrum

A single high-S/N spectrum (rest-frame) can be simulated at arbitrary redshift using the NON1ASED model (§9.7), and defining sim-input key PATH_NON1ASED to use a private NON1ASED directory. Instead of defining an SED time series covering a few-month range of epochs, the NON1ASED file can contain a single spectrum at one epoch. The DAY column can have any value since internally the simulation will shift the SED times such that DAY=0 at max flux. Hence by definition, a single epoch will have DAY=0. Recall that uniform wavelength binning is required.

To generate one spectrum, the sim-input key GENRANGE_PEAKMJD should be defined as a δ-function landing exactly on any SPECTROGRAPH MJD in the SIMLIB file. See §4.7.2 for how the SPECTROGRAPH is defined in the SIMLIB. It is also recommended to set "GENRANGE_TREST: -1 1" to remove photometry output from epochs with an undefined model. The simulated spectrum and photometry is artificially defined to appear at "peak," but they really correspond to the epoch from which the input spectrum was extracted. As a sanity check, the simulated mag should be compared with that from the input light curve.

Several spectra can be included in the NON1ASED model. For example, if there are 8 SEDs, then setting "NGENTOT_LC: 8" will generate each spectrum once. If several spectra come from the same SN, each spectrum should be defined as a separate SED file.

4.26.8 Output Observer-Frame Model SED

To output the true observer-frame model SED at each observation to the data files (TEXT for FITS format), set the following sim-input key

```
LAMBIN_SED_TRUE:    10                  # wavelength bin size (A) for true SED output
LAMRANGE_SED_TRUE:  <lammin> <lammax>   # optional; defaults explained below
VERIFY_SED_TRUE:    1                   # diagnostic dump
```

This option is unique because a spectrograph need NOT be defined in the kcor/calib file. The simulation internally defines a spectrograph with LAMBIN_SED_TRUE Å bin size and SNR$\sim 10^5$. The simulation creates an ideal spectrum, corresponding to the true SED, for each observation that occurs at least 1 hr later than the previous observation. BEWARE: to avoid excessively large output data files, define reasonable GENRANGE_TREST, NGENTOT_LC, and LAMBIN_SED_TRUE.

The default true-SED wavelength range is as follows. Min-wave is minimum of blue edge for bluest filter or model-SED. Max-wave is maximum of red edge for reddest filter or model-SED. The default range can be overwritten by sim-input key LAMRANGE_SED_TRUE, and this range can only extend the wavelength range beyond that defined by the bluest & reddest filter edge; if LAMRANGE_SED_TRUE defines a more strict wavelength range, the sim will abort.

4.27 Simulating Rise-Time Variations

The rise-time for any model can be adjusted using the following sim-input parameters:

```
GENPEAK_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 $2.3 \times T_{\rm rest}/18$ days with a Gaussian sigma of 0.6 days, and shifts past ± 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.

4.28 Simulating Atmosphere Effects

The simulation of atmoshperic effects includes differential chromatic refraction (DCR) and its effect on measured coordinates and magnitudes. The impact on spectra is not included. The DCR effects are enabled with sim-inputs

```
   ATMOSPHERE_OPTMASK: 1  # simulate DCR effect on coordinates
   ATMOSPHERE_OPTMASK: 2  # simulate DCR effect from PSF shape
   ATMOSPHERE_OPTMASK: 3  # simulate both DCR effects
   ATMOSPHERE_OPTMASK: 513 # 1+512: 512-> write [VERSION].DCR summary file

   ATMOSPHERE_SEDSTAR_FILE:   [sed_fileName]
   ATMOSPHERE_DCR_COORDRES_POLY: a0,a1,...  # poly for coordRes(asec) vs. PSF_FWHM/SNR
   ATMOSPHERE_DCR_MAGSHIFT_POLY: a0,a1,a2,. # poly for magShift vs. fracPSF shift
racPSF
```

The geo-location (degrees) and mountain elevation (meters w.r.t. sea level) of the instrument must be specified in the `$SNDATA_ROOT/SURVEY.DEF` file; e.g.,

```
   SURVEY:  LSST   12   geo:-30.244639,-70.749417,2647  # lat,long,elevation
```

The sim aborts if `ATMOSPHERE_OPTMASK`> 0 and the geolocation is not provided.

While the default simulation defines the measured RA,DEC to be their exact values without measurement noise, `ATMOSPHERE_OPTMASK`> 0 changes the sim to determine the following for each observation: i) measured RA,DEC using coordinate-smear based on a polynomial function of `PSF_FWHM/SNR` (`ATMOSPHERE_DCR_COORDRES_POLY`), and ii) calculated mag-shift based on a polynomial function of fractional PSF-shift (`ATMOSPHERE_DCR_MAGSHIFT_POLY`). These polynomial functions are computed externally from `SNANA`, such as from studies using `GALSIM`.

The average reference star SED (`ATMOSPHERE_SEDSTAR_FILE`) is used to compute a reference SED-weighted wavelength in each filter band ($\bar{\lambda}_{\rm refStar}$), and DCR coordinate shifts are defined as the shift-difference between the SN SED-weighted wavelength and $\bar{\lambda}_{\rm refStar}$. The RA,DEC shift for each observation is defined w.r.t. the filter-averaged RA,DEC over all observations with `SNR>3`.

The simulation of atmosphere effects automatically appends the following columns in the photometry table per observations:

```
 dRA             # RA  - RA_AVG_FILTER (arcsec)
 dDEC            # DEC - DEC_AVG_FILTER (arcsec)
 AIRMASS         # uses telescope geo-location in SURVEY.DEF file
 SIM_DCR_dRA     # calculated RA  shift from DCR (arcsec)
 SIM_DCR_dDEC    # calculated DEC shift from DCR (arcsec)
 SIM_DCR_dMAGOBS # calculated mag shift from fractional PSF-shift
```

The 512-bit of `ATMOSPHERE_OPTMASK` results in writing an extra auxillary table file, `[VERSION].DCR`, that contains a summary for each observation of DCR-related information that is easier to access compared to data files.

4.29 Altering Input SEDs

The options in this section work for NON1ASED and SIMSED models.

4.29.1 Simulating PEAKMJD or Time of Explosion

For all models, the default light-curve time window is defined by its time of peak brightnes, or PEAKMJD. In particular, the sim-input key

```
GENRANGE_PEAKMJD: <MJDmin>  <MJDmax>
```

specifies an MJD window to randomly generate PEAKMJD.

For triggered events, such as from gravity waves (LIGO) or neutrinos (ICECUBE), it is more practical to simulate sources based on time of explosion instead of PEAKMJD. For NON1ASED and SIMSED models, a time of explosion can be specified with

```
MJD_EXPLODE:  <MJD>
OPTMASK_TOSHIFT_EXPLODE: 0  # no shift: model T=0 at explosion (default)
   or
OPTMASK_TOSHIFT_EXPLODE: 1  # T_explode at .01*FLUXMAX
   or
OPTMASK_TOSHIFT_EXPLODE: 2  # T_explode at .001*FLUXMAX
```

Note that the time of explosion is not always observed, and thus can be ambiguous for for data-derived template models. For models with well-defined time of explosion at DAY=0, the default OPTMASK=0 adds no shift. OPTMASK=1 computes explosion time to be the epoch when the rest-frame SED flux, in any wavelength bin, falls below 1% of the maximum flux. OPTMASK=2 requires less than 0.1% of max flux. Other options based on rise-time shape fits may be added later.

4.29.2 Extrapolating UV Flux

At sufficiently high redshift, broadband fluxes in UV filters are not defined and thus not included in the ouptut data files. For example, consider a UV filter spannng 3000-4000Å, and input SEDs starting at 2000 Å. For redshifts $z > 3000/2000 - 1 = 0.5$, the UV flux depends on the undefined SED range below 2000 Å, and therefore this band is suppressed in the output. Missing UV bands are usually harmless, but this artifact can potentially fool classifiers training on data, or provide un-intended clues in a data challenge.

Since far-UV fluxes are typically well below the sky noise, simply reporting a random sky noise would be good enough. The simulation includes a somewhat better option, which is to extrapolate the UV flux down to arbitrary wavelength. The sim-input key is

```
UVLAM_EXTRAPFLUX: 500  # extrapolate SED flux down to 500 A.
```

Defining F_λ to be the flux at a particular wavelength, F_{2000} is the edge-flux at 2000Å. The UVLAM option above will extrapolate the flux to be linearly decreasing so that $F_{500} = 0$. This option extends the valid UV-filter redshift range from 0.5 to $3000/500 - 1 = 5$. Finally, users are cautioned to check that the UV flux is indeed negligible at high redshifts.

4.30 NGEN keys

There are three "NGEN" keys to control the number of generated events:

```
NGEN_LC:    1000  ! default is zero
    or
NGENTOT_LC: 1000  ! default is zero

NGEN_SCALE: 3           ! scale all models: default is 1
NGEN_SCALE_SCALE: 3     ! scale NON1A and SIMSED models: default is 1
```

The first key, NGEN_LC, specifies the number of SN generated and written out after trigger and selection cuts. Thus if the simulated efficiency is 10% and NGEN_LC = 1000 as shown in the example above, the simulation will generate 10,000 SNe in order to get 1000 SN written out. In short, SNe are generated until 1000 are written out, regardless of the efficiency. The sim-input key EFFERR_STOPGEN prevents an infinite loop if the efficiency is near zero (§4.19).

The second key, NGENTOT_LC, specifies the total number of SNe to generate regardless of the efficiency. Thus in the example above with NGENTOT_LC = 1000 and a 10% efficiency, only about 100 SNe are written out. If the efficiency is very low, it is possible that zero events are written out. This option is useful to generate statistics corresponding to a particular SN rate and survey length (§4.24). Only of of the NGEN_LC or NGENTOT_LC keys can be set; if both are set the simulation will abort.

Finally, NGEN_SCALE can be used to scale whichever NGEN key is set. This feature is useful, for example, to change the statistics on several independent simulated samples while preserving the relative ratios between samples.

4.31 "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.

4.32 Generating Redshift ($z_{\rm hel}$, $z_{\rm cmb}$) and Distance

An overview of redshift-related inputs are as follows:

```
GENRANGE_REDSHIFT:  0.01 1.0  # ZCMB-gen range
GENSIGMA_REDSHIFT:  0.001     # Gaussian z-error      (default=0)
GENBIAS_REDSHIFT:   1.0E-4    # global redshift bias  (default=0)
VEL_CMBAPEX:        370       # CMB dipole, (default = 370 km/sec)
```

The CMB redshift is generated first, and then the RA & DEC from the SIMLIB file is used to compute $z_{\rm hel}$ with $\ell = 264.14$ deg, $b = 48.26$ deg, and default VEL_CMBAPEX=370.

To generate $z_{\rm hel} = z_{\rm cmb}$ (sometimes useful for debugging), set GENSIGMA_VPEC (§4.32.1) and VEL_CMBAPEX to zero in the sim-input file, or use the command-line override "VEL_CMBAPEX 0 GENSIGMA_VPEC 0".

The simulated/true $z_{\rm cmb}$ range is defined by GENRANGE_REDSHIFT, and the probability vs. redshift is defined by the rate model (§4.24). The true $z_{\rm hel}$ is transformed from $z_{\rm cmb}$ using the sky coordinates and a peculiar velocity ($v_{\rm pec}$, §4.32.1). The measured $z_{\rm hel}$ (REDSHIFT_HELIO in data files) includes measurement noise drawn from a Gaussian with σ_z =GENSIGMA_REDSHIFT. The measured $z_{\rm cmb}$ (REDSHIFT_CMB in data files) is computed from the measured $z_{\rm hel}$ using the sky coordinates. The true quanties (without measurement noise) are stored in the data files as SIM_REDSHIFT_CMB and SIM_REDSHIFT_HELIO. Note that SIM_REDSHIFT_HELIO includes the peculiar velocity, and therefore the SIM_REDSHIFT_XXX quantities will not transform under the usual cmb↔heliocentric transformations unless $v_{\rm pec} = 0$.

It is important to pay attention to redshift ranges in different parts of the analysis. While the simulation generates a $z_{\rm cmb}$ range, the analysis programs (snana.exe and snlc_fit.exe) select a redshift range based on $z_{\rm hel}$ because $z_{\rm hel}$ is used for lightcurve fitting (see &SNLCINP namelist parameter CUTWIN_REDSHIFT). Finally, the cosmology fitting program is likely to apply cuts on the observed $z_{\rm cmb}$. To be on the safe side, one should generate a slightly larger $z_{\rm cmb}$-redshift range compared to the anticipated analysis cuts. The extended generation range should allow for $v_{\rm pec}$ variations[16] and measurement noise.

Finally, rather than simulating a z-bias with GENBIAS_REDSHIFT, a z-bias can be added in the analysis/fitting program (data or sim) with &SNLCINP parameters shown in § 5.24.

The simulated luminosity distance is computed as

$$D_L = (1 + z_{\rm hel}) r(z_{\rm cmb}), \quad r(z_{\rm cmb}) \equiv (c/H_0) \int_0^{z_{\rm cmb}} dz/H(z) \ . \tag{29}$$

To override either min or max redshift,

```
snlc_sim.exe <inputFile> GENRANGE_REDSHIFT .05 -1  # (a) increase zmin if < 0.05
      or
snlc_sim.exe <inputFile> GENRANGE_REDSHIFT -1  1.3 # (b) decrease zmax if > 1.3
```

and setting the -1 value in a sim-input file results in abort. Note that the half-range update is applied *only* if it is more strict than the original value. For example, if zmin=0.06 in the sim-input file, the first example above makes no change; but if zmin=0.04 in the sim-input file, then zmin is increased to 0.05. A usage example is a new HOSTLIB with min galaxy redshift above the minimun defined by GENRANGE_REDSHIFT. To avoid abort, add global command (a) in either the command-line override, submit_batch_jobs, or pippin.

[16] Beware that the maximum $v_{\rm pec}$-redshift variation is $(1+z)370/c$ and not just $370/c$.

4.32.1 Generating Peculiar Velocity

There are two methods to simulate peculiar velocity. The first method is to define a Gaussian sigma for the true profile and for the measured precision in the sim-input file,

```
GENSIGMA_VPEC:   300    # true sigma(v_pec), km/sec  (default=0)
VPEC_ERR:        150    # error on measured/computed VPEC (default=0)
```

A Gaussian-random peculiar velocity (v_{pec}) is defined by GENSIGMA_VPEC,[17] and corresponds to the true scatter. VPEC_ERR is the error on a v_{pec} estimate (see §5.29) used for analysis. The true peculiar-velocity value is

```
SIM_VPEC = GENSIGMA_VPEC * GaussRan
```

and the "measured VPEC" reported in the data file is

```
VPEC =  SIM_VPEC + VPEC_ERR*GaussRan  # VPEC_ERR <  GENSIGMA_VPEC
VPEC =  0                             # VPEC_ERR >= GENSIGMA_VPEC
```

For high-z samples that do not privde a v_{pec} estimate, set VPEC_ERR = GENSIGMA_VPEC as a flag to skip the VPEC estimation,

The estimated VPEC and VPEC_ERR are written to the data files for analysis, but they are not used by the sim to modify the reported redshift; this is analogous to providing Galactic extinction MWEBV without correcting the reported fluxes. The lightcurve fitting program (snlc_fit.exe) outputs a Hubble diagram redshift zHD that includes a VPEC correction, and the uncertainty zHDERR includes VPEC_ERR.

The first method has the advantage of simplicitly, but does not account for large-scale correlations and bulk flows. To account for such effects, the second method reads a VPEC column from the HOSTLIB if the 512-bit is set in HOSTLIB_MSKOPT (§4.23.1). The associated uncertainty can be defined in two ways: 1) a separate VPEC_ERR column defining a distinct uncertainty for each host, or 2) fixing the uncertainty with HOSTLIB header key "VPEC_ERR: <vpecErr>." The output README includes summary information as follows:

```
!*!*!*! TO DO: fix this to account for README refactor *!*!*!*!
```

Note that mixing the two methods is not allowed; e.g, the sim will abort if the 512-bit is set for HOSTLIB_MSKTOPT while GENSIGMA_VPEC or VPEC_ERR is defined in the sim-input file.

[17] SNANA users would be grateful if somebody provides code to compute v_{pec} based on RA, DEC, and z_{cmb}.

4.33 Population Parameters: Stretch, Color, Dust

For SNIa models, the stretch and color parameters are each drawn from a probability distribution function (PDF), also called a parent population. The PDF is a best estimate of the complete population without selection biases. Examples: for SALT2, a PDF is needed for color (c), stretch (x_1), and possibly host-galaxy dust parameters A_V and R_V; `MLCS2k2` needs A_V, R_V, and shape-parameter (Δ); SNCC models need A_V and R_V.

There are two methods for describing each PDF: i) analytical function using asymmetric Gaussian (`AG`) or exponential, and ii) multi-dimensional map including aribtrary dependence on `HOSTLIB` parameters. `AG` sim-inputs are shown in §9.3 for the SALT2 model, and exponential models for extinction (A_V) are shown in §9.1.

An illustration of a map with SALT2x1 dependence on host-galaxy mass is shown in Fig. 14. Multiple maps can be defined in the same file, where a new `VARNAMES` key is the start of each map. Any `HOSTLIB` parameter(s) can be included in the map. The first variable (e..g, `SALT2x1`) is name of the variable to be generated by the simulation; the last variable must be `PROB` to indicate "probability." The simulation internally normalizes the PDF so that the max `PROB` value is 1.0; however, it is recommended to create properly normalized maps to simplify debugging if visual inspetion of the maps is needed. A combination of `AG` and PDF maps can be used; e.g., `AG` for RV, and PDF maps for x1 and c.

```
VARNAMES: SALT2x1  LOGMASS  PROB
PDF:   -3.000    7.00   0.00387
PDF:   -3.000    7.20   0.00388
PDF:   -3.000    7.40   0.00389
etc ...
```

Figure 14: Example PDF map for simulating distribution of SALT2x1 parameter.

Sim-input keys controlling PDF maps are as follows:

```
GENPDF_FILE:     <fileName>
GENPDF_IGNORE:   <comma-sep list of map(s) to ignore>
GENPDF_OPTMASK:  <optMask>

GENPDF_FLAT:    SALT2x1(-4:4),SALT2c(-0.4:0.5)
     or
GENPDF_FLAT:    SALT2x1(-4:4),SALT2c(-0.4:0.5),RV(1:4),AV(0:2)
```

Only one map-file can be specified, so all maps must reside in this file. `GENPDF_IGNORE` disables a map, and allows for using an `AG` without creating a separate map file which excludes the relevant map. For example,

```
snlc_sim.exe <inputFile> GENPDF_IGNORE RV  \
     GENPEAK_RV 2.2 GENSIGMA_RV .2 .3
```

disables the RV-PDF map, and instead uses an `AG` for R_V: peak at 2.2, $\sigma_{-/+}- = 0.2/0.3$. If both the analytical `AG` and PDF map are defined for the same parameter, the simulation will abort.

GENPDF_FLAT forces a flat distribution for an arbitrary set of SALT2-related (x1,c) and dust-related (RV,AV,EBV) variables. Variable names are comma-separated, and the colon-separated numbers in parentheses are the min & max generated value for each variable. This option overrides other PDF options based on maps, asymmetric Gaussians (x1,c,RV), and exponential (EBV,AV).

GENPDF_OPTMASK options:

- += 1: for HOSTLIB parameters outside map range, extrapolate instead of abort.

In addition to specifying population maps in the GENPDF_FILE, an analytical AG map can be speicified for SALT2 α and β using the same keys as in the sim-input file:

```
GENPEAK_SALT2BETA:   2.1
GENSIGMA_SALT2BETA:  0.22 0.22
GENRANGE_SALT2BETA:  0.4  3.0

GENPEAK_SALT2ALPHA:   0.15
```

The β distribution here is a symmetric Gaussian, and the α distribution is a delta-function since the σ and range are not specified. Internally, the simulation sets $\sigma_\alpha = 0.15 \times 10^{-6}$, and the range to be $0.15 \pm 2\sigma_\alpha$. Both α and β Gaussians are converted into maps as if they had been read from the GENPDF_FILE; thus all PDF maps are processed using the same grid-interpolation code.
To use a command-line override to change the Gaussian function in the GENPDF_FILE, the relevant map must be disabled using the GENPDF_IGNORE key:

```
snlc_sim.exe  <inputFile>  GENPDF_IGNORE SALT2ALPHA   GENPEAK_SALT2ALPHA 0.16
```

4.33.1 Reweight Probability to Enhannce Low-Prob Regions

To simulate bias corrections, low-probability regions are difficult to populate with adequate statistics for a reliable bias correction. The following sim-input enhances low-prob regions:

```
GENPDF_EXPON_REWGT:  <REWGT>
```

where the physical probabilty ($P_{\rm Phys}$) is replaced with $P_{\rm REWGT} = P_{\rm Phys}^{\rm REWGT}$. For example, REWGT=1/2 and $P_{\rm Phys} = 0.01$ (e.g., very red color and or low-stretch) is modified to have $P_{\rm REWGT} = \sqrt{0.01} = 0.1$. In the analysis, each event has a weight of $P_{\rm Phys}/P_{\rm REWGT}$, and this weight is stored in the data file as SIM_WGT_POPULATION. For light curve fitting with snlc_fit.exe, the output FITRES table includes SIM_WGT_POP that is automatically used by the SALT2mu/BBC code.

Prior to using this feature for large sims, always run a short test with and without this feature and verify that distributions reweighted by SIM_WGT_POP match the distribution with REWGT=1.

4.34 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 & τ_V, SALT-II parameters α & β, and the population parameters for shape, color, etc ...

The z-dependence is specified as an additive shift. If the function is simple, you can specify a polynomial function of redshift with arbitrary order. More complex functions can be specified in a file with a z-dependent map. Examples for specifying both types of parameter-shift functions are given in this file,

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

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

```
> 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 (§12.2.3). You can also change the name of the "`ZVARIATION`" file.

The polynomial option can be specified in the sim-input file (without a separate file) to define redshift-dependent parameters with

```
ZVARIATION_POLY:  GENPEAK_VARNAM1      a0,a1,a2     # 2nd order poly
ZVARIATION_POLY:  GENPEAK_VARNAM2      a0,a1,a2,a3  # 3rd order
ZVARIATION_POLY:  GENSIGMA[0]_VARNAM3  a0,a1        # lo-side sigma,linear
ZVARIATION_POLY:  GENSIGMA[1]_VARNAM3  a0,a1        # hi-side sigma,linear
ZVARIATION_POLY:  GENSKEW[0]_VARNAM3   a0,a1,a2,a3,a4
ZVARIATION_POLY:  GENSKEW[1]_VARNAM3   a0,a1,a2

# hard-coded [LEGACY] 3rd order poly still supported:
ZVARIATION_POLY:  GENPEAK_VARNAM1      a0 a1 a2 a3
ZVARIATION_POLY:  GENPEAK_VARNAM2      a0 a1 a2 a3
  etc ...
```

Note that either the `ZVARIATION_POLY` key or the `ZVARIATION_FILE` key is allowed; specifying both results in an abort. Since populations are defined by two `GENSIGMA` values and two `GENSKEW` values, the redshift dependence is specified separately using the index in [] as shown above.

Beware that the function must give a shift of zero at $z = 0$, otherwise the simulation will abort: this ensures that the sim-input parameters are clearly defined at $z = 0$. For example, suppose we want `GENPEAK_SALT2c` to have the form $-0.10 + 0.01z$. The following illustrates the incorrect and correct sim-inputs:

```
GENPEAK_SALT2c:  0.0
ZVARIATION_POLY: GENPEAK_SALT2c -0.1,0.01  # ==> results in abort

GENPEAK_SALT2c:  -0.1
ZVARIATION_POLY: GENPEAK_SALT2c 0.0,0.01  # ==> valid
```

4.35 Estimating PEAKMJD

The SNANA analysis & fitting programs include options for estimating PEAKMJD from the measured fluxes (§5.5.1, Fig. 15), and these options can be applied to both data and the simulation. For non-SNANA analysis codes, it may be useful to include a PEAKMJD estimate in the simulated data files, and thus alleviate the need to run snana.exe to extract a PEAKMJD estimate. The PEAKMJD-estimate options are

```
OPT_SETPKMJD:     16    # Fmax-clump method (default)
OPT_SETPKMJD:     8     # naive Fmax over all epochs
GENSIGMA_PEAKMJD: 3.0   $ Gauss smear (days)
```

If no options are given, the default "Fmax-clump" method is used. Other OPT_SETPKMJD options using Bazin fits (Fig. 15) are implemented in the analysis codes, but are not supported in the simulation. Backward compatibility issue: if both GENSIGMA_PEAKMJD and OPT_SETPKMJD are defined, GENSIGMA_PEAKMJD is used and OPT_SETPKMJD is ignored. See output README file to verify method.

To check residuals with SIMGEN_DUMP feature, add variables PEAKMJD (true value) and PEAKMJD_SMEAR (estimate).

4.36 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/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 § 4.19. 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,

```
#                                out
#           sim-input key        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 χ^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.

4.37 Simulated Output Formats with FORMAT_MASK

Each simulated light curve is written to the directory

```
$SNDATA_ROOT/SIM/[GENVERSION]     # default
   or
[PATH_SNDATA_SIM]/[GENVERSION]    # optional output path with PATH_SNDATA_SIM key
```

For FITS format, two sets of FITS files are created: i) a "HEADER" file containing global information for each SN (SNID, redshift, RA, etc ...) and ii) a "PHOT" file containing all of the light curves, along with meta-data (sky noise, PSF, zeropoint ...). Pointers in the HEADER file are used to quickly extract the appropriate light curve from the PHOT file, without doing a full search for each PHOT row. These FITS 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 with a single integer. Either TEXT or FITS format can be selected, but not both. `TEXT` format is allowed only for interactive jobs; `FITS` format must be used for batch jobs. Here is a quick summary of the bit-mask format options,

```
FORMAT_MASK:   2  # TEXT format
FORMAT_MASK:  18  # 2(TEXT) + 16(RANDOM CID)
FORMAT_MASK:  26  # 2(TEXT) + 8(BLIND) + 16(RANDOM CID)
FORMAT_MASK:  32  # FITS format (default for version >= v9_82)
FORMAT_MASK:  48  # 32(FITS) + 16(RANDOM CID)
FORMAT_MASK: +64  # compact PHOT table
                    (remove GAIN,RDNOISE,SKYSIG_T,PSF_SIG2,PSF_RATIO,ZP_ERR)
FORMAT_MASK:  288  # 32(FITS) + 256(write filterTrans files)
FORMAT_MASK:  2080 # 32(FITS) + 2048(no SPEC.FITS, but keep VERSION.SPEC)

FOMAT_MASK:   10   # 2(TEXT)  + 8(no SIM_XXX output, looks like real data)
FOMAT_MASK:   40   # 32(FITS) + 8(no SIM_XXX output, looks like real data)
```

and the sub-sections below give more details. Note that the 2-bit (TEXT) or 32-bit (FITS) must be included for all options, except for the compact feature (64-bit) that automatically enables FITS format.

4.37.1 FITS Format

"**FORMAT_MASK: 32**" uses the 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, FITS files are portable to any computing platform, and there are public fits-viewing utilities such as "fv."

For data version `VVV`, the file `VVV.LIST` points a set of `HEAD.FITS` files with a one-row summary for each SN. The summary info includes the SNID, sky coordinates, redshift, Galactic extinction, host-galaxy information, and many other quantities. Each `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 to the PHOT-tables, and pointers in the HEAD file (see PTROBS_MIN and PTROBS_MAX) are used to select the appropriate rows from the PHOT table. To help catch pointer mistakes, MJD = -777 is written after each light curve to clearly indicate the end of the light curve.

If submit_batch_jobs is used to generate simulations on mutiple cores, there will be one HEAD and one PHOT file per core, all gzipped. Reading a gzipped FITS file requires the memory of the entire unzipped FITS file; to reduce memory for reading, the FITS files are not merged. For example, 40 1GB FITS files requires only 1GB of memory to read, while merging into one file would require 40 GB memory.

The HEAD and PHOT table columns are shown in § 2.2.

4.37.2 TEXT Light Curve Output

"**FORMAT_MASK: 2**" This option results one outut TEXT file per event. These files are human-readable, and is thus recommended for testing with small interactive jobs. For sims created in FITS format, a small number of light curves can be extracted into TEXT format with command:

```
quick_commands.py -v [GENVERSION] -cidlist_text [cid0],[cid1], ...
  or
quick_commands.py -v [GENVERSION] --cidlist_text TEN   # extract first 10 events
```

4.37.3 Suppress SIM_XXX Info

"**FORMAT_MASK: 10 or 40 (+8 bit)**" Suppress SIM_XXX header info. This option is useful for things like blind-testing photometric classifiers.

4.37.4 Random CID

"**FORMAT_MASK: 18 or 48 (+16 bit)**" Generate random (integer) CID from 1-300,000,000 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 300,000,000. 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 submit_batch_jobs), 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 submit_batch_jobs 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.

4.37.5 Suppressing Spectra Output

To avoid excessive disk volume output for spectra, there are two methods to suppress output without affecting the VERSION.SPEC summary table. The first method is to suppress all spectra

by adding 2048 to FORMAT_MASK. The second method is to pres-scale the number of output spectra with sim-input

```
WRSPEC_PRESCALE: 5  # write 1 of 5 spectra
```

4.37.6 PHOTFLAG Mask

The PHOTFLAG data column is a 4-byte integer mask containing the following user-requested information:

```
# override mask(s) in SEARCHEFF_PIPELINE_EFF_FILE
  PHOTFLAG_DETECT:  <MASK>  # set for each detection
  PHOTFLAG_TRIGGER: <MASK>  # set for epoch satisfying trigger

# overide mask(s) in SIMLIB global header
  PHOTFLAG_SATURATE: <MASK>  # saturation, see NPE_SATURATE in SIMLIB
  PHOTFLAG_SNRMAX:   <MASK>  # set for epoch with max SNR
  PHOTFLAG_NEARPEAK: <MASK>  # set for epoch closest to peak
```

These PHOTFLAG masks can be specified in either an input table file, or the sim-input file; latter has priority if both are specified. Setting PHOTFLAG no impact on the simulation; it simply provides additional information to the analysis programs.

4.37.7 Source of Each Redshift

The output data files include REDSHIFT_FINAL, the best redshift to use for Hubble diagrams. However, the source of this best redshift can be from the SN, from a host-spectrum, from a host photo-z, or from a wrong host. An integer flag, SIM_REDSHIFT_FLAG, is written to the data files, and the intepretations can be found with the following grep command:

```
grep REDSHIFT_FLAG $SNANA_DIR/src/snlc_sim.h | grep define
#define  REDSHIFT_FLAG_NONE      0
#define  REDSHIFT_FLAG_SNSPEC    1
#define  REDSHIFT_FLAG_HOSTSPEC  2
#define  REDSHIFT_FLAG_HOSTPHOT  3
#define  REDSHIFT_FLAG_WRONGHOST 4
```

In the output tables (SNANA,FITRES), this flag is called SIM_ZFLAG; it is written to TEXT, ROOT, and HBOOK files.

4.38 Simulation Dump Options

Here is an overview of the simulation table-DUMP options that produce text-formatted table summaries which are easily read with python. `[GENV]` is the `GENVERSION` argument, and is the prefix for all table-dump files names that are stored in the sim-data folder along with other auxiliary files (`[GENV].LIST` and `[GENV].README`).

table file name	sim-input key: value	comment
Optional:		
[GENV].DUMP	SIMGEN_DUMP: [below]	user specifies variable list
[GENV].NOISE	SIMGEN_DUMP_NOISE: 1	noise per obs: sky,galaxy,source
[GENV].TRAINSALT	SIMGEN_DUMP_TRAINSALT: 1	PEAKMJD table for saltshaker
[GENV].MWCL	SIMGEN_DUMP_MWCL: 1	Color law vs. lam and RV
[GENV].RATE	SIMGEN_DUMP_RATE: 1	z & vol-Rate in .01 z bins
SIMLIB_DUMP_*	SIMLIB_DUMP 1	command line arg; sim quits after dump.
Auto-generated:		
[GENV].SL	---	Strong Lensing
[GENV].DCR	---	Differential Chromatic Correction
[GENV].SPEC	---	TAKE_SPECTRUM

The most commonly used table-DUMP is `SIMGEN_DUMP` that produces a one-row-per-SN summary. This is the only table that requires a user-input list of variable names (§ 4.38.2); the other tables produce standard columns, and column changes require a code update. `SIMLIB_DUMP` (§ 4.38.1) is unique because the key is entered on the command line, and it quits after the dump, and thus is not used with nominal sim generation.

4.38.1 SIMLIB_DUMP Utility

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

```
   snlc_sim.exe  mysim.input  SIMLIB_DUMP 1
        or
   snlc_sim.exe  NOFILE   SIMLIB_FILE <simlib_file>  SIMLIB_DUMP 1

************************************************
  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

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

The sim-input file must contain the SIMLIB_FILE key; no other keys are needed. To avoid creating a sim-input file, the NOFILE argument ignores the sim-input file and reads all arguments on the command line.

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 DUMP_LIBID-[simlib]

where [simlib] is the name of the SIMLIB file that you specify. This file is self-documented like the "fitres" files.

A one-line dump per MJD, which includes ZP in photoelectrons, sky noise converted into mag/arcsec^2, and 5σ limiting mag calculation, can be obtained by setting the 2nd bit of the SIMLIB_DUMP mask,

```
   snlc_sim.exe  mysim.input  SIMLIB_DUMP 2
        or
   snlc_sim.exe  NOFILE  SIMLIB_FILE <simlib_file>  SIMLIB_DUMP 2
```

and the corresponding output file is SIMLIB_DUMP_AVG-[simlib]. To obtain both the LIBID and MJD dump-files, set the SIMLIB_DUMP argument to 3.

4.38.2 Creating SIMGEN_DUMP File

To quickly analyze generated distributions, there is an option to write generated quantities to a table file in TEXT format. For example, to check the generated redshift and SALT-II parameters, add the following to your sim-input file:

```
# dump same SN that are written to data files
  SIMGEN_DUMP: 5  CID  ZCMB  SALT2x0 SALT2x1 SALT2c
     or
# dump all SN, even those rejected by trigger and cuts
  SIMGEN_DUMPALL: 6  CID ZCMB SALT2x0 SALT2x1 SALT2c SIM_SEARCHEFF_MASK
    or
# no need to specify NVAR with comma sep list
# SIMGEN_DUMP: CID,ZCMB,SALT2x0,SALT2x1,SALT2c

# optional pre-scale (e.g., to limit output from DUMPALL)
  SIMGEN_DUMP_PRESCALE: 10
        or
  PRESCALE_SIMGEN_DUMP: 10
```

SIMGEN_DUMP[ALL] produces an auxiliary file [VERSION].DUMP in the same directory as the data files. The dump-file looks like:

```
VARNAMES:  CID ZCMB SALT2x0 SALT2x1 SALT2c
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
```

To list all valid SIMGEN_DUMP variable names to stdout, either specify zero variables or intentionally list an invalid variable name,

```
snlc_sim.exe mysim.input SIMGEN_DUMP 0
snlc_sim.exe mysim.input SIMGEN_DUMP CID,BAD
```

After printing the valid variable names, the program quits.

To switch from SIMGEN_DUMP (in the input file) to SIMGEN_DUMPALL via command line arg, use the "SWITCH" option,

```
snlc_sim.exe  mysim.input  SIMGEN_DUMPALL SWITCH
```

To append dump variables on the command line (or GENOPT using submit_batch_jobs),

```
snlc_sim.exe  mysim.input  SIMGEN_DUMPADD SALT2c,SALT2x1
```

The ADD-list argument must be a comma-separated list, and using this key in the sim-input file will trigger an abort.

To dump the noise components in photo-electrons (source, sky, host, readout):

```
SIMGEN_DUMP_NOISE: 1  # dump noise for obs closest to peak per event & band
SIMGEN_DUMP_NOISE: 2  # dump noise for every obs with defined source flux
```

which outputs a table file ([VERSION].NOISE) in the output data folder.

4.38.3 Model Dump

Model magnitudes can be dumped with the sim-input key

```
GENRANGE_DMPTREST: -20  60  # dump model for this Trest range, 1 day bins
```

This option will dump model magnitudes for a grid of

1. 1-day $T_{\rm rest}$ bins
2. each observer-frame band
3. hard-wired range of shape-parameter values for the selected model (e.g., x_1 for SALT2, $\rm dm_{15}$ for SNOOPY, etc ..).
4. color parameter (c or A_V) is set to zero.

and the first generated redshift is used for the dump. The GENRANGE_DMPTREST option results in an output text file and then the simulation stops. The name of the output file is DUMP_GENMAG_[MODELNAME].TEXT and it contains the generated observer-frame model mag for a grid of {Trest,band,shapePar}.

Instead of generating model mags on a grid, a model mag can be generated for fixed values of {Filter,Trest,ShapePar,Redshift} with the following sim-input keys,

```
GENFILTERS:         V            # pick filter
GENRANGE_DMPTREST:  4.3   4.3    # pick Trest (this key flags the dumpFile)
GENRANGE_SALT2x1:   0.36  0.36   # pick SALT2x1
GENRANGE_REDSHIFT:  0.13  0.13   # pick redshift
```

The SALT2x1 parameter can be replaced with the appropriate model-dependent parameter. To dump rest-frame mags, set the redshift to 2.335E-9 (10 pc).

4.39 Including a Mutlipe Sim-Input Files

A sim-input file can be split into mutliple input files using this keyword in primary input file:

```
INPUT_INCLUDE_FILE:  my2nd.input
INPUT_INCLUDE_FILE:  my3rd.input
INPUT_INCLUDE_FILE:  my4th.input
```

which instructs the simulation to read and parse these additional input files in exactly the same way as the original sim-input file. To see how this might be useful, consider the NONIASED model that has many "NON1ASED:" keywords. The NON1ASED keys can be stripped out into a separate file such as NON1ASED_keys.input, and then included in many sim-input files. Thus a dozen sim-input files can each include NON1ASED_keys.input. To modify or add a NONIASED key for all of the sim-input files, only one file needs to be modified.

INPUT_INCLUDE_FILE can be specified on the command line, but beware that this feature adds an additional sim-input file without overriding previous include files. To replace an already-specified INCLUDE file on the command line, a previous INCLUDE file must be explicitly excluded;

```
snlc_sim.exe <primary_input>  \
    EXCLUDE_INCLUDE_FILE 3rd \
    INPUT_INCLUDE_FILE   other3rd.input
```

The argument '3rd' excludes INCLUDE file names containing this string, and will remove only the my3rd.input INCLUDE file. If the EXCLUDE argument is replaced with 'my', then all of the INCLUDE files would be excluded for this example.

4.40 Multi-dimensional GRID Option

Instead of generating random distributions in the variables describing each SN (redshift, shape 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_SHAPEPAR:    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_TREST:   -20.0  90.0      # test epoch relative to peak (days)
GENFILTERS:       griz

# ---------- Use one of the following models below ----------
# for mlcs2k2
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

# for SALT2
GENRANGE_SALT2x1:     -3.0  3.0
GENRANGE_SALT2c:      -0.3  0.5
GENRANGE_SALT2BETA:    3.2  3.2

# for SIMSED model
SIMSED_SHAPEPAR:  DM15    # replace SIMSED_PARAM key to identify SHAPEPAR
GENRANGE_DM15:   0.6   1.59
SIMSED_COLORPAR:  AV      # replace SIMSED_PARAM key to identify COLORPAR
GENRANGE_AV:  -1.0   2.8
SIMSED_COLORLAW:  RV      # replace SIMSED_PARAM key to identify COLORLAW
GENRANGE_RV:   2.2  2.2
```

and explicit examples of complete sim-input files are in

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

Also see the `sim_SNgrid.pl` utility in §8.1.1. This GRID option allows external (non-`SNANA`) fitting programs to use the `SNANA` models. The original motivation is for the photometric SN id program (§8.1). 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:

$$\mathrm{ILC} = 1 + \sum_{i=1}^{4} \mathrm{ILCOFF}_i \times (\mathrm{INDX}_i - 1) \tag{30}$$

The parameter index $i = 1, 4$ runs over (1) redshift, (2) color (A_V or c), (3) color law (R_V or β) and (4) shape parameter. Each integer index $\mathbf{INDX}_i$ runs from 1 to $\mathtt{NGRID}_i$ for parameter i. Do NOT extend the summation to include the filter and epoch indices. While the physical grid-values corresponding to each $\mathbf{INDX}_i$ 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 $\mathbf{ILCOFF}_i$ are fixed, while the $\mathbf{INDX}_i$ 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 $-2 \le \log_{10}(z) \le 0$ and a logz binsize of 0.01. For $z = 0.1$, $\log_{10}(z) = -1$ and the GRID-index is $\mathbf{INDX}_1 = 100$.

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

```
PTR_I2LCMAG[ILC] = 1 + ((NGRID_FILT * NGRID_TREST) + NWDPAD) * (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 (× 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 NONIASED GRID, there is no physically meaningful shape parameter; this parameter is therefore used to store a sparse index that runs from 1 to the number of NONIASED templates that are specified with the "NON1ASED:" 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').

4.41 TGRIDSTEP: Linear Interpolation of Model Flux

It is sometimes useful to simulate with linear interpolation between model points. An example is to reproduce fakes overlaid on images, where each fake model flux is linearly interpolated from a pre-computed model grid. The sim-input key is

```
TGRIDSTEP_MODEL_INTERP:  4
```

which will compute the model flux on a 4-day grid, and then use linear interpolation to compute the flux at each MJD.

4.42 Marking Sub-Samples

Sometimes it is useful to generate many statistically independent samples. Rather than generating separate versions, there is less bookkeeping to generate one large sample and mark subsamples with sim-input

```
NSUBSAMPLE_MARK:  20
```

which will mark 20 sub-samples with an integer index. This index is automatically included in the output tables for all tables formats.

4.43 Applying Systematic Errors (RANSYSTPAR)

While systematic variations are typically evaluated in the analysis stage, it may be useful to simulate a large number of data-sized samples, each with a different random set of systematic errors applied. In principle, one can define each set of variations in a brute-force manner: e.g., picking random zero points offsets for each simulated version, and specifying "`GENOPT: GENMAG_OFF_ZP <list>`" for each `GENVERSION` (see §12.2.1).

Instead of the brute-force apporach, it is simpler to use the `RANSYSTPAR_XXX` input parameters so that the simulation picks random errors. Here are examples of available options for the sim-input file:

```
#       systematics that are 100 percent correlated among samples
   RANSYSTPAR_SIGSCALE_MWEBV:    0.05     # scale MWEBV
   RANSYSTPAR_SIGSHIFT_MWRV:     0.2      # shift RV
   RANSYSTPAR_SIGSHIFT_REDSHIFT: 1E-4     # shift measured redshift
   RANSYSTPAR_SIGSHIFT_zPHOT_HOST: 0.01   # shift zPHOT of host galaxy
   RANSYSTPAR_SIGSHIFT_W0:           0.1  # shift w
   RANSYSTPAR_SIGSHIFT_OMEGA_MATTER: 0.1  # shift OM

   RANSYSTPAR_RANGESHIFT_W0:           -0.5 0.5   # flat shift range for w
   RANSYSTPAR_RANGESHIFT_OMEGA_MATTER: -0.2 0.4   # flat shift range for OM

   RANSYSTPAR_GENMODEL_WILDCARD: "/path/to/genmodels*"
           # pick random model out of globbed list

   RANSYSTPAR_GENPDF_FILE_WILDCARD: "/path/to/genpdf_files*"
           # pick random GENPDF_FILE out of globbed list
```

```
#             systematics that depend on survey
  GENFILTERS:  griz
  RANSYSTPAR_SIGSCALE_FLUXERR:  0.02     # scale true & measured flux-errors
  RANSYSTPAR_SIGSCALE_FLUXERR2: 0.04     # scale measured flux-errors
  # filter-dependent variants:
  RANSYSTPAR_SIGSHIFT_ZP:           .01 .02 -0.01 .03      # sigma per band
  RANSYSTPAR_SIGSHIFT_ZP(g,r,i,z):  .01 .02 -0.01 .03      # idem
  RANSYSTPAR_SIGSHIFT_ZP(gr,iz):    0.01 0.02     # sig(g&r)=0.01, sig(i&z)=.02
  RANSYSTPAR_SIGSHIFT_ZP(g,riz):    0.01  .03     # sig(g)=0.01, sig(r&i&z)=.03

  RANSYSTPAR_SIGSHIFT_LAMFILT:          8 7 19 9   # sigma per band
  RANSYSTPAR_SIGSHIFT_LAMFILT(g,r,i,z): 8 7 10 9   # idem
```

```
# misc control parameters

# use this random to select sytematic shifts; then switch back to
# user-define RANSEED, or RANSEED_GEN above. submut_batch_jobs uses
# this key to define correlated syst for each of the N samples.
RANSYSTPAR_RANSEED_SYST: 12345

# after ransyst params are picked, reset randoms with fixed seed
# to suppress random fluctuations between GENVERSIONS
RANSYSTPAR_RANSEED_GEN: 12345

# if different surveys use same instrument, force the same
# IDSURVEY-dependent randoms for systematics such as ZP and FilterWave.
RANSYSTPAR_IDSURVEY: 15 # e.g, same shifts for Found and PS1
```

SIGSCALE_XXX parameters are interpreted as follows. A Gaussian sigma is defined as $\sigma =$ SIGSCALE_XXX, and then a random Gaussian number, r, is picked. The FUDGESCALE used in the simulation is $1 + r$. The SIGSHIFT_XXX variables define a Gaussian sigma used to determine a random shift. Finally, RANGESHIFT_XXX define a random shift with uniform probability over the defined range of shifts.

- RANSYSTPAR_SIGSCALE_FLUXERR applies a scale to the true and measured flux-uncertainties to either over- or under-estimate the uncertainties.
- RANSYSTPAR_SIGSCALE_FLUXERR2 applies a scale to the measured flux-uncertainties, but NOT to the true uncertainty used to smear the fluxes.
- RANSYSTPAR_SIGSCALE_MWEBV applies a scale to Galactic extinction.
- RANSYSTPAR_SIGSHIFT_MWRV applies a random shift to Galactice RV.
- RANSYSTPAR_SIGSHIFT_ZP parameters are Gaussian sigmas, not ZP shifts.[18] The simulation uses these sigmas to select a random set of ZP offsets. Changing RANSEED results in a different random set of ZP offsets, and there is no need for the user to pick random offsets.
- RANSYSTPAR_SIGSHIFT_LAMFILT applies a Gaussian-random shift to each filter bandpass.
- RANSYSTPAR_RANSEED_SYST is a flag for submit_batch_jobs.sh to select the same random for each split job. For exampe, consider 50 random data samples for LOWZ,PS1,DES. Split-sample 1 gets the same RANSYSTPAR_RANSEED_SYST value for each survey, split-sample 2 gets a different value, etc...
- RANSYSTPAR_RANSEED_GEN allows fixing the generation seed after systematic variations are selected. This removes statistical fluctuations from multiple GENVERSIONs.

[18] GENMAG_OFF_ZP key can be used to explicitly define ZP shifts.

4.44 Auto-Generated README (valid starting Jan 7 2022)

The simulation produces an auto-generated README file in the same directory as the data files,

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

where [GENVERSION] is the argument of the GENVERSION key. The README content is a yaml-compliant DOCUMENTATION block that includes 4 sub-blocks:

```
DOCUMENTATION:
   OVERVIEW:     # survey, genmodel, node, username, snana_version ...
     ...
   INPUT_KEYS:   # collection of all user input keys and args
      ...
   INPUT_NOTES:  # computed quantities & subtle explanations
      ...
   OUTPUT_SUMMARY:  # summary stats
      ...
```

The first 3 blocks are created during initialization and can therefore be viewed while the simulation is running. The OUTPUT_SUMMARY block is created at the end of the job. The simulation input can be distributed among several INCLUDE files and command-line overrides, and thus tracking the input can be difficult. To enable checking the inputs more easily, the INPUT_KEYS block lists all of the user-inputs in one location regardless of their origin. The INPUT_KEYS can be converted into a single sim-input file with quick-command (§12.2.4)

```
quick_commands.py -v [GENVERSION] --extract_sim_input
   (creates sim_input_[GENVERSION].input)
```

4.45 Understanding NGENEV and NGENLC

There are 3 NGEN quantities reported in the output README from the simulation:

```
NGENEV_TOT:     400   # number of generated events based on reading SIMLIB entry
NGENLC_TOT:     400   # number of generated LCs (until reaching input NGENTOT_LC)
NGENLC_WRITE:   19    # number of LCs passing trigger and written to data files.
```

NGENEV_TOT increments for each cadence read from the SIMLIB/cadence file. For most applications, a light curve is generated for each cadence and thus NGENEV_TOT=NGENLC_TOT as indicated above. For sim-inputs USE_SIMLIB_REDSHIFT and USE_SIMLIB_PEAKMJD (§ 4.7.1), SIMLIB entries failing user-defined GENRANGE_REDSHIFT[PEAKMJD] are counted as a generated events (hence NGENEV_TOT increments), but since the cadence is discarded there is no corresponding light curve and thus NGENLC_TOT is *not* incremented. In this case, NGENEV_TOT > NGENLC_TOT, and the sim runs until NGENLC_TOT matches sim-input request NGENTOT_LC.

The following sim-inputs control generation ranges that must be satisfied for NGENLC_TOT to increment:

```
GENRANGE_REDSHIFT:          <min>  <max>   # required
GENRANGE_PEAKMJD:           <min>  <max>   # required
GENRANGE_RA:                <min>  <max>   # optional
GENRANGE_DEC:               <min>  <max>   # optional
GENRANGE_PEAKMAG_[band]:    <min>  <max>   # optional
```

For SIMLIB files that do not contain REDSHIFT or PEAKMJD, the corresponding GENRANGEs are used to select these quantities and therefore the associated genrange cuts are satisifed. The default GENRANGE_RA[DEC] windows are wide open to select all coordinates in a SIMLIB. User defined GENRANGE_RA[DEC] can result in rejecting SIMLIB entries. GENRANGE_PEAKMAG_[band] are wide open by default; an example use case is if the spectroscopic-selection efficiency reaches zero at a particular mag in a particular band, or if a well-defined mag limit is noticed after analysis.

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/sample_input_files/mlcs2k2/snfit_SDSS.nml .
     or
 > cp $SNDATA_ROOT/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.[19] To create a pdf file showing the light curve fits, see §5.10.1.

5.2 Finding Data Folders

For SNANA analysis codes (snlc_fit.exe, snana.exe, psnid.exe), data folders are specified with

```
&SNLCINP
   VERSION_PHOTOMETRY = 'folder_name'
        or
   VERSION_PHOTOMETRY = 'folder1,folder2,folder3'
       or
   VERSION_PHOTOMETRY = 'folder/subfolder'
```

Multiple folders are useful, for example, when data from a survey are separated into seasons. However, folders from different surveys *cannot* be analyzed together. The SNANA codes automatically search for data folders in

```
# real data
  $SNDATA_ROOT/lcmerge             # public data release
  $SNDATA_ROOT/lcmerge/PREFIX      # public compilation
  argument of PRIVATE_DATA_PATH    # optional/proprietary data location

# sims
  $SNDATA_ROOT/SIM               # default sim output
  paths in $SNDATA_ROOT/SIM/PATH_SNDATA_SIM.LIST  # alternate sim output
```

$SNDATA_ROOT/lcmerge is the default location for public data releases. Compilations consisting of many folders can be under a main folder; e.g., $SNDATA_ROOT/lcmerge/JLA2014 contains 6 sub-folders correspondig to the 6 data samples in this compilation. More generally, for any VERSION_PHOTOMETRY input of the form "PREFIX_SUFFIX," the snana codes search under lcmerge/ and lcmerge/PREFIX. Note that the following &SNLCINP inputs are equivalent,

[19] http://wwwasdoc.web.cern.ch/wwwasdoc/minuit/minmain.html

```
VERSION_PHOTOMETRY = 'JLA2014_CSP'          ! auto-search under JLA2014
VERSION_PHOTOMETRY = 'JLA2014/JLA2014_CSP'  ! explicit reference
```

Since proprietary data are not under SNDATA_ROOT/lcmerge, &SNLCINP key PRIVATE_DATA_PATH specifies an arbitrary location (see §6.3).

The default simulation output is under $SNDATA_ROOT/SIM. However, disk quotas may require alternate sim output locations. The file

```
$SNDATA_ROOT/SIM/PATH_SNDATA_SIM.LIST
```

lists alternate sim-output directories, and the SNANA codes search each directory in this file. Alternate directories are typically in scratch locations with higher (short-term) quotas. PATH_SNDATA_SIM.LIST should list only a few directories, such as per survey, or major projects. Do not use this file to list a directory for each user. Using environment variables is recommended to work on multiple platforms.

5.3 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:

- &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.F90 .
- &FITINP: defines fitting options such as priors, marginalization, and range of $T_{\rm rest}$ in the second fit-iteration. All namelist options are defined and commented inside SNANA_DIR/src/snlc_fit.F90 .

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, Δ, μ), 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.

5.4 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[20]. `&SNLCINP` namelist parameter `NFIT_ITERATION` specifies the number of iterations (2 or 3 recommended). You must always use this option, even if you use the options below. For very high-SNR SNe the first fit-iteration can simetimes fall into a false minimum, leading to pathological fits on successive iterations. This problem can be alleviated with `&FITINP` namelist variable `FUDGEALL_ITER1_MAXFRAC` (set to few percent); on the first fit-iteration this option adds an extra error equal to its value $\times$ the peak flux, thus reducing the chance of finding a false minumum. Consider using `FUDGEALL_ITER1_MAXFRAC` for final fits, or at least as a crosscheck, along with `NFIT_ITERATION=3`.

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. WARNING: this option has not been used for years, so it may be broken.

5.4.1 MINUIT Covariances

As described in the MINUIT manual, each MINUIT fit returns a covariance status, `MNSTAT_COV`, with one of the following values:

```
MNSTAT_COV = 0 -> not calculated
MNSTAT_COV = 1 -> Diagonal approximation only, not accurate
MNSTAT_COV = 2 -> Full matrix, but forced positive-definite
MNSTAT_COV = 3 -> Full accurate covariance matrix (normal convergence)
```

After the last fit iteration, if `MNSTAT_COV` < 3 then the fit is repeated to try getting a better evaluation of the covariances. The final `MNSTAT_COV` value is included in the `FITRES` table.

Roughly $\sim 1\%$ of light curve fits with the SALT2 model result in a covariance matrix, COV(m_B, x_1, c), that is not invertible. The reason is subtle and related to how COV is determined. MINUIT-computed COV is approximated from second derivatives, while the MINUIT errors are determined from better methods (MIGRAD or MINOS). The COV diagonals are thus sometimes different than the error-squared. The FITRES output includes errors and off-diagonal COV terms, and analysis codes must therefore reconstruct COV diagonal terms using error-squared, which doesn't alway correspond to the approximations used for the off-diagonal terms.

There has long been a hack in the SALT2mu.exe program (which implements BBC method) to fix non-invertible matrices. To apply the same hack-fix to the FITRES output of `snlc_fit.exe`, use the following input:

[20] http://wwwasdoc.web.cern.ch/wwwasdoc/minuit/minmain.html

```
&FITINP
   OPT_COVAR_LCFIT = 1   ! fix non-invertible COV with old method
   OPT_COVAR_LCFIT = 129 ! idem, with one-row summary per fix
   OPT_COVAR_LCFIT = 385 ! idem, with full COV dump

   OPT_COVAR_LCFIT = 2   ! newer COV fix using reduced corr.
```

Note that OPT_COVAR_LCFIT is a bit-mask; 1 fixes non-invertible matrix, 128 gives one-row dump per fix, and 256 gives full COV dump per fix. A newer option (OPT_COVAR_LCFIT=2) preserves the reduced correlations (ρ_{ij}) from the MINUIT-computed COV, and uses the errors and ρ_{ij} to determine COV.

5.4.2 Tricks to Avoid Pathological Fits

As samples have grown as large as $\sim 10^7$ for bias-corrections, extremely robust light curve fitting methods are needed to avoid rare pathological fits or code crashes. Here is a list of &FITINP tricks that have been developed over the years (and might be useful with other fitting codes):

- PRIOR_MJDSIG = 10.0 :
  Apply Gaussian prior using pre-fit estimate of PEAKMJD, where estimate is from data file, or one OPT_SETPKMJD options (Fig. 15). OPT_SETPKMJD=16 is recommended because i) there is no fitting, and ii) it is insensitive to catastrophic outlier fluxes.

- PRIOR_SHAPE_RANGE = -4.0, +4.0 :
  Apply flat prior on shape parameter (e.g., $-4 < x_1 < 4$), with half-Gaussian roll-off at edges to preserve continuity in the derivative (beware that discontinuous derivatives can wreak havoc with MINUIT).

- PRIOR_SHAPE_SIGMA = 0.1 :
  Width of half-Gaussian roll-off for PRIOR_SHAPE_RANGE.

- PRIOR_COLOR_RANGE and PRIOR_COLOR_SIGMA
  Analogs of PRIOR_SHAPE_XXX above.

- TREST_REJECT = -15.0, 45.0 :
  Apply rest-frame phase cut. There is under-the-hood logic to ensure that observations do not migrate in/out as PEAKMJD varies with each fit-iteration.

- DELCHI2_REJECT = 16.0 :
  on 2nd fit-iteration, reject observations that contribute $\chi^2 > 16$ (e.g., for preliminary/search photometry that has rare pathologies).

- FUDGEALL_ITER1_MAXFRAC = 0.05 :
  For very high-SNR events (e.g., low redshift) the initial χ^2 can be very large, and sometimes MINUIT is not able to converge. To avoid extremely large initial χ^2, this FUDGEALL parameter adds flux-uncertainty on the first fit-iteration only; the extra uncertainty here is 5% of the maximum flux. A more complete set of FUDGE parameters to downweight/exclude epochs is in §5.13.

- OPT_COVAR_LCFIT = 1 (2) :
  See §5.4.1 for fixing covariances among SALT2-fitted parameters.

- USE_MINOS = T :
  Default fit errors use MIGRAD (USE_MINOS=F). Enabling MINOS gives better error estimates, but also increases the CPU time by a factor of few.

- **Avoid Negative Variance in χ^2 Function:**
  Since variance $V = V_0 + V_1 x_1^2 + (2 x_1 V_{01})$, large-negative x_1 can result in $V < 0$; hence the minimizer function needs protection requiring $V > 0$.

5.5 Initial Parameter Estimates

Before minimization, there are two stages for initializing light curve fit parameters. First is a rough estimate of peak MJD (§5.5.1). The second step is a refined peak MJD estimate and an estimate for the amplitude parameter (§5.5.2).

5.5.1 Rough Estimate of Peak-MJD ($\tilde{t_0}$)

Many light curve fitting programs rely on an approximate estimate of peak-MJD ($\tilde{t_0}$). In particular, the SALT-II fitting program won't work without $\tilde{t_0}$. &SNLCINP namelist variable OPT_SETPKMJD provides several SNANA options to estimate $\tilde{t_0}$, Options are based on analyzing fluxes, and/or a light curve fit in each pass-band using a generic Bazin function.

The $\tilde{t_0}$ value can be determined prior to a SALT-II fit within snlc_fit.exe so that only one program is run for both, or $\tilde{t_0}$ can be extracted into a TEXT-formatted table (column PKMJDINI) for a non-SNANA code as follows:

```
snana.exe NOFILE \
  VERSION_PHOTOMETRY <VERSION>   \
  SNTABLE_LIST 'SNANA(text:key)' \
  TEXTFILE_PREFIX    <PREFIX>    \
  OPT_SETPKMJD <OPT_SETPKMJD>        > stdout.log
```

For simulations, plot residuals with PKMJDINI-SIM_PKMJD.

A summary of the &SNLCINP bit-mask options is given in Fig. 15.

OPT_SETPKMJD=8 does a naive search for max flux and sets t_0 to be the corresponding MJD. This estimate, however, is subject to catastrophic errors if there is just a single outlier flux. To limit impact from outliers, the "Fmax-clump" method (OPT_SETPKMJD=16 or 32) finds max flux in the densest clump of SNR-detections. Each SNR-detection is defined by user-input SNRCUT_SETPKMJD, and MJDWIN_SETPKMJD defines the size of the time window to search for detections. The window is analyzed in sliding steps of 10 days throughout the duration of the survey; e.g., for a 1000-day survey, 96 10-day windows are analyzed, and each window 50 days long. The last 4 windows (97-100) are skipped because they contain no additional information. The window containing the maximum weight-sum among SNR-detections is used to determine max flux, and t_0 is MJD of the max flux observation. Each weight is 1.0 per detection for OPT_SETPKMJD=16, and weight is $\log_{10}(SNR)$ for OPT_SETPKMJD=32. If there are no observations with SNR>SNRCUT_SETPKMJD, the process is repeated with SNRCUT_SETPKMJD$\rightarrow$ 3. The CPU time for this max-flux method is $\sim$10 milliseconds per event.

OPT_SETPKMJD=64 sets $\tilde{t_0}$ to the MJD when the survey trigger is satisfied; e.g., MJD of the 2nd detection for a trigger requiring two detections. This option requires that the survey sets a PHOTFLAG mask for the observation when the survey trigger is satified; &SNLCINP input PHOTFLAG_TRIGGER

specifies this PHOTFLAG trigger mask. To date, no public data sets include a PHOTFLAG trigger mask. However, the simulation can include this information as shown in Fig. 8.

The Bazin fit options are baed on the SNLS CC rate paper (Bazin et al, 2008),

$$f(t) = A[1 + a_1(t - \bar{t}) + a_2(t - \bar{t})^2] \times \frac{\exp[-(t - \bar{t})/T_{\rm fall}]}{1 + \exp[-(t - \bar{t})/T_{\rm rise}]} \ , \tag{31}$$

where the fitted parameters are $A, a_1, a_2, \bar{t}, T_{\rm fall}, T_{\rm rise}$. OPT_SETPKMJD=1 sets $a_1 = 0$ and $a_2 = 0$, while OPT_SETPKMJD=2 floats both a_1 and a_2. The time at peak is obtained by setting the derivative equal to zero: $\tilde{t_0} = \bar{t} + T_{\rm rise} \ln(T_{\rm fall}/T_{\rm rise} - 1)$. Each filter is fit independently and the final $\tilde{t_0}$ is the filter-weighted average. A filter's $\tilde{t_0}$ is dropped from the average if: (1) its max-flux measurement has the smallest S/N ratio among filters and has $S/N < 3$, or (2) its $\tilde{t_0}$ value is more than 30 days away from the average of the other filter-$\tilde{t_0}$ values (tested only if there are 3 or more filters). The max-flux epoch must have $S/N > 3$ to make an estimate of $\tilde{t_0}$. The CPU time os ∼170 millsec per event.

The Bazin fit needs and initial $\tilde{t_0}'$ estimate, which by default is the MJD at maximum flux, MJD(Fmax). However, the Fmax-clump method can be used instead by combining bit-mask options,

```
OPT_SETPKMJD = 17  # 16(Fmax-clump) + 1(Bazin)
```

Once MJD(Fmax) is determined, user-input CUTWIN_TOBS defines an optional MJD-range to select observations for the Bazin fit: the MJD-range is

```
MJD(Fmax)+CUTWIN_TOBS(1) <  MJD <  MJD(Fmax)+CUTWIN_TOBS(2)
```

If $\tilde{t_0}$ cannot be detemermined, SNANA programs abort by default. To avoid aborts, OPT_SETPKMJD += 4, or OPT_SETPKMJD=21 for the example above. The Bazin fit parameters for each band can be included in output tables with OPT_SETPKMJD += 512. Users should compare the $\tilde{t_0}$ estimate (PKMJDINI in output tables) to the final t_0 from a light curve fit (e.g., from SALT-II) and check for outlier $\tilde{t_0}$ values.

Outlier $\tilde{t_0}$ estimates can be identified by visual inspection of the light curve, but the SNANA options may not be sufficient to fix the outliers. The most robust solution is to set the SEARCH_PEAKMJD keyword in the data files, and then run SALT-II fits with OPT_SETPKMJD=0, which sets $\tilde{t_0}$ = SEARCH_PEAKMJD. The advantage of this method is that outlier $\tilde{t_0}$ values can be manually fixed. For simulations, see §4.35 on how to leave a $\tilde{t_0}$ estimate in the output data files.

5.5.2 Refined Estimate of Peak-MJD and Amplitude Parameter

After $\tilde{t_0}$ is determined, a course PEAKMJD grid is searched from −4 to +4 days (2 day bins) w.r.t, PKMJDINI, and PEAKMJD from minimum χ^2 is used as the initial value. Next, the data/model ratio is used to estimate the amplitude parameter: x_0 for SALT2, or distance modulus for MLCS. For very high-SNR events, the initial parameter estimate may not be good enough to prevent catastrophic fits; in this case, try inflating the first-iteration flux uncertainties using FUDGEALL_ITER1_MAXFRAC (§5.13).

Figure 15: &SNLCINP options to estimate PEAKMJD

```
OPT_SETPKMJD  = 0    #  use SEARCH_PEAKMJD value in data file
OPT_SETPKMJD += 1    #  Bazin fit
OPT_SETPKMJD += 2    #  Bazin fit with 2nd order poly (see below)
OPT_SETPKMJD += 4    #  do NOT abort when PKMJD cannot be found
OPT_SETPKMJD += 8    #  PKMJD -> MJD at max-flux (no fit)
OPT_SETPKMJD += 16   #  Fmax-clump method, wgt=1 per obs (see below)
OPT_SETPKMJD += 32   #  Fmax-clump method, wgt=log10(SNR) per obs (see below)
OPT_SETPKMJD += 64   #  PKMJD = MJD_TRIGGER (must set PHOTFLAG bit)
OPT_SETPKMJD += 512  #  save fit-PKMJD params (each band) in SNANA table
OPT_SETPKMJD += 1024 #  DUMP info for each SN

SNRCUT_SETPKMJD  = 5    # for Fmax-clump method only (default=5)
MJDWIN_SETPKMJD  = 60   # for Fmax-clump method only (default=60)
PHOTFLAG_TRIGGER = MASK # for MJD_TRIGGER method (MASK set by survey)
CUTWIN_TOBS = -999,999  # Find MJD(Fmax), then apply TOBS cut for Bazin fit.

SHIFT_SETPKMJD = 5.0    # syst test to shift initial PKMJD by 5 days (default=0)
```

5.6 Galactic Reddening

In the fitting programs, Galactic extinction is applied to the model; i.e., the data are NOT de-reddened. For `snlc_fit.exe`, the extinction is computed at each filter-wavelength bin in the flux-integrals. In `psnid.exe` the Galactic extinction is approximated by the value at the central wavelength. The Galactic extinction covariance is controlled with the `&FITINP` namelist option

```
OPT_COVAR_MWXTERR = 1  ! default: full covariance matrix
```

The reduced $N_{\rm obs} \times N_{\rm obs}$ covariance matrix is 1 everywhere, and each diagonal element is the MWEBV uncertainty at the central wavelength of the observer-frame passband. The covariance matrix is computed and inverted between fit iterations. For each SN the MWEBV value and uncertainty are read from the header. For data, if MWEBV or its uncertainty are zero (or do not exist), then they are internally set to the SFD98 value and $0.16 \times$ MWEBV, respectively. To allow for arbitrary tests in simulations, any value of MWEBV and its uncertainty are accepted. For fitting both data and simulations, Galactic extinction can be disabled by setting either of the namelist parameters `OPT_MWCOLORAW` or `OPT_MWEBV` to zero.

Starting at snana version `v10_29n`, the Galactic reddening law and source of $E(B-V)$ are determined by the `&SNLCINP` namelist parameters

```
&SNLCINP
   RV_MWCOLORLAW  = 3.1 ! default
   OPT_MWCOLORLAW = 94  ! default: CCM89+ODonnel94 (default)
   OPT_MWEBV      = 1   ! default: read E(B-V) from data header

   MWEBV_SCALE = 1.0   ! default: scale all MWEBV values
   MWEBV_SHIFT = 0.0   ! default: shift all MWEBV values
   USE_MWCOR = F  ! default: do NOT correct data, include in fit-model
```

A similar set of parameters controls reddening in the simulation, and additional option-values for these parameters is illustrated in §4.21. If no reddening law is specified (`OPT_MWCOLORLAW`), the default CCM89+ODonnel94 model is used in all programs with the exception of fitting simulations; in this case the default reddening model is that used in the simulation. Similarly, if `OPT_MWEBV` is not specified, the data-header values are used for both data and simulation. In the fitting programs (`snlc_fit.exe` and `psnid.exe`), specifying `OPT_MWCOLORLAW` or `OPT_MWEBV` explicitly in the `&SNLCINP` namelist will override the simulation default.

See §4.21.1 for a description of the Galactic extinction calculation for observer-frame models, and for rest-frame models that require K-corrections.

Finally, `&SNLCINP` namelist parameter `USE_MWCOR` controls whether the data or fit-model is corrected. The default `USE_MWCOR=F` does *not* correct the data, and instead MW reddening vs. wavelength is included in the fit-model. `USE_MWCOR=T` corrects the data approximately, using the extinction at the central wavelength of each filter band, and is then ignored in the fitting model. We recommend using the default, except for special cases such as building de-reddened templates.

5.7 Selecting Filters

By default the snana codes read and store epochs from all defined filters. The fitted filters must be specified in the &FITINP namelist variable

```
FILTLIST_FIT = 'ugriz'
FILTLIST_DMPFUN = 'gri'  ! debug dump for SN model function
FILTLIST_DMPFCN = 'gri'  ! debug dump for chi2 function
```

The debug-dump options should be used only in special cases to debug a fit problem, and only 1 or 2 SN should be fit to limit the size of the stdout.

The filters in FILTLIST_FIT must be defined in the kcor/calib file and in the SURVEY filters defined in the data files. Any undefined filter results in an ABORT. To estimate peak-mags in non-survey filters (e.g., 'abc'), these extra filters must be defined in the kcor/calib file; to prevent the fitting program from aborting, the non-survey filters must be explicitly defined via the &SNLCINP namelist variable

```
NONSURVEY_FILTERS = 'abc'
```

Finally, filters can be selected based on the mean wavelength in the rest-frame ($\lambda_{\rm rest} = \lambda_{\rm obs}/(1+z)$);

```
&SNLCINP
   CUTWIN_RESTLAM = 4000, 8000.
   CUTWIN_LAMREST = 4000, 8000.  ! same as above
   CUTWIN_LAMOBS  = 4000, 8000.  ! cut on <LAMOBS> instead of <LAMREST>
   ...
or
 &FITINP
   RESTLAMBDA_FITRANGE = 4000. , 8000.
   ...
```

Note that RESTLAMBDA_FITRANGE can be used only to reduce the $\lambda_{\rm rest}$ range of the SN model; i.e., you cannot use this parameter to arbitrarily increase the wavelength range. If you really want to increase the $\lambda_{\rm rest}$ range, then you must define your own model and modify the corresponding parameters in one of the model-info files.

5.8 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 §5.12, 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_SHAPE_RANGE(2)**: range of flat prior for shape parameter to prevent crazy values. Typical prior-range values are $\{-0.5, 2.0\}$ for the MLCS2k2 parameter Δ, 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_SHAPE_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` Δ, the first two elements are $-\sigma$ and $+\sigma$ for the asymmetric Gaussian prior, the 3rd element is the Δ value at the Gaussian peak, and the 4th element is the minimum 'flat' probability for all Δ values within `PRIOR_SHAPE_RANGE`. A flat Δ prior is obtained by simply setting the 4th element to 1.0. Setting the 4th element to ~ 0.1 results in a Gaussian prior with a flat tail for large Δ 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.

5.8.1 Initial Values from Existing FITRES file

For the `SNCID_LIST_FILE` option (§5.20.1), the file can contain one of two formats: 1) list format with list of CIDs and nothing else, or 2) `FITRES` (keyed) format that includes CID in first column. For the `FITRES` format, the fit-par columns (PKMJD, x1, c, zHD) are automatically used as initial values to the fit. Using the uncertainties as a prior requires a code update.

5.9 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 (Δ), 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` (§4.36), 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
! $SNDATA_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 (§4.18).

5.10 Plot Utilities

5.10.1 Viewing Lightcurve Fits: `plot_snana_lightcurves.py`

To view light curves,

```
plot_snana_lightcurves.py -v <version> -i <snid1,snid2,snid3,etc..>
```

where `-v` is the data version with FITS (or TEXT) files, and `-i` specifies a comma-separated list of SNIDs to show. Optional arguments include `-p` for private data path, and `-f` for fit-input file to overlay light curve fit. Typing `plot_snana_lightcurves.py` with no arguments gives a help menu.

5.10.2 1D and 2D plots for table: `plot_table.py`

To make 1D plots, 2D plots, and overlays among multiple samples (e.g., data and sim), `plot_table.py` operates on (i) `FITRES` files produced by the light curve fitting program (`snlc_fit.exe`), (ii) `HOSTLIB` files (need to copy and remove DOCUMENTATION block), or (iii) any table with similar keyed format. Extensive help on syntax is obtained with

```
plot_table.py @@HELP
  or
plot_table.py @H
```

Each plot is viewd in a pop-up window by default, or saved to pdf or png file using `SAVE` option.

5.10.3 Plot peakmag vs. redshift: `plot_peakmag_redshift.py`

To quickly compare brightness among many different models, `plot_peakmag_redshift.py` plots peakmag vs. redshift for each band and sim-data folder specified by path wildcard argument. Simulations must be run with `SIMGEN_DUMP` feature and include the `PEAKMAG_[band]` variables.

5.11 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,

```
                          Number of SN passing incremental cut for
   CUT NAME      ITYPE=     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 labeled '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 table[21] 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.F90 | grep '&'
```

where this table is created with `LTUP_SNANA=T` (§12.1) and the variables can be extracted into text format using the `sntable_dump.py` utility (§12.1.3). `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 `SNANA` and `FITRES` table files.

`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`.

[21] The snana table id is 7100 for hbook or SNANA for root.

5.12 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 "cosmology-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, Δ, 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: $\mu = \mu(z, \Omega_M, \Omega_\Lambda, w_0, w_a)$ where z is floated in the fit, and $\Omega_M, \Omega_\Lambda, w_0, w_a$ are fixed by the user. The cosmology can be specified with &SNLCINP namelist parameters `OMAT_REF`, `OLAM_REF`, `WO_REF`, `WA_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 "unconstrained photoZ" fit, involves floating five parameters: μ, z, t_0, Δ, 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.

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 lightcurve-fit ouput where a distance-modulus prior is used with zero uncertainty.

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 cosmology-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[22]. If the host photo-z is described by a zPDF unpacked from quantiles, `PRIOR_ZERRSCALE<100` is simply a flag to use the host zPDF, but the value is not applied to the zPDF.

To switch from a cosmology-constrained-photoZ fit to a 5-parameter unconstrained-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`. `PRIOR_MUERRSCALE` controls the cosmology prior as described in § 5.12.8. The parameter `OPT_PHOTOZ` controls the source of the redshift prior as follows:

```
OPT_PHOTOZ += 1   ! use host zPhot  mean and Gauss sigma
OPT_PHOTOZ += 2   ! use best-z (spec or host photo-z)
OPT_PHOTOZ += 4   ! use photo-z PDF unpacked from quantiles; cubic interp
OPT_PHOTOZ += 8   ! use photo-z PDF unpacked from quantiles; Steffan interp
```

[22]Depending on user interest, non-Gaussian tails may be added later.

```
OPT_PHOTOZ += 10  ! use accurate zSpec, otherwise use quantiles.
OPT_PHOTOZ += 16  ! use photo-z PDF unpacked from quantiles; Linear interp
OPT_PHOTOZ += 32  ! compute INISTP = STD(zPDF)/3.0 (ignore INISTP_PHOTOZ)
OPT_PHOTOZ += 64  ! CHEAT: set initial zPhot=zSpec and skip grid search (for debug)
```

Setting `OPT_PHOTOZ=1` forces using host photo-z Gaussian prior, even if spec-z are available. For quantiles, the mean and stddev are used as Gauss prior. `OPT_PHOTOZ=2` uses the best redshift (spec or photo-z) as a Gaussian prior, and is useful for samples with a mix of spec-z and photo-z redshifts. `OPT_PHOTOZ=4` uses zPDF from quantiles, and ignores subset with accurate spec-z. `OPT_PHOTOZ=6` (2+4) uses accurate spec-z Gaussian prior if available; else it uses quantiles.

For quantile option, existance of spec-z is determined as follows. The most robust method is data variable `MASK_REDSHIFT_SOURCE` set by the survey team creating the data files, and automatically set for simulations. The mask values are as follows: 1: host spec-z, 2: SN spec-z, 4: host photo-z point estimate, 8: host photo-z quantiles. Thus, for example, `MASK_REDSHIFT_SOURCE=9` means that both spec-z and quantiles are available. If this mask is not available, an alternative (less robust) method is to define `&SNLCINP` input `QUANTILE_ZERRMIN = 0.01`; this identifies spec-z if redshift uncertainty is below 0.01.

To get going quickly, some useful examples of setting the namelist options are given below in Fig. 16.

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 (§ 4.23). 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, see § 5.24.

Figure 16: 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

! spec-z fit or host photo-z fit, for mixed spec+photo-z sample
  OPT_PHOTOZ       = 2     ! use REDSHIFT_FINAL as prior
  PRIOR_ZERRSCALE  = 1.0   ! use REDSHIFT_FINAL_ERR in prior
  INISTP_DLMAG     = 0.1   ! float DLMAG
```

5.12.1 Redshift-Dependent Selection in PhotoZ Fits

There is a subtle fitting issue concerning the usable observer-frame filters for which $\lambda_{\rm obs}/(1+z)$ is within the valid $\lambda_{\rm rest}$-range of the light curve model, and for which $T_{\rm rest} = T_{\rm obs}/(1+z)$ are valid. In addition, requirements on quantities such as the min & max $T_{\rm rest}$ 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 $T_{\rm rest}$-dependent quantities.

For regular cosmology fits using spectroscopic redshifts, a list of usable filters & the $T_{\rm rest}$-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 $z_{\rm phot} < 0.072$, *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 χ^2. In the latter case, the minimizer is trapped in this low-χ^2 well, and quite often the minimum occurs at the drop-out boundary $z_{\rm phot} = 0.072$. Another example is in DES & LSST, where *g*-band maps below 3000 Å at redshifts above about 0.5.

To make initial $T_{\rm rest}$-dependent cuts before the photoZ fit has started, the cuts are loosened by a factor of "$1 + Z_{\rm max}$", where $Z_{\rm max}$ = `PHOTOZ_BOUND(2)` is the maximum allowed redshift (specified in &`FITINP`). The $T_{\rm rest}$-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 and a requirement that the min-$T_{\rm rest}$ 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-$T_{\rm rest}$ 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 $T_{\rm rest}$-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 more filters 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 `&FITINP` namelist parameters

```
PHOTOZ_ITER1_LAMRANGE = 4000, 25000 ! 1st-iter obs-frame lambda range
PHOTOZ_BOUND          = 1.0E-6, 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 $T_{\rm rest}$-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 σ_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 $Z_{\rm min} = \lambda_g/9500 - 1 = -0.50$ and $Z_{\rm max} = \lambda_g/3200 - 1 = +0.50$. The g-band is excluded if the 1st-iteration photoZ satisfies $z_{\rm phot} > Z_{\rm max} -$ PHOTODZ_REJECT; in this example, $z_{\rm phot} > 0.45$. For Y-band ($\lambda_Y = 10095$ Å), $Z_{\rm min} = 0.062$ and this filter is excluded if $z_{\rm phot} < Z_{\rm min} +$ PHOTODZ_REJECT, or $z_{\rm phot} < 0.11$. 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 table file:

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

5.12.2 Initial Parameter Estimate

For SN-only photo-z fits, MINUIT is likely to find a local minimum, rather than the global minimum. It is therefore important to provide initial parameter estimates before the MINUIT stage. After using the recommended OPT_SETPKMJD=20 (Fmax-clump) to estimate PEAKMJD, the default opton

performs a 3D course grid-search over redshift (0.04 bin-width), color (4 bins) and stretch/shape (3 bins). If NBINt_COURSE_PHOTOZ>1 (see below), the 3D grid is expanded to a 4D grid search that includes t0=PEAKMJD. The distance parameter is estimated 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. Next we define a flux-scale, $S_F = x_0/\bar{x_0} = 10^{0.4(\mu-\bar{\mu})}$. The chi-squared is $\chi^2 = \sum_i [F^i_{\rm data} - S_F F^i_{\rm model}]^2/\sigma_i^2$, where σ_i is the quadrature sum of data and model errors and 'i' is the epoch index.[23] Analytical minimization is from setting $\partial\chi^2/\partial S_F = 0$, which results in

$$S_F = \sum_i (F^i_{\rm data} F^i_{\rm model}/\sigma_i^2) / \sum_i (F^i_{\rm model}/\sigma_i)^2 \tag{32}$$

The resulting x_0 or μ value is used to re-evaluate the χ^2 at this grid-point. The four initial values ($z_{\rm phot}$, color, stretch, distance) are set based on the minimum χ^2 from the 3D loop over {redshift,color,stretch}. A correction to the S_F formula (Eq. 32) is needed when there is a cosmology prior (Eq. 33 in § 5.12.8) but this has not yet been implemented. After the course-grid search, a 1D PEAKMJD grid is searched in 0.5 day bins (-2 to $+2$ days).

The number of bins and range per variable have default values, and the default values can be altered with these inputs:

```
&FITINP
  NBINc_COURSE_PHOTOZ = nnn ! default=4
  NBINs_COURSE_PHOTOZ = nnn ! default=3
  NBINt_COURSE_PHOTOZ = nnn ! default=1 (fixed at PEAKMJD from OPT_SETPKMJD)
  NBINz_COURSE_PHOTOZ = nnn ! default=43*[zph_max/(1+zph_max) - zph_min/(1_zph_min)]
  SCALE_NBIN_COURSE_PHOTOZ = nnn  ! scale number of bins for each param (default=1)

  RANGEc_COURSE_PHOTOZ = x,y ! default = -0.2, 0.25
  RANGEs_COURSE_PHOTOZ = x,y ! default = -3.0, 1.0
  RANGEt_COURSE_PHOTOZ = x,y ! default = -15, 15 (only if NBINt_COURSE_PHOTOZ>1)
  RANGEz_COURSE_PHOTOZ = x,y ! default = -9,-9 -> computed each event using prior
```

Beware that increasinging the number of bins can result in significantly longer fit times, expecially if NBIN[par]_COURSE_PHOTOZ is increased for multiple parameters; it is therefore good practice to run a CPU benchmark before submitting long jobs.

It is best to let the code internally compute NBINz and RANGEz for each event, and manual overrides should be done only for testing. Also note that the redshift grid increments in bins of $z_{\rm phot}/(1+z_{\rm phot})$ instead of $z_{\rm phot}$, in order to search finer $z_{\rm phot}$ bins at lower redshifts. If there is no host prior, then it is sensible to modify NBINz_COURSE_PHOTOZ and RANGEz_COURSE_PHOTOZ and re-process a large sample. However, a host prior results in internally altering these parameters for each event, and therefore overriding the defaults should be done only for testing individual events and not for a large sample. To see the actual bins used, grep (or search) "binsize" from stdout and visually examine output that looks like

```
  7      peakmjd  bins from 56972.31 to 57002.31   (binsize=   5.00)
 23  zPH/(1+zPH)  bins from    0.010 to    0.545   (binsize=  0.024)
  4      color    bins from   -0.200 to    0.250   (binsize=  0.150)
  3      shape    bins from   -3.000 to    1.000   (binsize=  2.000)
```

[23] See R4SN_FFSUM_XXX variables in snlc_fit.F90.

UNDER DEVELOPMENT: The second option (MCMC) processes all 5 parameters simultaneously. Initial `STEP` sizes are 1/3 of the range, and every 20 evaluations the `STEP` size is multiplied by 0.90. For each evaluation, the parameters are shifted by `STEP`$\times(R_{01} - 0.5)$, where R_{01} is a random number betweeo 0 and 1. For each trial $z_{\rm phot}$, a random distance shift (`dmu`) is added to the cosmology-constrained distance (-4 to $+4$ mag). To ensure reproducibility, the same random seed is used to initialize the random sequence for each event.

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

5.12.3 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 χ^2, and can cause fitting pathologies. This pathology is treated by smoothly transitioning the model error between ± 200 Å of the transition wavelength ($\bar{\lambda}$). The 200 Å 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).

5.12.4 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 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.

5.12.5 Including the log(σ) Term in the χ^2

The default fits minimize the usual function $\chi^2 = \sum_i \Delta F_i/\sigma_i^2$ where ΔF_i is the data-model flux-difference for observation i, and σ_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 χ^2, the actual term added is $\Delta\chi^2_{\sigma_i} = 2\ln(\sigma_i/\sigma_i^{\rm last})$ where $\sigma_i^{\rm last}$ is the uncertainty from the previous fit-iteration. Typically $\Delta\chi^2_{\sigma_i}$ adds ~ 1 to the total χ^2, but sometimes $\Delta\chi^2_{\sigma_i}$ can be very large (positive or negative) if a fitted uncertainty undergoes a significant change between fit iterations. Fits with a large $\Delta\chi^2_{\sigma_i}$ value can be rejected using the `&FITINP` namelist variable `FITWIN_CHI2SIGMA`, or using `SIGCHI2` variable in output FITRES table. Values of $\Delta\chi^2_{\sigma_i}$ 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 $\sigma_i^{\rm last}$ is undefined and $\Delta\chi^2_{\sigma_i}$ 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 $\Delta\chi^2_{\sigma_i}$ is defined for each filter in the last fit-iteration.

5.12.6 Avoiding Filter Drop-outs at High PhotoZ

A potential problem with photo-z fitting is that low-redshift events can result in high-redshift $z_{\rm phot}$ due to filter drop-outs. For example, if the u and/or g band are outside the SED model range at high redshift, these bands are dropped and the resulting χ^2 is artifically small because it ignores the rest-frame UV region. An approximate fix is to use the same UV-extrapolation feature used in the simulation (§4.29.2),

```
&FITINP
   UVLAM_EXTRAPFLUX    = 500
   RESTLAMBDA_FITRANGE = 2000, 13000
```

In this example, the model flux at the bluest wavelength (at each phase) is linearly extrapolated to zero flux at $\lambda = 500$ Å. The RESTLAMBDA_FITRANGE should be set so that all bands are defined at the highest allowed $z_{\rm phot}$ set by the PHOTOZ_BOUND input.

5.12.7 4-Parameter Fit Using Host Galaxy Photo-z

To run a 4-parameter light curve fit with the host-galaxy photo-z (instead of spectroscopic redshift), run `snlc_fit.exe` with

```
&SNLCINP
   USE_SNHOST_ZPHOT = T
```

Note that this will *not* perform a 5-parameter photo-z fit, but instead uses the already determined host-galaxy photo-z as a fixed redshift in the fit. If any photo-z fit flags are set with `USE_SNHOST_ZPHOT=T`, the code will abort.

5.12.8 Cosmology Prior and `PRIOR_MUERRSCALE`

For improved redshift fitting, a cosmology prior can be included; for example,

```
&SNLCINP
   OMAT_REF =  0.3, 0.03  ! value and uncertainty for prior
   W0_REF   = -1.0, 0.1   ! idem
   WA_REF   = -0.4, 0.2   ! idem
```

uses $\sigma_{\Omega_M} = 0.03$, $\sigma_{w0} = 0.1$ and $\sigma_{wa} = 0.2$ to calculate the μ-error for the photoZ at each fit-iteration. The χ^2 prior term is

$$\chi^2_{\mu-\text{prior}} = \left[\frac{\Delta\mu}{\sigma_\mu \times \texttt{PRIOR\_MUERRSCALE}}\right]^2 \tag{33}$$

where

- $\Delta\mu = \mu_{\text{Tripp}} - \mu_{\text{theory}}$, with $\mu_{\text{Tripp}} = m_x + 0.14x_1 - 3.2c$ and μ_{theory} is computed using z_{phot} and cosmology parameters from `&SNLCINP` namelist inputs `OMAT_REF`, `W0_REF`, `WA_REF`;
- $\sigma_\mu^2 = [\delta_\mu(\Omega_M)]^2 + [\delta_\mu(w_0)]^2 + [\delta_\mu(w_a)]^2 + [2\rho_{w0wa} \times \delta_\mu(w_0) \times \delta_\mu(w_a)]$ where $\delta_\mu(\Omega_M) = \mu(\Omega_M + \delta_{\Omega_M}) - \mu(\Omega_M)$ with $\Omega_M =$ `OMAT_REF(1)` and $\delta_{\Omega_M} =$ `OMAT_REF(2)`, and similarly for $\delta_\mu(w_0)$ and $\delta_\mu(w_a)$, and $\rho_{w0wa} = -0.8$ is hard-coded (see `RHO_WOWA` in `snlc_fit.F90`).
- `PRIOR_MUERRSCALE` is a `&FITINP` namelist input to soften the cosmology prior by inflating the computed distance uncertainty σ_μ.

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 $\bar{\lambda}_{\rm obs}/(1+z) < 3900$ Å. This option excludes data as if it were not part of the data file; hence the corresponding passband is not used for spectral warping, $T_{\rm rest}$ 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 $T_{\rm rest}$ sampling requirements. In evaluating the lightcurve fit-χ^2, an additional uncertainty of 100× 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 gives 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 ($\lambda_{\rm rest}$-range) to the fitting program. There are two different ways to change the $\lambda_{\rm rest}$-range, but note that you can only make the $\lambda_{\rm rest}$-range *more restrictive*; i.e, you cannot arbitrarily loosen the $\lambda_{\rm rest}$-range for a SN model. The two equivalent namelist options to change the $\lambda_{\rm rest}$-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 $\lambda_{\rm rest}$-ranges 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 $\lambda_{\rm rest}$-range for a particular SN model,[24] 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., `mlcs2k2.LAM2800`).

[24] All maintenance warranties are null & void if you change the default $\lambda_{\rm rest}$-range.

5.15 Better peak Flux Estimate

While the flux values are from the data, the estimate of the peak flux (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 → 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

$$w_i \equiv [\sigma_i^2 \times (|T_{\rm rest}| + 1)]^{-1} \ , \tag{34}$$

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

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 §5.5.1, 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 (MJD 53616) is interpolated for SNID=1, two epochs (53610,53626) are interpolated for SNID=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
```

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 (§12.2.3) 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 ($M^\star$): 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 $M^\star$ and associated colors include intrinsic variations. This section focuses on obtaining the true $M^\star$ 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
SHAPEFIX_FITRESTMAG  = -0.2  ! => fix SHAPEPAR for PEAKMAG calc
DLAMTOL_FITRESTMAG   = 100   ! include obs bands with 100 A of closest bands
```

The kcor/calibration file must include these extra (UBV) filters in addition to the observer-frame filters. Each $M^\star_{U,B,V}$ 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 $M^\star_{U,B,V}$ can be determined. For example, suppose that the bluest observer-frame filter is g band with a central wavelength of $\lambda_g = 4800$ Å, and the rest-frame U band has $\lambda_U = 3600$ Å. If `LAMEXTRAP_FITRESTMAG`= 0 (default) then the minimum redshift for which $M^\star_U$ can be determined is $z_{\rm min} = 4800/3600 - 1 = 0.333$; with `LAMEXTRAP_FITRESTMAG`= 250 Å we have $z_{\rm min} = 4800/(3600 + 250) - 1 = 0.247$. Similar logic is applied to the reddest filter.

`DLAMTOL_FITRESTMAG` allows the inclusion of multiple bands with similar central wavelengths. For example, consider a survey with G and g bands, and central wavelengths of 4800 Åand 4840 Å, respectively. If g is the closest band to the desired rest-frame band, then G is exluded by default. However, with `DLAMTOL_FITRESTMAG=100` both G and g are included.

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 $M^\star$ 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 $M^\star$ is computed as follows,

$$M^\star_X \equiv \text{modelMag}(T_X,\ t_0,\ s_0,\ \text{color},\ \text{distance}) - \mu_z \tag{35}$$

where T_X is the filter-transmission for filter $X = U, B, V$, and s_0 is replaced by `SHAPEFIX_FITRESTMAG` if this namelist parameter is set. The Galactic extinction `MWEBV` is set to zero for this rest-frame modelMag calculation. μ_z is the calculated distance modulus using the known redshift and an assumed cosmology. The subtraction of μ_z is done so that the $M^\star_X$ have reasonable values (~ -19), but it has no impact on the rest-colors since the same μ_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 (e.g., U,B,V). For each filter in a 2-band fit, the maximum S/N is required to satisfy the `&SNLCINP` namelist cut-variable `CUTWIN_SNRMAX2`. If both bands fail `CUTWIN_SNRMAX2`, the rest-mags are not evaluated and `M0_X=999`.

5.19 Peak-Mag Crosschecks: FITMAGDIF

To check the best-fit model magnitudes there is an option to compute a given observer-frame magnitude in two independent ways : 1) fitting only the one band, and 2) fitting the other bands and extrapolating the model. An example of this FITMAGDIF option is as follows,

```
FILTER_FITMAGDIF = 'u'
FILTLIST_FIT     = 'ugr'
```

The first NFIT_ITERATION fits are done with all of the FILTLIST_FIT (ugr) filters so that the time of peak brightness (t_0), stretch and color parameters are well determined and fixed for the fit-iterations that follow. The next fit-iteration is done only with filter "FILTER_FITMAGDIF" (u) in which the flux-scale (x_0 or μ) is the only floated parameter. The final fit-iteration is done with all of the FILTLIST_FIT (ugr) filters, but with u-band deweighted so that it contributes nothing to the χ^2 and only the g & r bands are included in the fit. The quantity MAGDIF_u and its erros are computed as

```
MAGDIF_u = peakMag(u, extrapolate gr) -  peakMag(u only)
         = peakMag(predicted)         -  peakMag(measured)
```

The MAGDIF variable is stored in the output FITRES table.

5.20 Selecting SNID(s), Field(s), and Telescope(s)

All namelist variables discuss here are in &SNLCINP.

5.20.1 Selecting/Ignoring SNID

By default all candidate IDs (CID) are processed. Here are some example on how to select a range of CIDs or a list of CIDs, and to ignore a list of CIDs:

```
CUTWIN_CID    =   700, 2000       ! CID integer-range to process
   or
SNCID_LIST    =   5944,   10550   ! integer list of SN to process
   or
SNCCID_LIST   =  '5944', '10550'  ! same as above, using strings
   or
SNCID_LIST_FILE = 'myCIDs.list'   ! file with list of CIDs to process;
                                  ! separated by space or <CR>
   or
SNCID_IGNORE  =   4524,   8151,   7017   ! list of SN to ignore
SNCCID_IGNORE =  '4524', '8151', '7017'  ! same as above with strings
SNCID_IGNORE_FILE = 'reject.list'        ! list of CIDs to ignore;
                                         ! separate CIDs by space or <CR>

OPT_SNCID_LIST = 1    ! ignore SN cuts & process all CIDs in list
OPT_SNCID_LIST = 2    ! if FITRES format, use SALT2 params as initial values
                      !    in LCFIT
OPT_SNCID_LIST = 3    ! 1+2: both of the above
OPT_SNCID_LIST = 5    ! 1+4: process CIDs in list;  use all FITPAR +_ SIGMA
                      !   as Gauss prior; force same bands for photo-z fit
OPT_SNCID_LIST = 7    ! 1+2+4:

MXEVT_PROCESS = 20    ! stop processing after 20 events
MXEVT_CUTS    =  5    ! stop processing after 5 events pass cuts
```

Internally all CIDs are strings. If the CIDs are integers then you can use SNCID_LIST and SNCID_IGNORE, otherwise you must use the character-CID variables "SNCCID_" and put each CID in quotes. A zero or blank acts as a terminator. For example, SNCID_LIST = 5944, 0, 1032, 10550 would process SN 5944 and ignore the rest.

CUTWIN_CID gives the integer range. For CIDs that are integers, you can specify the integer range without worrying about the internal string conversions. For CIDs that are strings, SNANA internally assigns integer CIDs that are sequential starting from 1. For example, consider the SNLS CIDs that are strings (e.g., 04D1cc); CUTWIN_CID = 1,10 would process the first 10 SNe. However, for a survey using integer CIDs that start at 1000, CUTWIN_CID = 1,10 would discard all SNe.

Using any of the "_LIST" options (e.g., SNCID_LIST), the analysis codes internally set CUTWIN_CID=0,0 so that only the list CIDs are processed. For large sets of events, it is more practical to store CIDs in a file and use SNCID_LIST_FILE, which allows two kinds of formats: 1) list of CIDs with no other strings or keys, and 2) FITRES formatted table file with VARNAMES key; the CID column is the list of CIDs.

If the argument of SNCID_LIST_FILE is in table format, OPT_SNCID_LIST provides several additional options that are motivated by systematics studies. Setting the 1-bit of OPT_SNCID_LIST ignores all cuts and forces all CIDs in the SNCID_LIST_FILE to be processed; example use is for systematics tests with small calibration shifts that could migrate events in and out of SNR or SALT2 parameter cuts. Beware that while SN-select cuts are ignored, epoch-select cuts are still applied. The 2-bit of OPT_SNCID_LIST uses the fitted SALT2 parameters as initial values for LC fit. The 4-bit of OPT_SNCID_LIST defines a Guassian prior on each SALT2 fit parameter using the previously fitted value and uncertainty in SNCID_LIST_FILE for the Gaussian mean and σ. For photo-z fits, the list of filters (BANDLIST column in FITRES file) is forced in the fit. For spec-z fits, the same filters are used without forcing the issue.

Finally, MXEVT_PROCESS stops processing after this many events, regardless of of how the other "CID" inputs are set.

5.20.2 Selecting SNID List with Command Line Override

To select a few CID values on the command-line, use a comma-separated list:

```
snana.exe <myInputFile>  SNCID_LIST 5,19,184
   or
snana.exe <myInputFile>  SNCCID_LIST 5,19,184
   or
snana.exe <myInputFile>  SNCCID_LIST 2004cx,2002cx,2005he
  or
snana.exe <myInputFile> MXEVT_PROCESS 5
  or
snana.exe <myInputFile> MXEVT_CUTS 5
```

and this syntax also works for the snlc_fit.exe and psnid.exe programs. SNCID_LIST works only for integers, so it is safer to always use the character-string SNCCID_LIST, and there is no need to use quotes for command-line options.

5.20.3 Selecting Fields and Overlapping Fields

By default all fields and overlapping fields are used. Using SDSS 82N and 82S fields as an example, fields are selected by

```
SNFIELD_LIST  = '82S'   ! select only 82S pointings (ignore 82N)

CUTWIN_NFIELD = 1, 1    ! select only SN with no 82N/82S overlap
  or
CUTWIN_NFIELD = 2, 2    ! select only SN with 82N+82S overlap
```

SNFIELD_LIST determine which field-pointings to use for lightcurves; these fields must be specified in SURVEY.DEF, otherwise the code aborts. In the first example above, only 82S pointings/epochs are used, and overlap pointings on 82N are discarded. SNe that overlap 82S and 82N are used, but only the subset of 82S pointings are selected. Similarly, SNFIELD_LIST='82N' would select the epochs with an 82N pointing.

CUTWIN_NFIELD=1,1 picks out SNe in which all observations are on the same pointing; i.e., no overlaps. CUTWIN_NFIELD=2,2 picks out only those SNe that overlap 82N and 82S.

Another example of selecting multiple fields is as follows:

```
SNFIELD_LIST  = 'C1', 'C2', 'C3'
  or
SNFIELD_LIST  = 'C1+C2',  'C3'
  or
SNFIELD_LIST  = 'C1+C2+C3'
  or
SNFIELD_LIST  = 'C1 C2 C3'
```

where the above three commands are equivalent. Either of the last two commands can be used as a command-line override with an arbitrary number of fields.

Even if you don't use the selection options above, there are table variables (§12.1) that can be used to apply some of these selections after the fitting: `NFIELD_OVP` is the number of overlapping fields for each SN, and `FIELD` is the name of the field(s). For SN on overlapping fields, such as 82N and 82S, `FIELD = '82N+82S'`. The order of the overlapping fields in the `FIELD` string is the same as that in `SURVEY.DEF`. The 'append' option in the sntable-dump utility (§12.1.3) can be used to add `NFIELD_OVP` into the text-fitres file. Note that the `FIELD` string cannot be appended to the text-fitres file (sorry!).

5.20.4 Selecting MJD Ranges

There are three `&SNLCINP CUTWIN` parameters to define MJD ranges:

```
CUTWIN_MJD         = 53600, 53700  ! use obs only in this MJD range
CUTWIN_MJD_EXCLUDE = 53622, 53628  ! excluce obs in this MJD range
CUTWIN_MJD_INCLUDE = 53630, 53660  ! require at least 1 MJD in this range
```

The first two `CUTWIN` parameters define epochs to select. The last cut (`CUTWIN_MJD_INCLUDE`) does not pick epochs, but rejects the entire light curve if there is no observation in the defined window.

5.20.5 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 §5.20 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 ($\sim$ 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.

5.21 Mag-Shifts in Zero Points and Primary Reference Star

The snlc_fit.exe program provides an interface to modify zero-point (ZP) offsets as well as the primary magnitudes. The primary mags can be individually adjusted using &SNLCINP namelist options

```
MAGOBS_SHIFT_PRIMARY  = 'B .02  V .01'
MAGOBS_SHIFT_PRIMARY  = 'B .01  V .01 R 0.01'
MAGOBS_SHIFT_PRIMARY  = 'BVR 0.01'    # same as above
MAGREST_SHIFT_PRIMARY = 'B .02  V .01'
```

which shifts the primary mags by 0.02 and 0.01 for B and V, respectively. 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 more convenient since you can use the same K-correction tables.

There are two ways to introduce ZP shifts. First, specify ZP 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 ZP 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 .01  i .008'
MAGOBS_SHIFT_ZP = 'g .01  r .01  i .01'
MAGOBS_SHIFT_ZP = 'gri .01'    # same as above
```

Note that the MAGOBS_SHIFT_ZP shifts are added to those in ZPOFF.DAT.

Wavelength-dependent shifts can be applied with polynomial parameters in &SNLCINP,

```
MAGOBS_SHIFT_ZP_PARAMS      = 0, 0.02, -0.01
MAGOBS_SHIFT_PRIMARY_PARAMS = 0, 0.01, 0
```

which applies a ZP shift of $0.02\lambda_{\rm obs} - 0.01\lambda^2_{\rm obs}$, and a PRIMARY mag shift of $0.01\lambda_{\rm obs}$. Note that $\lambda_{\rm obs}$ is the central wavelength in MICRONS, not Å, so that the PARAMS coeffients are about the size of the mag-shift.

5.22 Fudging the FLUXCAL Offsets and Uncertainties

Filter-dependent offsets and errors can be added to the calibrated fluxes and errors 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'
FUDGE_MAG_ERROR      = 'u .01  g .011  r .009 i .01  z .018'
FUDGE_MAG_ERROR      = 'ALL 0.02'       ! add .02 mag err to all bands
FUDGE_FLUXERR_SCALE  = 'g 1.04 r 1.02'  ! scale fluxerr in g,r bands
FUDGE_FLUXERR_SCALE  = 'ALL 1.03'       ! scale all bands by 1.03
```

These fudges are intended for systematic studies; 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.

5.23 Updating the Filter Transmission for each SN

In 2009 the SNLS reported that their filter transmission depends on the focal plane position.[25] 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 fillter 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.

[25] Regnault et al., **A&A 506** , 999 (2009).

5.24 Redshift Shifts

The following &SNLCINP inputs apply systematic shifts to redshifts:

```
REDSHIFT_FINAL_SHIFT = 0.0001  ! 1. global shift to all redshifts
HOSTGAL_SPECZ_SHIFT  = 0.0002  ! 2. apply only to host specz
HOSTGAL_PHOTOZ_SHIFT = 0.003   ! 3. apply only to host photoz
```

The first option is applied globally to all redshifts, regardless of how they are measured. For a mix of host-galaxy spectroscopic and photometric redshifts, options 2 and 3 apply to the respective z-shift method. For spec-confirmed SNe that have no host-galaxy information, only the 1st option applies. All z-shifts are applied before light curve fitting.

5.25 Shifting the Mean Filter Transmission Wavelength

Each band can be indpendenlty shifted via &SNLCINP input

```
FILTER_LAMSHIFT = 'g -3.4 r 5.6 i 12.3'
```

5.26 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 distance-modulus for each fitted light curve. Works for any fit-parameter: `AV, DELTA, PEAKKMJD, PHOTOZ`. Include the colon, or you will be overwhelmed by the screen dump.

- `grep ":x0" fit*.log | grep fitpar`
  lists the SALT2 amplitude for each fitted light curve.

5.27 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 appear in output table files.

5.28 Over-Riding Information in Data Files

5.28.1 Over-Riding Header Information: HEADER_OVERRIDE_FILE

For SNANA analysis programs (snana.exe, snlc_fit.exe, psnid.exe), here we describe how to over-ride header information in the data files without re-making the data files. This feature is useful to study the impact of data changes without altering a released data set. Beware that this feature is intended only for internal use by analysis teams, and should not be part of a publicly released data set; see Sec. 5.28.5 on updating data files with override information.

The header information includes any variable in a *HEAD.FITS file: host properties, peculiar velocities, redshifts (spec-z and photo-z), Galactic extintion, and many more. This override feature does NOT work on epoch-dependent quantities in the *PHOT.FITS files, but mag and flux corrections can be applied as described below in Sec. 5.28.4.

The input key for the analysis programs is

```
&SNLCINP
  HEADER_OVERRIDE_FILE = 'myUpdates.dat'  ! real data only
      or
  HEADER_OVERRIDE_FILE = 'myUpdates1.dat,myUpdates2.dat  ! real data only
      or
  SIM_HEADER_OVERRIDE_FILE = 'sim_overrides.dat'  ! sim data only
```

```
more myUpdates.dat
VARNAMES: CID  REDSHIFT_HELIO  REDSHIFT_HELIO_ERR  MWEBV   EXTRA
SN:   1346137      0.246        0.001              0.021    45
SN:   1346387      0.533        0.0011             0.038    46
etc ...
```

In the first example (muUpdates.dat), three header values are updated: helio redshift, the uncertainty, and Galactic extintion. The EXTRA column does not correspond to a known data variable and is thus ignored, which means that extra diagnostic variables are allowed to be mixed with data variables. For SNe missing from the update file, the original values in the data file header are preserved. The first column name can be either CID or SNID. The second example illustrates a comma-separated list of override table files; beware that the variables in each override file must be unique (except for CID).

Data-override variables must exist in the original data file, with the exception of photo-z quantiles (§4.23.8). Quantiles can be overwritten if they already exist in the original data files, or appended if the original data files do not include quantile information. For quantile overrides using HOSTGAL_ZPHOT_Qppp (where ppp=000, 010, ... 100), make sure to also include an NZPHOT_Q column with each value set to 11.

A common mistake is to accidentally replace data variable names with names in the output FITRES tables: unfortunately the variable naming is not consistent. Such substituion usually results in an abort since no valid override variables are found.

5.28.2 Over-Riding by HOSTGAL_OBJID instead of CID

OVERRIDE table row identification by CID (or SNID) is robust for real data because real data event names should not change (or change rarely with reprocessing). To modify host properties for simulated data, however, would require an intermediate script to create a new HEADER_OVERRIDE_FILE table for each simulated data set and for each bias-correction sim. To simplify altering host properties for sims, particularly photo-z information, the HEADER_OVERRIDE_FILE can be keyed by HOSTGAL_OBJID or GALID for galaxies in an entire HOSTLIB_FILE, e.g.

```
VARNAMES: GALID  HOSTGAL_PHOTOZ HOSTGAL_PHOTOZ_ERR
SN:  104754   .111   0.011
SN:  104049   .222   0.022
SN:  463249   .333   0.033
etc ...
```

This single table override file is valid for simulated data regardless of which random CIDs are chosen. This galaxy-based identification can be used for real data as well, but beware that host-less events cannot be modified using the GALID key.

5.28.3 Over-Riding Table Information in BBC/SALT2mu Input

The header-override list may include variables that are used later in BBC or cosmology fitting, but not used in light curve fitting; e.g., host-galaxy properties and peculiar velocity. To avoid losing time re-running identical light curve fits for such variable changes, it can be more efficient to make substitutions at the BBC/SALT2mu.exe stage using BBC-input

```
datafile_override=over1.dat,over2.dat,etc
```

Any variable in the input table (see datafile arg) can be included in an override table. For VPEC (VPEC_ERR) changes, the Hubble diagram redshift (zHD, zHDERR) is internally re-computed by subtracting the original v_{pec} and adding the override value. To update host matches, make sure to include the following variables in the header-override file:

- All of the HOSTGAL_XXX, except for HOSTGAL_SB_FLUXCAL_[band] (because SB depends on SN coordinates rather than host-match)
- All of the HOSTGAL2_XXX
- REDSHIFT_HELIO and REDSHIFT_HELIO_ERR
- REDSHIFT_FINAL and REDSHIFT_FINAL_ERR
- possibly VPEC

5.28.4 Epoch-Dependent Mag Corrections

Since photometry is typically performed separately for each passband, SED-dependent corrections are often missing for internal data releases. If SED-based corrections are computed as an afterburner task, it may be convenient to avoid re-making data files and instead read a supplemental MAGCOR file until a final data release is ready.

External epoch-dependent mag-corrections are applied to the FLUXCAL, but not to FLUXCAL_ERR, and thus there is an implicit assumption that changes are small ($\sim 1\%$). For large changes, the data files should be re-made with the new errors as well. The syntax is

```
&SNLCINP
   MAGCOR_FILE     = 'magCor.dat'   ! data only (not for sims)
   SIM_MAGCOR_FILE = 'magCor.dat'   ! force on sims (not data)
```

```
more magCor.dat
NVAR: 5
VARNAMES:     ROW           DUM1  MAGCOR  DUM2    DUM3
ROW:  1346137-56542.271-g    23   0.0123  10.23  22.456
ROW:  1346137-56542.272-r    33   0.0094  10.12  24.679
ROW:  1346137-56542.274-i    67  -0.0025  11.98  21.023
etc ...
```

The SNID, MJD and BAND are glued together to form a unique ROW-string identifying the correction. Only the MAGCOR column is used, and other optional columns (DUM1,DUM2,DUM3) are ignored. Each FLUXCAL is internally multipled by $10^{(-0.4 \cdot \mathrm{MAGCOR})}$.

For public data releases, such corrections may be included in the data files. To avoid accidentally applying these corrections twice, the MAGCOR_FILE should include the header key

```
VERSION_ADD:  <VERSION>
```

for the data version that has the corrections included. The SNANA analysis codes will abort if MAGCOR_FILE is specified for a data version that already includes these mag-corrections. If mag-corrections have been applied to data, the corrections can be removed with

```
&SNLCINP
   MAGCOR_FILE = '-magCor.dat'
```

where the minus sign in front is a command to subtract. This subtract option works only with the VERSION_ADD key.

5.28.5 Create New Data Files with OVERRIDE Updates

While the data override features in § 5.28.1 and § 5.28.4 are useful for quick testing, it is not always practical to analyze data with multiple sets of data-override files, and public data releases should never use this feature. To update data files with the override information, use the OVERRIDE feature(s) above and also use the REFORMAT feature in §12.4.2. Although reformatting is typically for translating TEXT to FITS (and vice-versa), it preserves the override values and it works for TEXT→TEXT and FITS→FITS.

5.29 Peculiar Velocity Corrections

Although SNANA has no explicit code to compute peculiar-velocity corrections (v_{pec}), such corrections can be made with external codes and be included in the data files with VPEC and VPEC_ERR keys. The v_{pec} values can also be read from an external file as described in §5.28.1. The light curve fitting program (snlc_fit.exe) reads v_{pec} and computes a corrected CMB redshift,

```
zHD = (1.0+ZCMB) / (1.0+VPEC/c) - 1.0
```

which is written to the output tables and used by the SNANA program SALT2mu.exe. A summary of the three redshift variables in the output table from snlc_fit.exe are:

- zHEL: heliocentric redshift
- zCMB: CMB-frame redshift, corrected from zHEL using these hard-wired parameters: CMBapex_l=264.031, CMBapex_b=48.253, CMBapex_v=369.82.
- zHD: zCMB with v_{pec} correction; use this quantity for Hubble diagram.

5.30 SIMCHI2_CHEAT

For simulations, the χ^2 is automatically computed in which the fitted parameters are replaced by the exact simulated paarameter values. The resulting χ^2 variable is called SIMCHI2_CHEAT and it is stored in the FITRES table output.

There is no ambguity for fits with fixed redshifts. For photo-z fits, however, the defined filter bands based on the fitted photo-z may be incorrect for the true redshift. For example, a fitted photo-z of 0.4 would include the z-band filter for the SALT2 model, but if the true redshift is 0.1 then trying to evaluate the z-band model-mag would result in an abort because it is too red for the SALT2 model in the rest-frame. When undefined filters are detected, SIMCHI2_CHEAT is set to $-$NFILT_UNDEFINED and the χ^2 calculation is skipped to avoid the abort.

5.31 Cuts on true SIM_XXX Parameters

To apply cuts on generated (true) parameters,

```
&SNLCINP
   SIMVAR_CUTWIN_STRING = 'SIM_c -0.1 0.1 SIM_REDSHIFT_CMB .1 .2'
   !  and/or
   SIM_TEMPLATE_INDEX_LIST = 202,203
   ! and/or
   SIMVAR_CUTWIN_STRING = 'NEP_SIM_MAGOBS 3 9999' ! >=3 obs with trueFlux>0
   ! and/or
   CUTWIN_TREST_TRUEFLUX = -20, 100  ! require trueFlux>0 in this range
```

The first option requires the true SALT2-color (for SALT2 model only) and true redshift (for any model) to satisfy the above cut-windows. Also works on SIM_x1. The second option is for NON1ASED, SIMSED, and LCLIB models; it selects a user-defined list of template indices.

The 3rd option requires at least 3 observations with true flux>0. The last option (CUTWIN_TREST_TRUEFLUX requires a true positive flux in the specified TREST range (days). For fakes overlaid on image, these last two options are useful to avoid pathologies from missing fake overlays that leave unphsyical zero-flux observations.

5.32 IDEAL Fits with True Flux (Sim only)

To determine the intrinsic scatter matrix from simulations ($\Sigma_{\rm int}$), this section describes how to perform "IDEAL" light curve fits that use 1) true fluxes instead of the reported fluxes with Poisson fluctuations, 2) true Galactic extinction value for MWEBV, and 3) fixes peak-MJD to its true value. The flux uncertainties are not modified with the options presented here. There are two methods for running IDEAL fits:

```
&SNLCINP                      or     &FITINP
  USESIM_TRUEFLUX = T                   SIMFIT_IDEAL_PRESCALE = 4
```

The first option (`USESIM_TRUEFLUX`) prepares IDEAL fits and works with `psnid, snlc_fit` or any user analysis. The second option (`PRESCALE`) works only with `snlc_fit`, but it performs both the IDEAL and nominal fit in the same job, thus simplifying higher-level pipelines. For example, fitting with the SALT2 model results in the following fit parameters store in the same `FITRES` table (TEXT,HBOOK,ROOT):

```
PKMJD c x1 x0 mB                                    # Nominal
PKMJD_IDEAL  c_IDEAL x1_IDEAL x0_IDEAL mB_IDEAL     # IDEAL
```

The difference between IDEAL-fitted parameters and the true generated parameters can be used to compute $\Sigma_{\rm int}$. For example, the mB-c element of $\Sigma_{\rm int}$ is compute with `FITRES` table elements as follows:

$$\Sigma_{\rm int}(mB, c) = N^{-1} \sum [(\text{mB\_IDEAL} - \text{SIM\_mB})(\text{c\_IDEAL} - \text{SIM\_c})]$$

Since fitting every event twice doubles the CPU time, this option is enabled with an explicit pre-scale to allow reducing the CPU time if a smaller sample of IDEAL fits is adequate. In the example above, 1/4 of the events are fit twice (IDEAL plus nominal), thus increasing the CPU time by 25%. Events which do not have an IDEAL fit have their IDEAL parameter values set to 999.

6 Private Options

6.1 Creating Your Private Code

Here we describe how to create a private code version for the snana program (`snana.exe`) or the light curve fiting program (`snlc_fit.exe`). This feature is useful to write your own analysis package without the overhead of reading data files and applying selection cuts, and also to make small modification to the existing analysis codes. Modifications include writing out specific information to a file, calculating a quantity that is not available, or testing variations on algorithms in the public code. This feature is available for the fortran analysis codes, but is not available for the simulation.

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.F90` or `snlc_fit.F90`, 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 (`cvs co`), 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.

6.2 Private Sim Path: PATH_SNDATA_SIM

By default, the simulation creates a "GENVERSION" sub-directory under
PATH_SNDATA_SIM = $SNDATA_ROOT/SIM.
The simulated data files can be re-routed to a user-defined path with sim-input key

```
PATH_SNDATA_SIM:  <path>
```

This key works in the sim-input file with interactive snlc_sim.exe. To submit batch jobs with the submit_batch_jobs.sh script (§12.2.1), this key must be in the sim-master-input file. This output re-route feature is *not* recommended, except for special reasons such as insufficient disk space under $SNDATA_ROOT.

The simulation automatically stores use user-defined path names in

```
$SNDATA_ROOT/SIM/PATH_SNDATA_SIM.LIST
```

so that analysis programs (snana.exe, snlc_sim.exe, psnid.exe) know where to check for simulated samples without user input. If a simulation needs to be moved out of $SNDATA_ROOT/SIM, the file above (PATH_SNDATA_SIM.LIST) can be manually edited.

6.3 Private Data Path for Analysis: 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 $SNDATA_ROOT/lcmerge. SNANA and fitter jobs can read data (not simulations) 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 $SNDATA_ROOT/lcmerge. Users are cautioned to use this feature only for testing, and not as long-term data storage for your analysis. For simulations, PRIVATE_DATA_PATH is ignored.

6.4 Private Model-Path: $SNANA_MODELPATH

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

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

While these public directories are intended for stable models, a private "`setenv SNANA_MODELPATH`" 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_MODELPATH/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 (§12.4.2). Using a private fitter option (§6.1), 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)`'.

6.5.1 CUTWIN-selection on Private Variables

Arbitrary cuts on private variables can be applied. Using the private variable example above, cuts are defined using the `&SNLCINP` namelist string

```
PRIVATE_CUTWIN_STRING = 'HOST_PROPERTY 20.4  100.',  'PHOTOFLAG 1 600'
  or
PRIVATE_CUTWIN_STRING = 'AGN_SCAN  !=  2'   ! veto cut
```

where the first example requires values within the specified ranges, and the second example shows how to reject (veto) based on a value.

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 $SNDATA_ROOT/SURVEY.DEF. Check if your survey and/or telescope is already defined before making changes.

2. Add new filters in $SNDATA_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,[26] 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 $SNDATA_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 $SNDATA_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 $SNDATA_ROOT/lcmerge ; ls *.README". 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 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. 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.

[26] Sample kcor-input files are in $SNDATA_ROOT/sample_input_files/kcor.

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    Bessell90_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.

7.2 Combining Surveys

For SNIa-cosmology analyses, there are many low-z samples consisting of different surveys, and different filter systems within a survey. Because of duplicate filter names (e.g., UBVR appear in multiple samples), SNANA requires fitting each low-z sample separately, which can complicate analysis pipelines. In addition, low-z samples do not include meta-data (SKY,PSF,ZP) needed create SIMLIB files for simulations (§4.7). To simplify analysis, and produce a SIMLIB file for simulations, data samples can be combined with the following example based on hypothetical low-z samples:

```
combine_dataVersions.py  myVersions.input

more myVersions.dat
VERSION:  LOWZ_SET1   LOWZ/kcor_LOWZ_SET1.input   # UBVRI
VERSION:  LOWZ_SET2   LOWZ/kcor_LOWZ_SET2.input   # UBVri
VERSION:  LOWZ_SET3   LOWZ/kcor_LOWZ_SET3.input   # uBVgri
SURVEY_OUT:  LOWZ_COMBINED
```

The input file contains a list of data versions, and associated kcor-input file (§3) for each data version. Note that kcor-INPUT files are specified, not the fits files. The filters are listed in the comment field, but not required since the filters are read from the kcor-input files. Each kcor-input file is checked locally, and under $SNDATA_ROOT/kcor. The script 'combine_dataVersions.py' performs the following tasks:

- merge all kcor-input files into a combined kcor-input file. Each original filter is mapped alphabetically to "abcde...xyzABCDE...XYZ01..89." For the example data versions above, the filter remapping is

  ```
  LOWZ_SET1:    UBVRI  -> abcde
  LOWZ_SET2:    UBVri  -> fghij
  LOWZ_SET3:    uBVgri -> klmnop
  ```

  These 3 samples include 16 unique filter bands, which are mapped to 16 unique characters: "abcdefghijklmnop." To help preserve the original filter name, the auto-generated combined kcor-input file defines filters as follows,

  ```
  FILTER:  LOWZ_SET1-U/a   LOWZ_SET1_U.dat  9.69
  FILTER:  LOWZ_SET1-B/b   LOWZ_SET1_B.dat  9.82
  FILTER:  LOWZ_SET1-V/c   LOWZ_SET1_V.dat  9.77
  FILTER:  LOWZ_SET1-R/d   LOWZ_SET1_R.dat  9.44
  FILTER:  LOWZ_SET1-I/e   LOWZ_SET1_I.dat  9.23
  ```

  Thus we can still see the original filter name before the slash; SNANA only needs the last character in the filter string, which is a,b,c,d,e for the above FILTER keys.

- produce combined data set with filter names replaced as indicated in the kcor-input files. The combined survey name is specified by the SURVEY_OUT key in the myVersions.dat file above. Input data versions must be in TEXT format.

- create SIMLIB file from data. For samples without meta-data (SKY,PSF,ZP), such as low-z, some assumptions are needed. First, the PSF is fixed to 1″ FWHM. Second, SKYMAG vs. wavelength is taken from an old LSST-DEEP simulation. For space-based surveys (HST,JWST,WFIRST), the assumptions are modified: PSF is 0.2″, and SKYMAG vs. wavelength is from WFIRST simulation. The zeropoint (ZP) for each observation is computed in order to match the measured S/N in the data.

To run light-curve fitting on the combined version, use the auto-generated kcor file and specify all bands in the &FITINP name-list with

```
FILTLIST_FIT  = 'abcdefghijklmnop'
```

Similarly, run the simulation using the same kcor file and sim-input key

```
GENFILTERS:   abcdefghijklmnop
```

The SALT2-fitted parameters to not depend on how the filters are named and neither does the cosmology fitting step. The main inconvenience with the remapped bands is defining the spectroscopic selection function for the simulation. For example, defining $\epsilon_{\rm spec}$ as a function of the original B-band is really a function of the newly-defined b, g, l- bands in the combined data file. There are a few SNANA featurs to overcome this inconvenience:

1. For simulations, the $\epsilon_{\rm spec}$ map can be defined as a logical-or of bands,

   ```
   NVAR: 2  # obsolete key starting v10_70
   VARNAMES:  b+g+l  SPECEFF    # peakMag for B band
   SPECEFF:   12.10   1.0000
   SPECEFF:   12.30   1.0000
   SPECEFF:   12.50   1.0000
   SPECEFF:   12.70   0.998
   etc ...
   ```

   which is equivalent to defining 3 identical maps, each with a different band.

2. for the output SNANA and FITRES tables, the filter band names can be re-mapped back to their original names using the following &SNLCINP namelist input:

   ```
   SNTABLE_FILTER_REMAP =
     'afk->U  bgl->B  chm->V  n->g   di->r  ej->i'
   ```

   This feature has no effect on fitting, as the fit still uses all 16 distinct bands. However, the output tables are only a function of the re-mapped bands "UBVgri". Thus instead of output peak-mag table columns 'm0obs_b, m0obs_g, m0obs_l,' the REMAP feature results in just one column, 'm0obs_B,' to replace the original 3 bands.

3. While the combined data version has a new name (e.g., 'LOWZ_COMBINED'), each original survey name is stored as a SUBSURVEY_NAME. IDSURVEY in the output tables is based on the sub-survey (LOWZ_SET1, LOWZ_SET2, LOWZ_SET3), allowing for studies to correlate with each input sample.

In principle, this script can be used to combine all data samples (e.g., low-z + SDSS + SNLS + PS1 + DES + ...). The kcor-file and light-curve fitting would be fine. However, the auto-generated SIMLIB file is based on rough guesses for missing meta-data. For surveys that provide the meta-data, it is better to use a proper SIMLIB with random sky locations.

8 Photometric Classification

8.1 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.F90. 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 §4.40. The current psnid.exe method is based on [29], and is called "Bayesian Evidence with Supernova Templates" (BEST); other methods can be added in psnid.exe using either C or fortran functions. Example namelist files are in

```
$SNDATA_ROOT/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'
   FILTLIST_PEAKMAG_STORE = 'ri'     ! store peakmag for each model

   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.F90) to control the detailed behavior of the classification algorithm.

The multi-CPU distribution script (submit_batch_jobs.sh ; see §12.2.1) can be used in the same manner as for snlc_fit.exe by specifying an additional key

```
JOBNAME:  psnid.exe
```

at the top of the namelist file.

8.1.1 Preparing Photometric Templates for psnid.exe

SN templates for both Ia and NONIASED are prepared with standard sim-input files. The script

```
$SNANA_DIR/util/sim_SNgrid.pl
```

is convenient for processing multiple sim-jobs and organizing the output. See top of script for usage instructions.

The simulation input file (for snlc_sim.exe) is the same as for a normal simulation job, but with the following additional keys:

```
GENSOURCE:       GRID   # make sure to remove RANDOM GENSOURCE
NGRID_LOGZ:        50   # logarithmic redshift bins
NGRID_SHAPEPAR:    10   # Delta or x1 or dm15 ...
NGRID_COLORPAR:     2   # AV  or color
NGRID_COLORLAW:     1   # RV  or Beta
NGRID_TREST:       56   # GENRANGE_TREST
GRID_FORMAT:     FITS   # TEXT or FITS
```

The standard GENRANGE_XXX keys give the range for each parameter.

8.1.2 Redshift Priors for psnid.exe

There are three choices for a redshift prior in psnid using the &PSNIDINP namelist parameter OPT_ZPRIOR:

OPT_ZPRIOR Value	prior	data header keyname
0	flat/default	---
1	best/spectro	REDSHIFT_FINAL
2	host photo-z	HOSTGAL_PHOTOZ

where the above table shows which redshift key from the data header is used. Note that REDSHIFT_FINAL should correspond to the best available redshift, usually a spectroscopic redshift of either the SN or the host. However, if only a host photo-z is available then REDSHIFT_FINAL = HOSTGAL_PHOTOZ and OPT_ZPRIOR values of 1 and 2 will give the same result. For OPT_ZPRIOR=1 you can select the redshift precision with namelist variable "CUTWIN_ZERR= zmin,zmax."

The photo-z result implicitly assumes that the redshift prior information is independent of the SN light curve. Thus beware when setting REDSHIFT_FINAL to the SN photo-z; in this case setting OPT_ZPRIOR=1 results in a photo-z that is strongly correlated with the input [SN] redshift, and hence the photo-z error is likely to be underestimated.

8.1.3 Rate Priors for `psnid.exe`

By default there is a flat rate prior such that the evidence is $\exp^{-\chi_i^2/2}$, where χ_i^2 is the best-fit chi-squared for the i'th template (SNIa or SNCC). A rate prior can be used to compute the evidence as $W_i \times \exp^{-\chi_i^2/2}$, where W_i is the relative rate. The rate prior options can be defined in the `&PSNIDINP` namelist as

```
OPT_RATEPRIOR = 1  ! default is 0 (OFF)
ZRATEPRIOR_SNIA  = 2.6E-5, 2.2, 1.0  ! alpha, beta, Zmax (default)
ZRATEPRIOR_NONIA = 6.8E-5, 3.6, 2.0  ! alpha, beta, Zmax (default)
```

where `ZRATEPRIOR_XXX` define the redshift-dependent rate model as

$$R(z) = \alpha(1 + z')^{\beta} \tag{36}$$

with $z' = z$ for $z < Z_{\rm max}$ and $z' = Z_{\rm max}$ for $z > Z_{\rm max}$. The $Z_{\rm max}$ option allows giving a flat rate at high redshift where the rate-uncertainty is large. In general one needs to specify only `OPT_RATEPRIOR=1` since the default `ZRATEPRIOR_XXX` parameters are reasonable.

For SNIa, $W_i = R(z)$. For SNCC we have $W_i = R(z) \times f_i$, where f_i is the NONIA fraction (i.e., weight) defined in the sim-input file used to generate the GRID. The f_i are internally renormalized so that $\sum_i f_i = 1$.

9 Light Curve Models

Here we discuss the available light curve models *classes* for the simulation and light-curve fitting program. This is brief a technical discussion on how to use the models in SNANA; a reference for each model is provided for more details. While each model class described below can be used for simulations, only the semi-analytic SNIa models can be used for light-curve fitting with the snlc_fit.exe program; these fitting models (MLCS2k2, SALT-II, SNooPy) include &FITINP namelist information in addition to sim-input keys. Simulated output data from any model class can be fit with the snlc_fit.exe program.

GENMODEL specification is based on a directory path (class=SALT-II, BAYESN, SIMSED, NONIASED, BYOSED[27]) or a file (class=NONIAGRID, LCLIB). Examples are

```
# models based on directory: model class name is between slash and dot
GENMODEL:  [PATH]/SALT2.JLA-B14
GENMODEL:  [PATH]/BAYESN.T21
GENMODEL:  [PATH]/SIMSED.SNII-MOSFIT
GENMODEL:  [PATH]/NONIASED.UV2IR

GENMODEL: BYOSED [AnyPathName]   # option without PATH name restriction

# models based on file; no restriction on PATH or fileName
GENMODEL:  LCLIB      [PATH]/fileName
GENMODEL:  NONIAGRID  [PATH]/fileName
GENMODEL:  SIMLIB     # use MAG column in SIMLIB_FILE argument
```

An environement variable can be used as part of the[PATH]; e.g.,

```
GENMODEL: $DES_ROOT/models/SIMSED.TEST
```

Note that the model class must be specified between the last slash and a dot so that both class the path are parsed from a single argument. The BYOSED class has an option to remove path-name restriction as shown above.

Many model parameters are described by Gaussian or asymmetric Gaussian distributions. For parameter "XXX" the following input keys specify the distribution:

```
GENPEAK_XXX:  <PEAK>     (or legacy key GENMEAN_XXX)
GENRANGE_XXX: <MIN> <MAX>
GENSIGMA_XXX: <SIGNEG> <SIGPOS>
GENSKEW_XXX:  <SKEW>
```

The GENRANGE values truncate the distribution. The GENSIGMA key has two values to specify an asymmetric (or symmetric) Gaussian distribution. GENSKEW is used to define a value-dependent sigma, $\sigma \to \sigma \times |\text{skew} \times (x - \text{Peak})|$, where the sign of GENSKEW determine which side of the peak to add the longer tail. GENSKEW typically results in a larger tail compared with an asymmetric Gaussian.

[27] https://byosed.readthedocs.io/en/latest

9.1 Host Galaxy Extinction

The following host-galaxy extinction parameters can be included with any of the SN models below. R_V and A_V specifications are required for BayeSN, `MLCS2k2`and `SNooPy`, and optional for the other models. The color law (e.g., CCM89, Fitzpatrick99) is the same as that chosen for Galactic extinction (see `OPT_MWCOLORLAW` in §4.21).
Simulation input keys for `snlc_sim.exe` :

```
GENPEAK_RV:       xxx        # peak prob dust parameter
GENRANGE_RV:      xxx  xxx   # min and max limits for RV generation
GENSIGMA_RV:      xxx  xxx   # lo & hi Gaussian sigmas

GENRANGE_AV:      xxx  xxx     # CCM89 V-band extinction
GENTAU_AV:        xxx          # dN/dAV = exp(-AV/xxx)
GENSIG_AV:        xxx          #      += Guass(AV,sigma)
GENRATIO_AV0:     xxx          # Gauss/exp ratio at AV=0

# instead of selecting AV from distribution, here E(B-V)
# is randomly picked so that AV = RV*E(B-V)
GENRANGE_EBV_HOST: xxx xxx
GENTAU_EBV_HOST:   xxx
GENSIG_EBV_HOST:   xxx
GENRATIO_EBV_HOST: xxx
```

9.2 MLCS2k2

Reference: Jha, Riess, Kirshner, **AJ 659**, 122 (2007). [1]

Note that host-galaxy extinction parameters (for A_V,R_V) are required from §9.1.
Simulation input keys for `snlc_sim.exe` :

```
GENPEAK_DELTA:    xxx         # shape  parameter
GENRANGE_DELTA:   xxx  xxx
GENSIGMA_DELTA:   xxx  xxx
```

&FITINP namelist variables for `snlc_fit.exe` :

```
OPT_SNXT       = 1   ! Use CCM89 + ODonnel94 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_SHAPE   = xxx  ! 0 => fix DELTA to INIVAL_SHAPE; else float
INIVAL_SHAPE   = xxx

PRIOR_DELTA_PROFILE = xxx, xxx, xxx, xxx  ! grep snlc_fit.F90 for details
```

9.3 SALT-II

Reference: J. Guy et al., **A&A 466**, 11 (2007). [2]
Simulation input keys for `snlc_sim.exe` :

```
GENPEAK_SALT2x1:   xxx         ! peak PDF for shape parameter
GENSIGMA_SALT2x1:  xxx  xxx    ! sigma-lo and sigma-hi
GENRANGE_SALT2x1:  xxx  xxx    ! generation range

GENPROB2_SALT2x1:   xxx         ! optional PROB(2nd peak)/All
GENPEAK2_SALT2x1:   xxx         ! optional 2nd peak PDF
GENSIGMA2_SALT2x1:  xxx  xxx    ! optional sigmas

GENPEAK_SALT2c:    xxx          ! peak PDF for color parameter
GENRANGE_SALT2c:   xxx  xxx
GENSIGMA_SALT2c:   xxx  xxx

# mag = mB* + alpha*x1 - beta*color
GENPEAK_SALT2BETA:  3.2   ! color coeff
GENSIGMA_SALT2BETA: 0 0   ! default is no beta smearing
GENRANGE_SALT2BETA: 0 5   ! restrict if SIGMA is non-zero
SALT2BETA_cPOLY: b0,b1,b2 ! polyFun of c; overrides GENPEAK_SALT2BETA

GENPEAK_SALT2ALPHA:  0.14    ! x1 coeff
GENSIGMA_SALT2ALPHA: 0 0.0   ! default is no alpha smearing
GENRANGE_SALT2ALPHA: 0 0.3   ! restrict if SIGMA is non-zero
```

and special grid keys for BBC are described in §10. Note that host-galaxy extinction parameters (A_V,R_V) are optional from §9.1.

To avoid transcription errors, `x1` and `c` populations can be selected from a list of asymmetric-Gaussian models as follows,

```
GENPOP_ASYMGAUSS: $SNDATA_ROOT/models/SALT2/POPULATIONS.DAT SK16_G10_HIZ
```

and `SK16_G10_HIZ` can be replace with other models in this file. For testing, a private `POPULATIONS.DAT` file can be used. For population models that depend on other variables (e.g., host mass), or cannot be described by an asymmetric Gaussian, use the more general `GENPDF_FILE` option (§ 4.33).

&FITINP namelist variables for snlc_fit.exe :

```
INISTP_COLOR   = xxx  ! 0 => fix color to INIVAL_COLOR; else float
INIVAL_COLOR   = xxx
INISTP_SHAPE   = xxx  ! 0 => fix x1 to INIVAL_SHAPE; else float
INIVAL_SHAPE   = xxx

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

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

SALT2alpha     = xxx  ! to compute approx mu-resid; does not affect fit
SALT2beta      = xxx  ! idem
```

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.

9.3.1 Simulating α, β, γ Grid for SALT2mu/BBC

For SALT2mu-BBC, the "biasCor" sample is generated on a grid of α, β, γ in order to dynamically interpolate the bias-correction at each fitting step. The sim-input keys are as follows:

```
BIASCOR_SALT2ALPHA_GRID:  0.10  0.20
BIASCOR_SALT2BETA_GRID:   2.50  3.50
BIASCOR_SALT2GAMMA_GRID:  0.0   0.10

# or ... below is legacy method requiring 4 keys per variable
GENPEAK_SALT2ALPHA:   0.14
GENSIGMA_SALT2ALPHA:  1E8   1E8
GENRANGE_SALT2ALPHA:  0.10  0.20
GENGRID_SALT2ALPHA:   2       # <=== only 2 allowed for BBC

GENPEAK_SALT2BETA:   3.2
GENSIGMA_SALT2BETA:  1E8  1E8
GENRANGE_SALT2BETA:  2.5  3.5
GENGRID_SALT2BETA:   2       # <== only 2 allowed for BBC
```

BIASCOR_SALT2ALPHA_GRID and BIASCOR_SALT2BETA_GRID specify 2×2 grid for BBC interpolation. If γ were implemented the same way as for α and β, BBC could only fit a step function with a $M_{\rm host}$-split at exactly the same value used in the biasCor sample. To allow for an arbitrary standardation function of $M_{\rm host}$ in the BBC fit, the γ parameter is treated in a different manner compared with α and β. BIASCOR_SALT2GAMMA_GRID specifies two grid-points to randomly generate mag-offsets. This is not quite γ because it has no $M_{\rm host}$ dependence, but in the BBC fit these mag offsets are associated with γ and $M_{\rm host}$, where γ is the maximum SNIa mag-splitting.

9.3.2 Propagating Calibration Systematics from SALT2 Training

The batch-submit script (`submit_batch_jobs.sh`) includes `TRAINOPT` options to apply calibration systematics for the SALT2 training process; these calibration shifts (primary mag and filter wavelength) are recorded in the `SALT2.INFO` file as follows:

```
MAGSHIFT:  SDSS g  0.01
MAGSHIFT:  SDSS z -0.01
MAGSHIFT:  CFA2 B  0.01
WAVESHIFT: SDSS r  10   # Angstroms
```

Starting with `SNANA` version `v11_02b`, the `SNANA` simulation (`snlc_sim.exe`) and light curve fit program (`snlc_fit.exe`) read and propagate these shifts. This feature captures the coherent effect of calibation errors in data samples used in both training and cosmology. The `MAGSHIFT` and `WAVESHIFT` keys are enabled by default; to disable these keys,

```
# fit-input file
 &FITINP
    ENABLE_MAGSHIFT_SALT2  = F  ! disable MAGSHIFT
    ENABLE_WAVESHIFT_SALT2 = F  ! disable WAVESHIFT
    ALLOW_NEGFLUX_SALT2    = F  ! if Flam<0; Flam=0 (test only)

# sim-input file
   GENMODEL_OPTMSK:  4  ! disable MAGSHIFT
   GENMODEL_OPTMSK:  8  ! disable WAVESHIFT
   GENMODEL_OPTMSK: 12  ! disable both
   GENMODEL_OPTMSK: 16  ! if Flam<0; Flam=0 (test only)
```

9.4 BayeSN

References: K. S. Mandel et al., **MNRAS 510**, 3939 (2022); S. Thorp et al., **MNRAS 508**, 4310 (2021); M. Grayling et al., **MNRAS 531**, 953 (2024). [3, 30, 31].

The BayeSN SED model is based on the Hsiao template [32], modified by a t- and λ-dependent warping surface $\propto 10^{-0.4W(t,\lambda)}$, and extinguished by dust. Extrapolation of a BayeSN model outside its defined phase and wavelength range is achieved by extrapolating $W(t, \lambda)$.

Note that host-galaxy extinction parameters (for A_V,R_V) are required from §9.1. The form of the host-galaxy dust extinction curve will be inherited from the Milky-Way color law. Only R_V-dependent extinction curves are currently supported (§4.21.2).
Simulation input keys for for `snlc_sim.exe` :

```
GENMEAN_THETA:    xxx        # shape parameter
GENRANGE_THETA:   xxx  xxx
GENSIGMA_THETA:   xxx  xxx

GENMODEL_MSKOPT:  <mask>     # additional settings
```

The additional settings that can be enabled via setting bits in `GENMODEL_MSKOPT` are as follows:

- `GENMODEL_MSKOPT+=1`: Enable BayeSN's default (recommended) residual scatter model. Includes a gray component, $\delta_M \sim N(0, \sigma_0^2)$, and a time- and wavelength-dependent component, $\epsilon \sim N(\mathbf{0}, \mathbf{\Sigma}_\epsilon)$. The (co)variance of these components is set by the YAML file defining the version of the BayeSN `GENMODEL` specified in the sim input.
- `GENMODEL_MSKOPT+=2`: Enable the non-gray (ϵ) component of the BayeSN residual scatter model.
- `GENMODEL_MSKOPT+=4`: Enable the gray (δ_M) component of the BayeSN residual scatter model.
- `GENMODEL_MSKOPT+=32`: Enable extrapolation beyond the BayeSN phase range by setting $W(t, \lambda) \to 0$. This reverts to the Hsiao template.
- `GENMODEL_MSKOPT+=64`: Enable extrapolation beyond the BayeSN phase range by setting $W(t, \lambda) \to W(t_{\rm edge}, \lambda)$. This gives Hsiao-like behavior, but re-normalized to minimize any jumps in magnitude.
- `GENMODEL_MSKOPT+=128`: Enable a "verbose" mode for debugging purposes.
- `GENMODEL_MSKOPT+=256`: Enable a "super verbose" mode for debugging splines.
- `GENMODEL_MSKOPT+=512`: Enable extrapolation beyond the BayeSN wavelength range by setting $W(t, \lambda) \to 0$. This reverts to the Hsiao template.
- `GENMODEL_MSKOPT+=1024`: Enable extrapolation beyond the BayeSN wavelength range by setting $W(t, \lambda) \to W(t, \lambda_{\rm edge})$. This reverts to the Hsiao template, re-scaled to match the normalization of the BayeSN model at the edge of its defined wavelength range.
- `GENMODEL_MSKOPT+=2048`: Generic development mask for testing new code.

Multiple bits can be combined, when this is unambiguous. For example, `GENMODEL_MSKOPT=6` would enable both components of the residual scatter model (equivalent to `GENMODEL_MSKOPT=1`). An ambiguous combination of bits will cause the program to abort.

Please note that extrapolation in wavelength, or by > 10 d in phase, is not recommended in most circumstances. Without setting any of the extrapolation-related `GENMODEL_MSKOPT` bits, the model will *not* be extrapolated in wavelength, and will be extended in phase by linearly extrapolating $W(t, \lambda)$. This phase-dependent extrapolation can lead to surprising results (e.g. rebrightening at late times) if pushed too far! If extensive extrapolation of the model is needed, setting `GENMODEL_MSKOPT=576` is probably the best option, but this is experimental and should be used with caution! It is currently not possible to extrapolate beyond the phase ($t \in [-20, 85]$ d) and wavelength ($\lambda \in [1000, 25000]$ Å) ranges of the Hsiao [32] template.

9.5 SNooPy

Reference: C. Burns et al., **AJ 141**, 19 (2011). [4]

Note that host-galaxy extinction parameters (for A_V,R_V) are required from §9.1.
Simulation input keys for `snlc_sim.exe` :

```
GENPEAK_DM15:    xxx         # shape parameter
GENRANGE_DM15:   xxx  xxx
GENSIGMA_DM15:   xxx  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_SHAPE_RANGE = 0.7, 2.0 ! constrain DM15 in this range
PRIOR_SHAPE_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_SHAPE   = xxx  ! 0 => fix DELTA to INIVAL_SHAPE; else float
INIVAL_SHAPE   = 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, ΔM_{15}) using the GRID option from §4.40. 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.

9.6 SIMSED

A SIMSED model is a ensemble of SED time series, where the ensemble spans an arbitrary set of parameters characterizing the object. SIMSED models can be derived from specialized explosion-model codes (e.g., MOSFIT, Sedona, Pheonix ...), or determined from high-resolution spectroscopic data. The SIMSED model is implemented in the SNANA simulation, but is not included in a fitting code. However, simulated data using SIMSED can be fit with SNANA fitting codes (snlc_fit.exe, psnid.exe) the same way as for real data. SIMSED models tend to be much larger than SNDATA_ROOT and are thus available through specialized releases such as PLAsTiCC.[28] A few examples in the pulic SNDATA_ROOT are here,

```
$SNDATA_ROOT/models/SIMSED/SIMSED.SNANA_tester           # GCD explosion
$SNDATA_ROOT/models/SIMSED/SIMSED.KNovae_BarnesKasen2013 # theory KN
$SNDATA_ROOT/models/SIMSED/SIMSED.GW170817_AT2017gfo     # GW170817 KN
```

Each directory above includes an "SED.INFO" file listing each SED and the corresponding parameters. The only limit to the number of parameters (and SEDs) is the amount of memory on your computer. An example of an SED.INFO file is as follows:

```
# optional keys:
FLUX_SCALE:        1.000         # global flux scale (e.g., to fix units)
RESTLAMBDA_RANGE:  2500  15000   # valid range of <lamObs>/(1+z)
OPTMASK_TOSHIFT_PEAKMAG: 0       # do NOT time-shift PEAKMAG to DAY=0
MAG_ERR:                  0.1    # not used
MINSLOPE_EXTRAP_LATE:   0.05     # force min mag/day slope for late-time extrap
LOGZBIN: 0.05                    # force logz bins to reduce storage

# required keys
NPAR: 6
PARNAMES: SN_INDEX MNI COSANGLE MBSED DM15SED TEXPL
SED: GCD2D_Ni0.47_Pre80_1.SED  1  0.47  -0.967  -18.350  0.571  -21.628
SED: GCD2D_Ni0.47_Pre80_8.SED  2  0.47  -0.500  -18.413  0.574  -20.977
SED: GCD2D_Ni0.47_Pre80_1.SED  3  0.47  -0.033  -18.426  0.654  -21.398
SED: GCD2D_Ni0.47_Pre80_2.SED  4  0.47  +0.500  -18.400  0.767  -21.385
etc ...
```

Users prepare the SED.INFO file, along with each SED time-series file with format

```
epoch(days)   wavelength(A)   flux(erg/cm^2/s/A at 10pc) .
```

The wavelength binning must be uniform, but the epoch binning can be non-uniform. If the fluxes have incorrect units, the FLUX_SCALE key can be used to make a global unit conversion. For each SED, there is a default internal time-shift applied such that the bolometric flux is maximum at $T = 0$, and this rest-frame epoch corresponds to PEAKMJD in the data header.

In cases where the explosion time is well defined (e.g., triggered event from LIGO or ICECUBE), it may be useful to leave the original $T = 0$ definition in each SED file by specifying "OPTMASK_TOSHIFT_PEAKMAG: 0" in the SED.INFO file. With this option, beware that input

[28] https://zenodo.org/records/6672739

GENRANGE_PEAKMJD and output PEAKMJD correspond to the explosion time, not the time of maximum brightness.

MINSLOPE_EXTRAP_LATE forces extrapolated mags to get fainter with time; this feature is useful if late-time model fluctuations result in extrapolations that give brighter mags at later times.

The default redshift binning in log-space has LOGZBIN=0.02. For models consuming significant memory, LOGZBIN can be increased to reduce memory.

The first parameter with _INDEX in the name can be used to select random indices as shown below. The next two parameters (MNI and COSANGLE) are the physical parameters for generation; the remaining three *baggage* parameters (see below) depend on the first two and they are inertly passed along to the output tables, just as a small pebble in your food is passed through the digestive system.

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

```
SIMSED_PARAM:  MNI          # mass of Ni56
GENPEAK_MNI:  0.9           # mean of bifurcated Gaussian
GENRANGE_MNI: 0.47 1.26     # generation range
GENSIGMA_MNI: 0.4  0.25     # bifurcated Gaussian
   and/or
SIMSED_GRIDONLY:  COSANGLE        # cosine of viewing angle
GENPEAK_COSANGLE:   0
GENRANGE_COSANGLE: -0.96  0.96
GENSIGMA_COSANGLE:  1.E7  1.E7   # large sigmas => flat distribution

# optional covariances among parameters
SIMSED_REDCOR(MNI,COSANGLE):  -0.26  # optional reduced covar; -1 to +1
    or
SIMSED_COV(MNI,COSANGLE):  -1.34  # optional covar

# other GRIDONLY options=
SIMSED_GRIDONLY: SEQUENTIAL  # sequential generation of grid values
     or
SIMSED_GRIDONLY: SN_INDEX    # pick random SN_INDEX
    or
SIMSED_GRIDONLY: SN_INDEX(3,5,6,9) # pick random among these SN_INDEX values

SIMSED_USE_BINARY:    1  # create binarys if they don't exist
SIMSED_USE_BINARY:    2  # force creation of SED.BINARY
SIMSED_USE_BINARY:    4  # force creation of flux-integral binary table
SIMSED_USE_BINARY:    6  # force creation of both binarys
SIMSED_PATH_BINARY:   <path for flux-table binary file>
```

Specify only those parameters that are used in the generation. For the SIMSED_PARAM keyword (MNI), the simulation interpolates synthetic light curve magnitudes as a function of the explosion-model parameter MNI, resulting in a continuous distribution of the specified bifurcated Gaussian. For the SIMSED_GRIDONLY keyword (COSANGLE), only values at the grid nodes are generated, and the specified bi-Gaussian distribution is respected; each randomly chosen parameter is 'snapped'

to the nearest grid value. If bi-Gaussian is not specified, a flat COSANGLE distribution is generated. This GRIDONLY option may be useful in cases such as a flag that specifies a random ignition point, and therefore continuous interpolation makes no sense. 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.

A correlation between any pair of parameters can be introduced with either a reduced correlation key (SIMSED_REDCOR) or a covariance key (SIMSED_COV). Default correlatios are all zero, and correlations are allowed only for symmetric Gaussian profiles. If correlations between more than 2 parameters are specified, the full off-diagonal correlation must be specified, noting that GENSIGMA_XXX define the diagonal covariances. For example, correlations between 3 parameters require 3 correlation keys; correlations between 4 parameters require 6 correlation keys.

"SIMSED_GRIDONLY: SEQUENTIAL" steps through each grid-value sequentially, and there is no need to specify parameter names or their distributions. 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.

While "SIMSED_GRIDONLY: KN_INDEX" selects a random KN_INDEX among the entire SED set, "SIMSED_GRIDONLY: KN_INDEX(3,5,6,9)" selects a random KN_INDEX among the subset 3,5,6,9. The subset feature works only if the column name includes and index sub-string (INDEX or index or INDX), and if the values are integers. Attemping the subset feature on MNI, for example, results in an abort.

All SIMSED parameters ("baggage" and for generation) are automatically included in the output table(s) and in the SIMGEN_DUMP list (§4.38.2). An explicitly defined SIMSED parameter in the SIMGEN_DUMP list will cause the simulation to abort because of a redundant parameter.

9.6.1 Arbitrary Distribution using SIMSED Weights

If the desired distribution is more complex than bifurcated Gaussian or picking a random SED, there are two mechanisms to assign an arbitrary selection weight for each SED. The first method is to include a WGT column in the SED.INFO file. The second method is to define a separate weight map in the sim-input file using key

```
SIMSED_WGTMAP_FILE:  <wgtmap file>
```

The WGTMAP file has the same form as the HOSTLIB_WGTMAP_FILE in §4.23.4 and Fig. 12, except that there is no SNMAGSHIFT column for the SIMSED_WGTMAP. The weight map file works whether or not there is a WGT column in the SED.INFO file.

9.6.2 Binary SIMSED Files

The SIMSED initialization takes ∼1 sec per SED, and is somewhat time-consuming for 100's or 1000's of SEDs. About half the time is reading the SED text files, and the other half is doing the flux-integrals as a function of passband, redshift, Trest, and SED surface. To greatly speed up this slow initialization, two internal binary files are created the first time a new SIMSED model is simulated. The first simulation must be interactive; batch jobs abort to avoid multiple cores writing to the same binary file.

The first binary file contains the SEDs (SED.BINARY), and it is stored in the same model (MMM) directory as the SED text files, $SNDATA_ROOT/models/SIMSED/MMM. This SED.BINARY file is automatically used by any SNANA user who specifies the MMM version.

The second binary file is the filter-specific flux-integral table, and this survey-specific file is stored in the directory specified by SIMSED_PATH_BINARY. 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 occurs if the 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.

To simplify creation of binary files for multiple SIMSED models,

```
make_simsed_binaries.py -H  # get help on input config file
```

Beware that this script uses the force-create options.

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
```

Finally, host-galaxy extinction parameters (for AV,RV) are optional from §9.1.

9.6.3 How to generate Spectra and True SED with SIMSED

The SIMSED model class pre-computes flux-integrals for each passband, phase and redshift, and does not hold the SEDs in memory. The advantage is that all of the model fluxes can be stored in memory, enabling more flexible selection algorithms among the SED templates. The disadvantage is that spectra and true SEDs cannot be generated.

For generating spectra or true SEDs for a SIMSED class, a simple work-around is to transform a SIMSED model into a NONIASED model (9.7) using utility

```
convert_SIMSED_to_NON1ASED.py -h
```

which uses the same SEDs to define model class for both SIMSED and NONIASED, the latter of which can be used to generate spectra and true SEDs. Beware that the NONIASED model does not read the physical parameters and thus selection criteria is more limited compared to the original SIMSED model.

9.7 NONIASED

While the Type Ia light curve models are based on a parametric equation or photometric templates, the "NONIASED" models are based on smoothed light curves and spectral templates. A spectral template time-series corresponds to a particular (well-observed) SN where a composite spectrum is warped to match the observed photometry. In addition to non-Ia SNe, this feature can be used to simulate peculiar Ia, theoretical models, or any transient object.

The observer-frame "NONIASED" model assumes that each rest-frame SED has 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. The KCOR_FILE key is needed to read in the filters and primary reference. Optional host-galaxy extinction (from R_V and A_V) is computed at $\bar{\lambda}_{\rm obs}/(1+z)$, where $\bar{\lambda}_{\rm obs}$ is the central wavelength of each filter; this approximation is numerically accurate to about 0.01 mag. The distribution of R_V and A_V is controlled by the parameters described in §9.1 and §9.2.

Publicly available NONIASED templates are given in $SNDATA_ROOT/models/NON1ASED . Each NON1ASED.XYZ directory contains a NON1A.LIST file, and a sample SIMGEN_INCLUDE_NON1A.INPUT file that lists properties for each NON1ASED as described below. You can include this file as-is using the INPUT_FILE_INCLUDE key, or copy it locally and make modifications.

As explained below, the user selects NONIASED templates using the integer indices in the NON1A.LIST file. As more NONIASED templates become available, the NON1A.LIST files should be updated by the relevant experts. The NONIASED model is specified with the following sim-input keys,

```
GENMODEL: $SNDATA_ROOT/models/NON1ASED/NON1ASED.V19_CC+HostXT

#  or more generally for any path
GENMODEL: $PATH/NON1ASED.[XYZ]

NON1ASED_KEYS: 5 INDEX  WGT  MAGOFF  MAGSMEAR  SNTYPE
NON1ASED:         2     0.2   0.0     1.2        2
NON1ASED:         3     0.2  -0.2     0.8        2
NON1ASED:         4     0.4  -0.6     0.9,0.5    3
etc ...
```

The GENMODEL base directory name must be of the form NON1ASED.[XYZ] where XYZ is an arbitrary suffix. Following the GENMODEL specification, each SED template is assigned a weight (for relative rate), mag-offset (MAGOFF), and Gaussian mag-smear (MAGSMEAR). The legacy keys "NON1A" and "NON1A_KEYS" have the same meaning as NON1ASED and NON1ASED_KEYS," respectively. The SED index is specified by "SIM_TEMPLATE_INDEX" in the data files and fitres-tables.

To process all SEDs with the same weight, magoff and mag-smear, set the first index to zero as follows:

```
NON1ASED_KEYS: 5 INDEX  WGT  MAGOFF  MAGSMEAR  SNTYPE
NON1ASED:         0     1.0   0.0     0.5        2
```

which automatically loops over all SEDs in the NON1A.LIST file.

The NON1ASED_KEYS are used to specify additional parameters that depend on NONIASED type.

- **INDEX** is the index in the NON1A.LIST file.
- **WGT** are relative rates. Since the weights are re-normalized to sum to unity, the first 1/4 of the generated SNe use INDEX = 2, the second 1/4 use INDEX = 3, and the latter 1/2 use INDEX = 4.
- **MAGOFF** is a magnitude offset applied to all passbands, and enables adjusting the luminosity function to match data.
- **MAGSMEAR** is a Gaussian σ to generate coherent intinsic mag fluctuations (i.e, equivalent to GENMAG_SMEAR for SNIa). Each random mag shift is stored in the data file (SIM_MAGSMEAR_COH key) and in the SNTABLEs. Comma-separated values specify two sigmas for bifurcated Gaussian; MAGOFF is the distribution peak, not the mean.
- **SNTYPE** is a spec-confirmed typing-index that appears after the SNTYPE keyword in the SNDATA files (see §4.6 for SNTYPE details). In the above example, NONIASED indices 2 & 3 are both assigned SNTYPE=2; NONIASED index 4 is assigned SNTYPE=3.

Each INDEX is generated sequentially based on a pre-computed number of events per INDEX: *i.e., the* NONIASED *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 NONIASED types. While the NONIASED indices are processed sequentially, random SNIDs can be assigned as described in §4.37. With random SNIDs, any subset of the simulation is a truly random subset.

Since the SEDs are read into memory, this model class is compatiable with TAKE_SPECTRUM keys (§4.26).

9.7.1 Peculiar SNIa

Peculiar SNIa (e.g., 91bg, 91T, Iax) are simulated the same way as NON1A, with an important caveat: the rate-vs-redshift for peculiar Ia is different than for NON1A (core collapse). There are three input modifications for peculiar Ia:

1. In the sim-input file, the 'NON1ASED:' keys are replaced with 'PEC1ASED:' (or 'PEC1A:')
2. In the NON1A.LIST file, the 'NON1A:' keys are replaced with 'PEC1A:' keys.
3. In the sim-input file, the rate-vs-redshift model for peculiar-Ia is defined with the 'DNDZ_PEC1A:' key. The arguments following this key have the same meaning as the arguments following the 'DNDZ:' key (§4.24) which defines the NON1A rate model.

9.7.2 Force Single NON1ASED Index

To select a specific NON1ASED index without modifying the sim-input file,

```
snlc_sim.exe <inputFile>  NON1A_INDEX_FORCE 4  NGENTOT_LC 3
```

which will generate 3 events using only NON1A index 4. The NON1A_INDEX_FORCE key is allowed only as command-line override; if this key is found in a sim-input file, the simulation will abort.

9.8 NON1AGRID

This is a model of pre-computed observer-frame magnitudes on a grid of redshift, rest-frame epoch, and template index. Compared to the NON1ASED model, the advantages of the NON1AGRID model are 1) the input grid can be created by non-SNANA codes using more sophisticated models, and 2) the NON1A index is randonly picked for each event, instead of the sequential selection for NON1ASED. The syntax for this model is

```
GENMODEL:  NON1AGRID  <GRIDFILENAME>
```

where GRIDFILENAME can reside either in your current working directory, include a full path, or reside under

```
$SNDATA_ROOT/models/NON1AGRID
```

An input GRID file can be made from the NON1ASED model as described in §4.40; simply remove the SNIa model keys and add the NON1A keys from §9.7. The MAGOFF values are included in the mag calculations, but MAGSMEAR is not. When using the NON1AGRID model, the MAGSMEAR values are applied with a Gaussian random number per event.

9.9 FIXMAG and RANMAG

These flat-lightcurve models are not intended to reflect an astrophysical transient, but are instead intended for debugging or generating fake magnitudes to overlay on images. Examples are as follows:

```
GENMODEL:  fixmag  20      # every obs mag=20
GENMODEL:  fixmag  20:24   # random obs mag between 20 & 24
GENMODEL:  ranmag  20:24   # same as above

GENMODEL:  FIXMAG  -19     # obs mag = -19 + MU
GENMODEL:  FIXMAG  -19:-17 # obs mag = ran(-19:-17) + MU
```

In the first case, all observer-frame magnitudes are 20. The 2nd and 3rd cases are equivalent, where the observer-frame mag for each event is randomly selected between 20 and 24. The same mag is used at all epochs, but a different random mag is chosen for each event. The range of fixed-mag epochs is determined by the sim-input key GENRANGE_TREST. For these observer-frame models, GENRANGE_REDSHIFT is ignored.

The upper case FIXMAG specifies rest-frame mag for which the distance modulus is added. For this option, GENRANGE_REDSHIFT must be specified.

9.10 SIMLIB: Read True Mags from SIMLIB File

This model option does not compute true magnitudes, but instead reads the true magnitudes from the last column of the SIMLIB file. An example use of this feature is to simulate mags of "fake" SN overlaid on images. The redshift, peakMJD, and GALID are automatically read from the SIMLIB headers, and the original GALID is used to model host noise. Sim-input file syntax:

```
GENMODEL:  SIMLIB     # use MAG column in SIMLIB_FILE argument
```

9.11 LCLIB: Galactic Transients

9.11.1 Overview

Galactic transient models include three general classes:

1. RECURRING-NONPERIODIC (e..g, AGN)
2. RECURRING-PERIDODIC (e.g., RRlyrae)
3. NON-RECURRING (e.g., micro-lens)

These models are simulated using a TEXT-file library and a list of template epochs specified in the sim-input file as

```
GENMODEL:   LCLIB   MYMODEL.TEXT   1+2+3+4
GENMODEL_MSKOPT:   <mask>
```

LCLIB (Light Curve LIBrary) is the generic model type in SNANA, MYMODEL.TEXT is the text-file library (Fig 17) and 1+2+3+4 is a list of relative epochs to average for the template magnitude. GENMODEL_MSKOPT enable options described in §9.11.4.

9.11.2 Defining the Library

The TEXT library begins with a global header specifying the name of the SURVEY, and a list of broadband FILTERS. Since there is no redshifting, these LCLIB models are defined as pre-computed light curves (broadband magnitudes vs. time), and not as an SED time series. The SURVEY & FILTERS in the header are compared against the user-request to prevent accidental mixing (e..g, cannot use an LSST library to simulate DES). The global header also includes a model name and list of model parameters names (MODEL_PARNAMES), for which floating-point (or integer) values are given for each LCLIB event. These model parameters are not used by the simulation, but they are written into the output data files to be used in offline analyses. A series of comment fields should be used to provide model references and to define the model parameters. An optional key NEVENT is used to re-use each event a total of NUSE=NGENTOT/NEVENT times in order to reduce read-time. Without the NEVENT key, the simulation would wrap around the library NUSE times, and waste time reading each event multiple times. Finally, the RECUR_CLASS key specifies which of the three Galactic transient classes.

After the global header there is a list of LCLIB "events," where each event includes an event header followed by a light curve in all filters. The event header contains the following information: 1) event ID (for debugging), 2) number of rows for the light curve, including T+S (NROW), 3) parameter values after PARVAL key, 4) optional coordinate, either RA&DEC or l&b, 5) optional angle-matching radius if event depends on Galactic coordinates.

Following the header, the light curve rows are defined with the keys "T:" (Template) and "S:" (Search). The LCLIB time steps can be non-uniform, and should be optimized to limit the library size. For example, the time step should be small during periods of rapid variation, and then increase during periods of slow change. Also note the recommended precision (%.3f for mags) to reduce file size.

For each of the three Galactic transient classes above, there are specific LCLIB features that must be included as described below:

1. **Recurring-NonPeriodic** events require explicit template (T:) epochs prior to the search (S:) epochs. The time-span of T-epochs should correspond to how templates are made in the survey, and the S-range must be at least as long as the survey. If the S-range is longer than the survey time, a random `LCLIB` start-time used. For example, consider a 3 year survey and a 5 year S-range: the simulation selects a random start-time within the first two years of the `LCLIB` event.

2. **Recurring-Periodic** events require search (S:) epochs covering exactly one cycle. Explicit template epochs are forbidden in the library because the simulation internally computes template epochs using the periodicity. The first and last S-epoch must have identical magnitudes, otherwise the simulation aborts.

3. **Non-Recurring** events require a single template (T:) epoch since all previous epochs must have the same quiescent magnitude. For the last S-epoch, the magnitudes should match the template epoch, indicating that the light curve has finished.

9.11.3 Implementation

Here are a few details about how the simulation interprets the library defined in §9.11.2. Note that "simulated event" (or generated event) refers to the light curve generated on the survey cadence, while "`LCLIB` event" refers to the library events illustrated in Fig. 17.

For each simulated light curve with duration $\Delta T_{\rm survey}$, the corresponding range of the `LCLIB` event is extracted and interpolation is used to compute the true mag at each simulated epoch. For recurring-nonPeriodic, the `LCLIB` event duration must be greater than $\Delta T_{\rm survey}$ (if not, sim aborts). For recurring-periodic, the `LCLIB` event cycle is internally repeated until it spans $\Delta T_{\rm survey}$. For non-recurring events that are shorter than $\Delta T_{\rm survey}$, the undefined epochs are given template magnitudes.

`LCLIB` Event Probabilities

Each recurring event is assumed to have the same probability. Non-Recurring events are fundamentally different because any given event has an arbitrary start time. For example, an event with a 1,000 year duration can start any time within the previous 1,000 years and still leave a signal. To describe the implementation we first define

$$\Delta T_{\rm Tot} \equiv \Delta T_{\rm survey} + \Delta T_{\rm evt} \tag{37}$$

where $\Delta T_{\rm evt}$ is the `LCLIB` event duration. A key assumption for the library is that the rate per unit time is the same for each `LCLIB` event, and thus an arbitrary rate of 1 per $\Delta T_{\rm survey}$ is assigned. The mean number of generated events is therefore $\langle N_{\rm gen}\rangle = \Delta T_{\rm Tot}/\Delta T_{\rm survey}$. A Poisson generator picks $N_{\rm gen}$ and the `LCLIB` event is re-used $N_{\rm gen}$ times, each time with a different cadence and sky location. For a 3-year survey ($\Delta T_{\rm survey} = 3$), a 1 month event ($\Delta T_{\rm evt} = 0.083$) results in $\langle N_{\rm gen}\rangle = 1.028$. For the same survey, a century-long event ($\Delta T_{\rm evt} = 100$) results in $\langle N_{\rm gen}\rangle = 34.33$. Beware that including `LCLIB` events with vastly different $\Delta T_{\rm evt}$ values results in very large asymmetries in how many times each `LCLIB` event is used.

Template Mags

A template magnitude is determined for each simulated event and each band. The last argument after the `GENMODEL` key (§9.11.1) determines a list of user-requested template epochs to average, and the "T:" rows in the `LCLIB` are used to interpolate mag values at the user-requested template epochs. Each template mag is converted to flux, the flux values are averaged, and the average flux

is converted back into a template mag to use for subtraction. Variations in observing conditions are NOT included in template-mag computation.

For long-duration non-recurring events ($\Delta T_{\rm evt} \gg \Delta T_{\rm survey}$) that start well before the survey, the true template mags are not observed. In this case, the event mags a few months prior to the survey are used to compute the template mags. For short-duration non-recurring events ($\Delta T_{\rm evt} \ll \Delta T_{\rm survey}$), the true template mag is used. For intermediate cases ($\Delta T_{\rm evt} \sim \Delta T_{\rm survey}$), the true template mag is used if the event starts after the survey begins; otherwise the template is computed from light curve mags shortly before the survey.

Angle Matching

The optional `ANGLEMATCH` or `ANGLEMATCH_b` key in the event header specifies how close the simulated sky location must be to the `LCLIB` event. The simulation will keep reading `LCLIB` events until an angle-match is found. This option allows Galactic event profiles to depend on sky location. `ANGLEMATCH` specifies a radius (degrees), while `ANGLEMATCH_b` applies only to Galactic latitude (b), and ignores the longitude coordinate.

To increase the efficiency for finding an `LCLIB` event satisfying `ANGLEMATCH_b`, a symmetry for events at $-b$ and $+b$ is used to require

$$| \, |b_{\rm SIM}| - |b_{\rm LCLIB}| \, | < \texttt{ANGLEMATCH\_b} \tag{38}$$

where $b_{\rm SIM}$ is for the simulated event and $b_{\rm LCLIB}$ is for the `LCLIB` event.

Parameter Selection

To select a limited range of `PARVAL` values from the library, add the following key(s) to the sim-input file:

```
LCLIB_CUTWIN:  <PARNAME>  <CUTMIN> <CUTMAX>
```

Note that this option does not reduce the number of generated events; rather it reduces the `LCLIB` size by excluding `LCLIB` events.

9.11.4 `GENMODEL_MSKOPT` Options

- `GENMODEL_MSKOPT += 1`: ignore ANGLEMATCH.
- `GENMODEL_MSKOPT += 8`: After selecting RA,DEC from `DNDB` distribution, replace with RA,DEC in LCLIB, and also update MWEBV. This option is useful if `DNDB` rate model is inadequate, but *beware* that incorrect cadences are used with this option; i.e, the cadence from SIMLIB-{RA,DEC} is used even though the LCLIB coordinates are different. Note that a more accurate strategy is to create a SIMLIB cadence library with the correct RA,DEC distribution (and `GENMODEL_MSKOPT=0`).
- `GENRANGE_MJD: <MJDmin> <MJDmax>` : select MJD window

```
DOCUMENTATION:
  PURPOSE: model XXX galactic transient
  REF:
  - AUTHOR:  A. Author et al 2020 (few word summary/reminder)
    ADS:     https://ui.adsabs.harvard.edu/abs/XXX
  USAGE_KEY: GENMODEL
  USAGE_CODE: snlc_fit.exe
  NOTES:
  - e.g., created from another code
  - e.g., model limitations
  PARAMS:
  - PAR1 is bla bla
  - PAR2 is bla bla
DOCUMENTATION_END:

# global header info
SURVEY:  LSST
FILTERS:  ugrizY
MODEL:    MY_MODEL_NAME
MODEL_PARNAMES:   PAR1,PAR2,PAR3    # comma-separated strings, no spaces
NEVENT:   <NEVENT>
RECUR_CLASS:  RECUR-NONPERIODIC  (or RECUR-PERIODIC or NON-RECUR)

START_EVENT:  <ID1>             # internal integer ID
NROW:  <NROW>     RA: <RA>      DEC: <DEC>
PARVAL:   <VAL1>,<VAL2>,<VAL3>        # comma-separated or space-separated
ANGLEMATCH:    10.0         # optional (deg) to match with cadence library
T:  <DAY %7.4f>   <mag_u %.3f>  <mag_g> . . . . <mag_Y>
T:  <DAY %7.4f>   <mag_u %.3f>  <mag_g> . . . . <mag_Y>
S:  <DAY %7.4f>   <mag_u %.3f>  <mag_g> . . . . <mag_Y>
S:  <DAY %7.4f>   <mag_u %.3f>  <mag_g> . . . . <mag_Y>
etc ...
END_EVENT:  <ID1>

# ... continue next event

# Multiple key options
#   NOBS: and NROW:
#   MODEL_PARNAMES: and MODEL_PARAMETERS:
```

Figure 17: TEXT-library format for LCLIB model type.

9.11.5 Special MODEL_PARNAMES Features

The MODEL_PARNAMES are read and passed to output data files and simgen-dump file. These parameters are not used to generate events, except for two special features:

- **MWEBV (mwebv):** indicates that MWEBV has been computed and applied to the model light curves, perhaps using a 3D dust model. The simulation reports MWEBV based on OPT_MWEBV (e.g., SFD98), but does not make a redundant MWEBV correction to the true magnitudes (SIM_MWEBV=0). To avoid a MWEBV-naming conflict, the LCLIB-MWEBV is renamed MWEBV_LCLIB for the data files and simgen-dump file.
- **REDSHIFT (redshift):** indicates a true non-zero redshift for the model light curve (default LCLIB redshift is $z = 0$), and can be used with a HOSTLIB. This feature may be convenient for extraGalactic models such as AGN.

9.12 PySEDMODEL: Python SED models

To accomodate a broader class of SED models in the simulation, there is a C-python interface for python SED models (`PySEDMODEL`) to return i) rest-frame SED at requested phase, and ii) relevant internal parameters. The rest-frame SED is used by the simulation C-code[29] to compute broadbad fluxes and noise in the same way as any other SED model (e.g., SALT2, NON1ASED ...). The internal parameters are not used by the C-code, but are passed to the output data files and/or `SIMGEN_DUMP` file. There are currently 2 python models under development which can be called with the following syntax:

```
GENMODEL:  BYOSED  [path]  # Build-Your-Own SED
GENMODEL:  SNEMO   [path]  # based on SNFactory data

GENMODEL_MSKOPT: [mask] # integer option or mask
GENMODEL_ARGLIST: 'key0 val0 key1 val1 key2 x0,x1,x2,x3 etc ...'
```

The `GENMODEL` key is followed by a `PySEDMODEL` name and path. The path contains formatted information & instructions that are parsed by the init function in the python code.

The last two keys allow passing additional options to the python code, both in the form of an integer flag or char-string. These arguments can be passed on the command line to override default behavior, e.g.,

```
snlc_sim.exe <myInputFile> GENMODEL_MSKOPT 8
    or
snlc_sim.exe <myInputFile> GENMODEL_ARGLIST 'TAU_AV 0.33 ZETA 4.3'
```

The simulation C code does not parse these argument, but simply passes these arguments to the python init function.

[29] `PySEDMODEL` does not connect with SNANA light curve fitting codes.

10 Computing Distance Moduli from BBC Method: SALT2mu.exe

The SNANA program SALT2mu.exe reads SALT-II fitted parameters, performs a global fit for α, β, γ, which are SNIa standardization parameters associated with stretch (x_1), color (c), and host-galaxy mass ($M_{\rm host}$), respectively. SALT2mu.exe also fits for distance offsets in redshift bins, and the output includes binned distance moduli. The SALT2mu program can implement the "BEAMS with Bias Corrections" (BBC) method in [33, 34], to produce bias-corrected distances in redshift bins. To include bias corrections, SALT2mu reads SALT2-fitted parameters for the data and for a large "biasCor" simulation. The biasCor simulation should be generated on a grid of α, β, γ as described in §9.3.1.

A sample SALT2mu.exe-input file is in

```
$SNDATA_ROOT/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.

10.1 Catenating FITRES Files from Multiple Samples

SNIa-cosmlogy analyses include multiple samples (e.g., LOWZ, SDSS, PS1, SNLS, DES ...). While SNANA codes perform light-curve fitting and simulations separately for each sample, the SALT2mu/BBC program performs a global fit to all of the data samples combined. Through SNANA code versions v10_76e, each FITRES table file (produced from snlc_fit.exe) was required to have the same columns so that a simple concatenate combines the data into a single file. Starting at v10_77, input FITRES tables are allowed to have different columns, and the output FITRES table from SALT2mu is created with the following logic:

- column names appearing in each FITRES file are included, regardless of column locations.
- varname_pIa column is included; if missing (e.g., from LOWZ), then -9 values are substituted.
- the same "varname_pIa" logic is applied to variables specified with SALT2mu-input key append_varnames_missing. The default is PROB* to include all classifier probabilities.

As an example, consider an analysis combining a spec-confirmed LOWZ sample and a photometrically-identified HIGHZ sample. The LOWZ sample contains several peculiar-velocity estimates (VPEC1, VPEC2, VPEC3) that are not in the HIGHZ table; the HIGHZ sample includes several classifier probabilities (PROB_A, PROB_B, PROB_C) and sSFR that are not in the LOWZ table. All of these values are included in the SALT2mu output using the following SALT2mu-input key,

```
append_varnames_missing='PROB*,VPEC*,sSFR'
```

The wildcard (*) picks up any column name containing PROB or VPEC. An exact sSFR column match is also selected. The VPEC values are set to −9 for the HIGHZ sample, and similarly the PROB and sSFR values are set to −9 for the LOWZ sample. The top of the output FITRES file includes comment-summary of i) input FITRES file names, ii) append_varnames_missing, and iii) column names that were discarded.

Combining fitres files with different columns may be useful for other (non-SNANA) codes, and therefore this SALT2mu "FITRES-combine" logic can be implemented as a utility without the overhead of fitting,

```
SALT2mu.exe cat_only \
     datfile=LOWZ.FITRES,HIGHZ.FITRES \
     append_varname_missing='PROB*,VPEC*,sSFR' \
     catfile_out=CATFILE.FITRES
  or
sntable_cat.py \
     -i  LOWZ.FITRES,HIGHZ.FITRES \
     -a  PROB*,VPEC*,sSFR \
     -o  CATFILE.FITRES
```

The cat_only argument (replacing input file) directs SALT2mu to skip fitting and perform only the catenation. Beware that the wildcard (*) causes problems with unix; therefore use either single quotes as shown above, or use a backslash (PROB*,VPEC*,sSFR).

A python wrapper (sntable_cat.py) is also available with a more obvious code-name to search in $SNANA_DIR/util. This python wrapper can parse wildcards without single quotes. The -i and -o arguments are required; -a is optional (default is PROB*).

10.2 Blinding BBC Output from SALT2mu.exe

There are two methods for blinding results in the BBC output: 1) totally blind, and 2) blurry. For the totally blind method,

$$\text{MUDIF} \rightarrow \text{MUDIF} + \cos(10z) \tag{39}$$

where the cosine argument is ten × redshift. The MUDIF (μ-offsets) are $O(1)$ and viewing these values provides no information about the true residuals; hence "totally blind." The disadvantage is that the cosmology fitting program, which reads the BBC output, must subtract $\cos(10z)$ from each blinded μ. Also, the cosmology-fitting program must have its own blinding applied to the fitted cosmology parameters.

The blurry method adds unknown offsets to the reference cosmology parameters (Ω_Λ, w_0),

$$\Omega_\Lambda^{\rm ref} \rightarrow \Omega_\Lambda^{\rm input} + A_L \cos(B_L) \tag{40}$$

$$w_0^{\rm ref} \rightarrow w_0^{\rm input} + A_w \cos(B_w) \tag{41}$$

where $A_{L,w}$ and $B_{L,w}$ are arbitrary. The cosmology fitting program need not modify any of the BBC outputs since all of the BBC outputs are correct. $A_{L,w}$ and $B_{L,w}$ can be modified separately for each user, or each analysis iteration, and there is no need to pass these blinding parameters to other colleagues or programs. However, the same blinding params must be used for systematic comparisons of the Hubble diagram. The blinding strategy is that the user does not know the reference-cosmology parameters, and therefore the output μ-residuals (MUDIF) are offsets from an unknown cosmological model. However, if $A_{L,w}$ are not too large, the μ-residuals might still be somewhat close to zero. In this case, human inspection of the μ-residuals provides some information; hence we call this method "blurry."

By default, SALT2mu.exe will blind real data with the blurry method, and not apply blinding to simulated data. The following SALT2mu.exe inputs control blinding:

```
p9=0.7    # input Omega_L -> ref cosmology for MUDIF
p11=-1.0  # input w0      -> ref cosmology for MUDIF

blindflag=1           # totally blind, MUDIF += cos(10z)
blindflag=2           # blurry method (DEFAULT)
blindpar9=0.1,4400    # A_L,B_L: OL_ref = p9  + 0.10 * cos(4400)
blindpar11=0.2,3300   # A_w,B_w: w0_ref = p11 + 0.20 * cos(3300)

# add 64 to blind simulated data
 blindflag=65  # 1+64 -> same as blindflag = 1 for real data
 blindflag=66  # 2+64 -> same as blindflag = 2 for real data
```

It is highly recommended to practice blinding on simulated data before processing real data. Another useful feature is to generate simulations without the SIM_XXX truth values, which checks if the analysis codes are mistakenly using some SIM_XXX values. This sim option is invoked by adding 8 to FORMAT_MASK. For example, "FORMAT_MASK: 10" (2+8) produces text formatted data files that look like real data, and "FORMAT_MASK: 40" (2+32) produces FITS-formmated data files that look like real data. If the simulated data has no SIM_XXX keys, the default BBC output is blinded with the blurry method.

The output BBC tables include comment fields specifying SNANA_VERSION, BBC_VERSION, and a summary of the blinding parameters. For cosmology-fitting, the parsing/translation module should keep track of the integer BBC version to maintain backward compatibility in case of future modifications. If the BBC_VERSION key is not included, set BBC_VERSION=1.

11 Cosmology Fitting

State-of-the-art cosmology fitting tools include CosmoMC and CosmoSIS that are not included in `SNANA`. Within `SNANA`, `wfit.exe` is a fast cosmology fitting program that is not nearly as sohopisticated as CosmoSIS, but is useful for systematics tests and forecasts. `wfit.exe` models wCDM (w, Ω_M) and CPL w_0w_aCDM model, both assuming a flat universe. The CMB prior is based the R parametrization using simulated cosmology parameters and $\sigma_R = 0.007$ to reproduce the DES3YR uncertainties. The BAO prior is based on Alam 2020. Table 1 compares `wfit.exe` results to published DES3YR and Pantheon values using

```
wfit.exe  [HubbleDiagram] -cmb_sim -sigma_Rcmb 0.007   -wsteps 101 -omsteps 101 \
          -mucov_file [mucov_file]  \
          -bao_sim               # only for DES w0waCDM only
```

where `HubbleDiagram` and `mucov_file` are from public data releases.

	Ω_m	$\sigma(\Omega_m)$	w_0	$\sigma(w_0)$	w_a	$\sigma(w_a)$	FoM
DES3YR wCDM SN+CMB	0.321	$\pm$0.018	$-$0.978	$\pm$0.059			
`wfit.exe`	0.326	$\pm$0.018	$-$0.964	$\pm$0.059			
Pantheon wCDM SN+CMB	0.307	$\pm$0.012	$-$1.026	$\pm$0.041			
`wfit.exe`	0.308	$\pm$0.012	$-$1.029	$\pm$0.041			
DES3YR w_0w_aCDM SN+CMB+BAO	0.316	$\pm$0.011	$-$0.885	$\pm$0.114	$-$0.387	$\pm$0.430	46
`wfit.exe`	0.309	$\pm$0.012	$-$0.965	$\pm$0.092	$-$0.354	$\pm$0.442	64
Pantheon w_0w_aCDM SN+CMB	0.308	$\pm$0.018	$-$1.009	$\pm$0.159	$-$0.129	$\pm$0.755	31
`wfit.exe`	0.305	$\pm$0.015	$-$0.981	$\pm$0.159	$-$0.302	$\pm$0.725	32

Table 1: Comparison of published cosmology parameters to those obtained with `wfit.exe`. DES3YR results are from Table 2 in `https://arxiv.org/pdf/1811.02374.pdf`. Pantheon results are from Table 12 and 13 of `https://arxiv.org/pdf/1710.00845.pdf`. CMB prior in published results is from Planck 2016, and used in all four cases. CMB prior in `wfit.exe` uses R-parameter with $\sigma_R = 0.007$ tuned to match DES3YR w-uncertainty. The `wfit.exe` BAO is based on `https://arxiv.org/abs/2007.08991` . All fits use redshift binned data.

11.1 Interpreting Redshift Variables in SNANA Tables

The following redshift variables are included in the output tables from the fitting programs (`snlc_fit.exe, SALT2mu.exe`):

- `zHEL, zHELERR`: heliocentric redshift.
- `zCMB, zCMBERR`: redshift transformed to CMB
  ($\ell = 264.14$ deg, $b = 48.26$ deg, $v_{apex} = 370$ km/s)
- `VPEC, VPECERR`: peculiar-velocity correction.
  If host galaxy is moving away from (toward) Earth, then `VPEC<0` (`VPEC>0`).
- `zHD, zHDERR` $= (1 + \texttt{ZCMB})(1 + \texttt{VPEC}) - 1$. `zHDERR` is the quadrature sum of `zCMBERR` and `(1+zCMB)*VPECERR`. This corrected CMB redshift, or "Hubble-diagram" redshift, should be used to compute the luminosity distance (D_L) in cosmology-fitting programs.

To check that `VPEC` is used correctly with simulations, generate a large low-z sample, analyze with `snana.exe` or `snlc_fit.exe` program, and then verify the following from the `SNANA` or `FITRES` table:

```
1) zCMB - SIM_ZCMB  distribution has RMS=GENSIGMA_VPEC and mean=0
2) zHD  - SIM_ZCMB  distribution has RMS=VPEC_ERR and mean=0
```

where `GENSIGMA_VPEC` and `VPEC_ERR` are sim-input keys defined in §4.32.

11.2 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 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 [35], and these are available in

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

The minimization function is given by $\chi^2 = \sum_{ij}[\Delta_i V_{ij}^{-1}\Delta_j] - B^2/C$,[30] where $\Delta_i \equiv \mu_i^{\rm data} - \mu_i^{model}$ 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 [36]. The B and C parameters are

$$B = \sum_i (\Delta_i/\sigma_i^2) \quad \longrightarrow \quad \sum_{i,j} (\Delta_i V_{ij}^{-1}) \tag{42}$$

$$C = \sum_i 1/\sigma_i^2 \quad \longrightarrow \quad \sum_{i,j} V_{ij}^{-1} \ , \tag{43}$$

where σ_i is the diagonal-uncertainty on the distance modulus. The expressions left of the arrows (Eqs. 42-43) are from [36], 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; $\Delta_i \rightarrow \Delta_i - <\Delta>$.

[30] We leave out the constant term $\ln(C/2\pi)$.

12 Miscellaneous Tools and Features

12.1 Analysis-Output: Files, Tables, Variables

For each analysis program (`snana.exe, snlc_fit.exe, psnid.exe`) the output results are stored in tables. The primary table-file formats are `ROOT` or `HBOOK`, which allow writing multiple structures in parallel and include a plotting interface. `TEXT`-tables are also available for a subset of the variables, and there is a separate utility (see `sntable_dump`, §12.1.3) to extract information from a `ROOT` or `HBOOK` file and convert into text format. The available output tables are

- `SNANA` (ID=7100): general variables for all SN before fitting
- `FITRES` (ID=7788): fit results for SN passing fit requirements. Fit residuals for each epoch can be optionally included with `FITRES+RESIDUALS` .
- `OUTLIER` (ID=7800): flux-outliers w.r.t. sim-truth for `snana.exe` job; w.r.t. fit model for `snlc_fit.exe` job. Default is $> 3\sigma$; for `snana.exe`, default requires non-zero true flux.
- `SNLCPAK`: used by `mkfitplots.pl` (§5.10.1) to make light curve plots for data and best-fit model.

The `SNANA,FITRES,OUTLIER` tables are intended for continued analysis after the `SNANA` code finishes; The `SNLCPAK` table is for internal plotting scripts, and is not intended for general use. The user-selection of format(s) and table(s) is described below.

COMPILATION FLAGS

The `SNANA` codes will compile with `HBOOK` if ENV `$CERN_DIR` is defined, and will compile with `ROOT` if ENV `$ROOT_DIR` is defined. To compile without `HBOOK` or `ROOT`, make sure that the associated ENV is not defined. Do not modify any files to adjust compilation with `HBOOK` or `ROOT`.

SELECTING TABLE FORMAT(S)

The output table format is defined by the user in the `&SNLCINP` namelist,

```
HFILE_OUT       = 'myout.hbook'
ROOTFILE_OUT    = 'myout.root'
TEXTFILE_PREFIX = 'myout'
```

Multiple output formats can be specified, including all 3 as shown above. For the `ROOT` and `HBOOK` formats, all tables are written to each `ROOT` and `HBOOK` file. For the TEXT format, however, each table is written to a separate file and thus a file *prefix* is specified rather than a file-name. The corresponding output-file names are "`myout.[tableName].TEXT`" where the tableNames are `SNANA`, `FITRES`, `OUTLIER`, and `LCPLOT`.[31]

SELECTING TABLE(S)

The tables are selected by `&SNLCINP` namelist string `SNTABLE_LIST`,

```
SNTABLE_LIST = 'FITRES'                          ! default
SNTABLE_LIST = 'SNANA FITRES OUTLIER LCPLOT'  ! select'em all
```

[31] `LCPLOT` corresponds to the `SNLCPAK` table.

To protect memory and output file size for large jobs, the number of light curve plots (LCPLOT) is limited by &SNLCINP namelist variable MXLC_PLOT=100: set to a higher value to get more plotted light curves.

If a ROOT file is specified via &SNLCINP namelist variable ROOTFILE_OUT, then all SNTABLE_LIST arguments are created under the ROOT structure. Similarly, all tables are created under the HBOOK structure if HFILE_OUT is defined. For TEXT format with TEXTFILE_PREFIX, only the FITRES table is created by default; to generate text output for other tables requires additional specifications such as

```
SNTABLE_LIST = 'SNANA(text:key)  FITRES(text:key)  LCPLOT(text:col)'
     or
SNTABLE_LIST = 'SNANA(text:key)  FITRES            LCPLOT(text:col)'
```

where the available text formats are

- key : key before each row, plus keyed header.
- csv : comma-separated with header.
- col : space-separated columns with no header.

The two SNTABLE_LIST strings above are equivalent because "text:key" is the default text format for the FITRES table. Also note that the SNTABLE_LIST string is case-insensitive and thus the following are equivalent,

```
SNTABLE_LIST = 'snana(text:csv)  fitres(text:csv)  lcplot(text:csv)'
SNTABLE_LIST = 'SNANA(text:csv)  FITRES(text:csv)  LCPLOT(text:csv)'
SNTABLE_LIST = 'SNANA(TEXT:CSV)  FITRES(TEXT:CSV)  LCPLOT(TEXT:CSV)'
```

For the FITRES table, data-fit χ^2-residuals can be included for each epoch using a feature of both ROOT and HBOOK that allows vector elements in tables columns. In addition to the data-fit chi-squared, the following are also included: PSF, sky-noise, zero point, best-fit model flux, etc ... Residuals can be included with

```
SNTABLE_LIST = 'SNANA  FITRES+RESIDUALS  LCPLOT'
```

There is no text-format option for the residuals, but the sntable_dump utility (§12.1.3) has an "outlier" option to extract the residual information into text format. Finally, be aware that defining FITRES+RESIDUALS results in a significantly larger output file.

For the SNANA table, an optional cut-mask on CUTFLAG_SNANA can be defined to select the following subsets,

```
SNTABLE_LIST = 'SNANA(cutmask:0)'  ! write only events failing cuts
SNTABLE_LIST = 'SNANA(cutmask:1)'  ! write only events passing snana cuts
SNTABLE_LIST = 'SNANA(cutmask:2)'  ! write only events passing fit cuts
SNTABLE_LIST = 'SNANA(cutmask:3)'  ! default -> write all events
```

A few other miscellaneous table options are

```
SNTABLE_LIST = 'SPECPLOT'              # plot spectra
SNTABLE_LIST = 'SNANA+EPOCHS'          # include epoch info in SNANA table
SNTABLE_LIST = 'SNANA+SIM_MAGOBS'      # idem, but only MJD,IFILT,SIM_MAGOBS
SNTABLE_LIST = 'SNANA FITRES NOSIMVAR' # suppress SIM_xxx variables.
SNTABLE_LIST = 'SNANA FITRES MODELSPEC'# SALT2 model spectrum for each epoch
SNTABLE_LIST = 'SNANA(PS:10) FITRES'   # pre-scale SNANA table by 10
SNTABLE_LIST = 'FITRES NOZPHOT'        # suppress ZPHOT variables
SNTABLE_LIST = 'SNANA(text:host)'      # include HOSTGAL info in text tables
SNTABLE_LIST = 'SNANA(text:host2)'     # include HOSTGAL & HOSTGAL2 in text table
SNTABLE_LIST = 'SNANA(text:key,text:host)' # same, with format spec
SNTABLE_LIST = 'OUTLIER(nsig:4)'       # table with >4sig outlers (default: 3sig)
SNTABLE_LIST = 'OUTLIER(nsig:4,ftrue:0.0)'   # include obs with Ftrue=0
```

- `SPECPLOT` plots spectra (real or simulated); works with all formats. Note that `SNANA` has no spectral-analysis tools.

- `SNANA+EPOCHS` and `MODELSPEC` options work with `HBOOK` and `ROOT` format, but not with `TEXT` format.

- `SNANA+SIM_MAGOBS` is a compressed table to contain only the model light curves. EPOCH is defined on a grid w.r.t. Tpeak, and SIM_MAGOBS is interpolated at each grid-EPOCH so that model colors can be easily examined.

- The `NOSIMVAR` option removes all simulated truth values, and is useful for *blind* challenges.

- The `MODELSPEC` table works for SALT2 light-curve fits only, and is filled for epochs ($T_{\rm rest}$) defined by the SALT2 model: includes scalars (CID,MJD,BAND,TOBS,etc) and a model spectrum (flux vs. wavelength). The `MODELSPEC` table may be useful for calibration corrections that require an SED. To save storage, each spectrum is stored within the wavelength range defined by the passband.

- The pre-scale option (PS:10) reduces table output for large simulated samples.

- `NOZPHOT` suppresses photo-z output so that photo-z and spec-z (4-parameter) light curve fits have the same `FITRES` table columns and can therefore be combined. Note that `zHD` contains the relevant redshift information so that `ZPHOT` is not needed. In addition, `NOZPHOT` results in adding `OPT_PHOTOZ` (`&FITINP` input) to the `FITRES` table, allowing analysis programs (e.g., `SALT2mu`) to distinguish spec-z and photo-z events. For spec-z fits (i.e., NOT photo-z), `NOZPHOT` adds `OPT_PHOTOZ` to the `FITRES` table, but does not remove anything.

- The `OUTLIER` table is useful to identify and characterize flux outliers from non-Gaussian tails. The recommended method is using the `snana.exe` program on samples where the true flux is known (i.e., data files which have a `SIM_MAGOBS` column), such as fakes overlaid on image, or `SNANA`-simulated light curves. Running `snana.exe` with the `OUTLIER` table option will abort on real data because there is no true flux to compare. For real data (and sims), the `OUTLIER` table works with the lightcurve fitting program `snlc_fit.exe` by comparing fluxes to the best-fit model flux. This LC-fit option requires a high-confidence SNIa sample (e.g., spec-confirmed), otherwise some outliers may be artifacts of non-SNIa contamination.

An outlier-fraction summary (per band) is printed to stdout, and also to the top of TEXT formatted output. Finally, note that `snana.exe` is $\sim 10^2$ faster than `snlc_fit`, and thus the former can typically run interactively while the latter should run in batch mode.

12.1.1 SNTABLE_APPEND

An arbitrary column name and value can be added to output tables with command-line input

```
snlc_fit.exe <myInputFile>  SNTABLE_APPEND MYVAR 0.03
```

which will include `MYVAR` column in all output tables with value 0.03. An example usage is measuring derivatives of SALT2 parameters w.r.t. changing the i-band zeropoint using the followig FITOPT options in the `CONFIG` yaml block,

```
FITOPT:
- MAGOBS_SHIFT_ZP 'i -0.03'   SNTABLE_APPEND ZP_i -0.03
- MAGOBS_SHIFT_ZP 'i -0.01'   SNTABLE_APPEND ZP_i -0.01
- MAGOBS_SHIFT_ZP 'i  0.01'   SNTABLE_APPEND ZP_i  0.01
- MAGOBS_SHIFT_ZP 'i  0.03'   SNTABLE_APPEND ZP_i  0.03
```

By concatinating the output `FITRES` files, it is easy to plot, for example, c vs. `ZP_i` for individual events.

12.1.2 Combining Ascii "Fitres" Files: combine_fitres.exe

As discussed in §5.3, 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 (§4.38.2). The utility '`combine_fitres.exe`' is useful to combine (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
```

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 Z_2 DLMAG_2
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 extra SNe will be ignored. If there are SNe in the first fitres file that are not in the other fitres files, then values of −888 are stored for the missing SNe. In addition to creating a merged fitres file, a merged Table file (`combine_fitres.hbook/root`) is also created in hbook and/or root format depending on the compile flag(s) in `sntools_output.h`. This table can be analyzed with `PAW` or `ROOT`, or using the `sntable_dump.py` utility (§12.1.3)

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`.'

12.1.3 `SNTABLE` Dump Utility: `sntable_dump.py`

The fitres text-file output from the light curve fit program `snlc_fit.exe` contains a small fraction of the analysis variables. The entire set of analysis variables is stored in a more compact "`SNTABLE`" structure using ROOT based on the &SNLCINP namelist inputs

```
ROOTFILE_OUT = 'mjob.root'
```

Additional `SNTABLE` formats (e.g., HDF5, Pickle) can be added within the existing architecture. There are two primary `SNTABLEs`:

```
                            snlc_fit/psnid
                   snana    fit results
-----------------------------------------------
ROOT  TABLE ID    SNANA      FITRES   (trees)
-----------------------------------------------
```

If you are fluent in `ROOT`, then you can analyze these tables directly without using text files. However, if you prefer a different analysis platform then `sntable_dump.py` can be used to extract table information into text format.

The script usage is explained at the top of `&SNANA_DIR/util/sntable_dump.py`, and here we illustrate its basic features:

1. Print list of `SNTABLE` variables:

```
sntable_dump.py  myjob.root   SNANA
sntable_dump.py  myjob.root   FITRES
```

2. For an arbitrary set of table variables, extract SN values into a fitres-formatted text file:

```
sntable_dump.py  myjob.root  SNANA  -v RA,DEC,SNRMAX_r
sntable_dump.py  myjob.root  FITRES -v m0obs_g,m0obs_r
```

3. Append an existing text file with variables from a `SNTABLE`:

```
sntable_dump.py  myjob.root  SNANA  -v RA,DEC,SNRMAX_r  -a MYJOB.FITRES
sntable_dump.py  myjob.root  FITRES -v m0obs_g,m0obs_r  -a MYJOB.FITRES
```

4. Dump epoch-list of dataFlux-fitFLux outliers:

```
sntable_dump.py  myjob.root   FITRES --outlier  3  99
sntable_dump.py  myjob.root   FITRES --outlier  3  99 -v FITPROB,SNRMAX3
      (print epochs to ascii file with 3 < Nsigma < 99)
```

5. Dump epoch-list of dataFlux-simFLux outliers (sim only):

```
sntable_dump.py  myjob.root   FITRES --outlier_sim  3  99
sntable_dump.py  myjob.root   FITRES --outlier_Sim  3  99 -v FITPROB,SNRMAX3
      (print epochs to ascii file with 3 < Nsigma < 99)
```

6. Dump observations & variables needed to compute flux-error maps in §4.14.1. Note that ERRTEST= $\sigma_{\rm SIM}/\sigma_F$, where $\sigma_{\rm SIM}$ is the calculated error and σ_F is the reported error.

```
  sntable_dump.py  myjob.root   FITRES  --OBS
```

Using the append option (#3 above with --a), a new text file is created containing the original variables in MYJOB.FITRES plus the appended variables specified by the --v argument. The appended variables are extracted from the SNTABLE (2nd argument of sntable_dump.py).

To see pre-explosion epochs with the outlier option, open the low-side of &FITINP namelist variable FITWIN_TREST, such as "FITWIN_TREST: -100, +45"

12.1.4 Extract Value from Fitres File: get_fitresValue.pl

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

12.1.5 Working with IAUC names

Data files contain a required SNID and an optional IAUC name. By default, only the SNID is used and propagated to the output tables as CID. However, using the IAUC name may sometimes be more practical. For example, SNID could be an integer that keeps changing as data are reprocessed, while the IAUC name remains fixed.

On the input side, &SNLCINP namelist inputs that specify CID strings will work with either SNID or IAUC names. These namelist inputs include

```
SNCCID_LIST
SNCCID_LIST_FILE
SNMJD_LIST_FILE
```

SNCCID_LIST is an explicit list of SNIDs and/or IAUC names, and the list can contain a mix of SNID and IAUC names. The next two XXX_LIST_FILE inputs point to a file containing a list of SN names, and these can also be an arbitrary mix of SNID and IAUC names.

In the output tables, the CID column can be filled with the IAUC name by specifying an IAUC option in SNTABLE_LIST,

```
SNTABLE_LIST = 'SNANA FITRES(iauc)'
  or
SNTABLE_LIST = 'SNANA FITRES(text:key,iauc)'
  or
SNTABLE_LIST = 'SNANA(iauc) FITRES(text:key)'
  etc ...
```

If any one table has the IAUC specification then all tables will have IAUC written into the CID column.

12.1.6 MARZ Format for Spectra

After simulating host spectra, the spectra can be translated for analysis with the Marz utility[32] by specifying Marz output format for the snana.exe program,

```
 &SNLCINP
   MARZFILE_OUT = '***marz***.fits'
   VERSION_PHOTOMETRY = 'XYZ'
 &END
        or
snana.exe NOFILE VERSION_PHOTOMETRY XYZ   MARZFILE_OUT '**marz**.fits'
```

The argument of MARZFILE_OUT must include "marz" in the file name. Only the first spectrum per event (assumed to be HOST) is stored. SN spectra are not stored because Marz works only with host spectra.

12.2 General Misc. Tools

12.2.1 Batch Submit Script: submit_batch_jobs.sh

The SNANA script "submit_batch_jobs.sh" is a utility to run analysis codes in batch queue, and it includes preparation and post-processing. The script parses your input file and automatically determines which task to prepare and execute: snlc_sim.exe, snlc_fit.exe, SALT2mu.exe, SALT2-training code. Command line options are obtained with

```
submit_batch_jobs.sh -h
```

and detailed help on the config file for each task is given with

```
# detailed help on analysis programs
 submit_batch_jobs.sh -H SIM          # simulation
 submit_batch_jobs.sh -H FIT          # light curve fitting
 submit_batch_jobs.sh -H BBC          # BBC: Beams with Bias Corrections
```

[32] http://samreay.github.io/Marz (works with Chrome & Firefox, not with Safari)

```
   submit_batch_jobs.sh -H TRAIN_SALT2   # SALT2-traing

  # misc help
   submit_batch_jobs.sh -H MERGE      # post-processing merge task
   submit_batch_jobs.sh -H AIZ        # abort-if-zero logic
   submit_batch_jobs.sh -H TRANSLATE  # translate input file for perl scripts
```

The misc help options provide explanations that may be useful for debugging or further script development. The `MERGE` task can include complex operations after all of the batch jobs have finished. `AIZ` explains the subtle logic to abort when jobs return zero events, but not abort on low-stat jobs (e.g., Kilonova sims) where a subset of jobs are expected to have zero events. Finally, `TRANSLATE` is a feature to automatically translate legacy input files used by the obsolete perl batch scripts.

Historical note: prior to `SNANA` version `v11`, three separate perl scripts ran batch jobs: `sim_SNmix.pl`, `split_and_fit.pl`, `SALT2mu_fit.pl`. After `v11`, these three scripts are replaced by an improved python script invoked with `submit_batch_jobs.sh`.

12.2.2 List Valid Input Keys

Each `SNANA` program has many input keys that are difficult to find and remember. For `SALT2mu` and simulation programs, the following results in a dump of all valid input keys and then quits:

```
   snlc_sim.exe KEY_DUMP
   SALT2mu.exe  KEY_DUMP
```

Using grep can help find keys related to a particular topic, although a decent guess is needed for a sub-string. Examples for the simulation are:

```
# to find peculiar-velocity keys in the sim
   snlc_sim.exe KEY_DUMP | grep VPEC
       VALID_KEYNAME:   124   GENSIGMA_VPEC:
       VALID_KEYNAME:   125   VPEC_ERR:

# to find keys controling SALT2x1 population.
   snlc_sim.exe KEY_DUMP | grep SALT2x1
       VALID_KEYNAME:   200   GENPEAK_SALT2x1:
       VALID_KEYNAME:   201   GENMEAN_SALT2x1:
       VALID_KEYNAME:   202   GENSIGMA_SALT2x1:
       VALID_KEYNAME:   203   GENSKEW_SALT2x1:
       VALID_KEYNAME:   204   GENRANGE_SALT2x1:
       VALID_KEYNAME:   205   GENGRID_SALT2x1:
       VALID_KEYNAME:   206   GENPROB2_SALT2x1:
       VALID_KEYNAME:   207   GENPEAK2_SALT2x1:
       VALID_KEYNAME:   208   GENSIGMA2_SALT2x1:
```

and here is a SALT2mu example to find file-related inputs,

```
SALT2mu.exe KEY_DUMP | grep file
    VALID_KEYNAME:     1   catfile_out=
    VALID_KEYNAME:     9   file=
    VALID_KEYNAME:    10   datafile=      (same as file=)
    VALID_KEYNAME:    11   simfile_bias=
    VALID_KEYNAME:    12   simfile_biascor=
    VALID_KEYNAME:    25   simfile_ccprior=
    VALID_KEYNAME:    70   cid_select_file=
```

The KEY_DUMP command-line option works for the C programs, but not for the fortran programs (snlc_fit.exe, snana.exe, psnid.exe). Instead, the fotran programs have a built-in "name-list" (NML) feature, and the contents of each NML are printed to standard out. Therefore, use grep on stdout to find NML inputs; e.g.,

```
snlc_fit.exe SNFIT.NML  | grep CUTWIN_TREST
  CUTWIN_TRESTMIN= -99.000000    , -2.0000000    ,
  CUTWIN_TRESTMAX=  10.000000    ,  9999.0000    ,
  CUTWIN_TRESTRANGE=-9.99999996E+11, 9.99999996E+11,
  CUTWIN_TREST= -20.000000    ,  80.000000    ,
  CUTWIN_TREST_TRUEFLUX= 2*999.00000      ,
  CUTWIN_TREST2=-9.99999996E+11, 9.99999996E+11,
```

The large range values (+- 9.999E+11) indicate that the NML input was not defined in the user-input file, but is a valid input.

12.2.3 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.

12.2.4 quick_commands,py

Many useful/quick commands can be run from a python script

```
quick_commands.py -h   # standard python args help
quick_commands.py -H   # examples
```

Options include

- translate TEXT formatted data to FITS format
- extract a few events from FITS format to TEXT format
- create SIMLIB file from LOWZ data that has no metaData.
- make human-readable table of redshifts and vpec
- Create SNANA-formatted table(s) from data or sim
- extract info about a photometry version
- extract info about SNANA code

12.2.5 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.

12.2.6 Write List of Rejected Epochs

The most commonly used output of the snlc_fit.exe fitting program is the "FITRES" table, which includes summary information for each SN passing selection cuts. However, there may be analyses that require a list of selected epochs, such as the SuperNNova photometric classifier[33] that reads light curves instead of the summary FITRES file. To avoid duplicating epoch cuts in other codes, there is an SNANA utility to write a list of rejected epochs, with the assumption that writing rejected epochs results in a much smaller file compared to writing selected epochs. An example for the DES analysis is

```
&SNLCINP
   PRIVATE_DATA_PATH  = '$DES_ROOT/lcmerge'
   VERSION_PHOTOMETRY = 'DESALL_forcePhoto_real_snana_fits'
   PHOTFLAG_MSKREJ  = 1016, 0     ! reject bits for PHOTFLAG columm
   CUTWIN_PHOTPROB  = -999, 999   ! wide open -> not used here
```

[33] https://arxiv.org/abs/1901.06384

```
   CUTWIN_PSF       = 0.5,  2.75  ! PSF select window, arcsec
   CUTWIN_ZPNPE     = 30.5, 100.  ! zeropoint select window, Npe
   OUT_EPOCH_IGNORE_FILE = 'DES_EPOCH_IGNORE.DAT'
&END
```

In this example there are four epoch-based cuts on PHOTFLAG, PHOTPROB,[34] PSF, and zeropt. The last user input, OUT_EPOCH_IGNORE_FILE, writes every rejected epoch to the specified epoch-reject file. The output table columns are

```
VARNAMES: CID MJD BAND PHOTFLAG PHOTPROB ZP PSF CUTMASK
```

and the bottom of the epoch-reject file includes a commented statistics summary for all rejected epochs, and for each cut. To enable more detailed cut studies, the CUTMASK column indicates which cuts resulted in rejecting each epoch; see CUTMASK comments at top of epoch-reject file.

12.2.7 Restoring Bugs

Following the publication of an analysis, SNANA bugs are found and fixed, and algorithms are improved. While all SNANA code versions are archived for reproducibility, it may be convenient to include a flag to restore original code, including bugs. Such a flag allows using the same [current] code version to process data with & without the code changes. The first such flag is for the DES 3-year (DES3YR) cosmology results published Winter 2019:

```
RESTORE_DES3YR: 1  # sim-input

&SNLCINP
   RESTORE_DES3YR = T  # LC fit input
```

[34] Example PHOTPROB uses are machine-learning prob or PSF fit-chi2.

12.2.8 Crosscheck $z_{\rm hel} \rightarrow z_{\rm cmb}$ in Data Files

In some data files, the CMB-frame redshift has been evaluated using the additive approximation, $z_{\rm cmb} = z_{\rm hel} + z_\odot$, instead of the exact formula, $1+z_{\rm cmb} = (1+z_{\rm hel})/(1-z_\odot)$. If both `REDSHIFT_HELIO` and `REDSHIFT_CMB` are included in the data files, the snana code can run a crosscheck with the following input,

```
&SNLCINP
   ZTOL_HELIO2CMB = 0.001  ! abort if dz/zCMB > 0.001 (default=0.02)

or command-line arg:
  snana.exe <inpFile> ZTOL_HELIO2CMB 0.001
```

The `SNANA` code computes $z_{\rm cmb}$ from $z_{\rm hel}$ and compares with `REDSHIFT_CMB` in the data file; if the difference (`dz`) satisfies `dz/zCMB > ZTOL_HELIO2CMB`, the code aborts. The default tolerance is 0.02, which may catch severe bugs, but will not catch the additive approximation.

12.2.9 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 KCOR_FILE  Hsiao/kcor_SNLS_Bessell90_VEGA.fits  \\
               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 `KCOR_FILE`). 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 README and DASHBOARD Utilities

```
snlc_sim.exe mysim.input README     # write README file and quit
snlc_sim.exe mysim.input DASHBOARD  # print all map and library file names
```

The README argument forces the sim to write the standard README ouptut file with the OVERVIEW, INPUT_KEYS and INPUT_NOTES yaml blocks, and then the sim quits. For speed, most initializations are skipped such as for HOSTLIB and transient model.

There are well over a dozen possible input files (maps, libraries ...) to the simulation, and keeping track can be difficult, especially with include files and command-line overrides. A brief summary of every input file can be displayed with the DASHBOARD command. Unused files are labeled NONE.

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,

$$\texttt{ZPT}(\texttt{COADD}) = 2.5\log_{10}\left[\sum_i 10^{(0.4\cdot\texttt{ZPT}_i)}\right] \qquad \texttt{NOISE}(\texttt{COADD}) = \sqrt{\sum_i \texttt{NOISE}_i} \quad , \tag{44}$$

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 [12],

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

When the "MWEBV" option is used, observation sequences with MWEBV> 2 are rejected. Read top of $SNANA_DIR/src/simlib_coadd.c for additional options.

12.3.3 Creating a SIMLIB from Data

Ideally, a SIMLIB is created from a library of observations containing PSF, sky-noise and zero-point. However, there may be cases where it is useful to create a SIMLIB directly from a data sample. A few examples are 1) low-z sample which has no meta-data to construct a SIMLIB file, 2) simulating fake SNe that have already been overlaid on images.

The snana.exe program can create such a SIMLIB via the &SNLCINP input key(s)

```
&SNLCINP
   SIMLIB_OUT = 'mydata.simlib'

 ! for sims, or fake data with SIM_MAGOBS :
   OPT_SIMLIB_OUT = 1    ! write SIM_MAGOBS in last SIMLIB column, all obs
   OPT_SIMLIB_OUT = 2    ! idem, but only when true flux > 0

! for systematic tests only
   SIMLIB_OUT_TMINFIX = 5   !  force PEAKMJD = MJD_MIN - 5

! optional method to include SALT2x1 and SALT2c in header
! if fitted c and x1 are in the header-override file:
  HEADER_OVERRIDE_FILE = '[fileName with c and x1 colums]'
```

and the NOFILE feature can be used, e.g.,

```
snana.exe NOFILE VERSION_PHOTOMETRY [version] SIMLIB_OUT mydata.simlib
```

If the light curve data include meta data (PSF,SKY,ZP) for each epoch, these values are written out to the SIMLIB. If the meta data are not available (e.g., low-z samples), then two assumptions are used to compute these values. First, the PSF(FWHM) is fixed to 1″. Second, the sky brightness (mag/asec^2) vs wavelength is taken from an old version of the LSST deep-drill cadence. For an arbitrary filter, the skyMag is interpolated as a function of the central wavelenth of the filter. Fortran subroutine COMPUTE_SIMLIB_INFO computes the SKYSIG and ZP values using the above assumptions and the S/N from the data.

If there are no meta-data, the default ZPERR is 0.01, which imposes a true and/or reported SNR ceiling of 100 in the simulation (for ZPERR usage, see SMEARFLAG_ZEROPT input in §4.7.1) Arbitrary ZPERR values can be input via &SNLCINP as shown in the following example:

```
 SIMLIB_ZPERR_LIST = 'u .02  gri .01  z 0.015'
```

Any band not listed above gets the default ZPERR=0.01.

If spectra are included in data (e.g., SNIa training sample), a TAKE_SPECTRUM key for each spectrum is automatically written to each LIBID header as follows

```
TAKE_SPECTRUM: TOBS(4.6)   SNR(1179)   SNR_LAMOBS(3590:7535)
```

The SNR and wavelength range (SNR_LAMOBS) correspond to the full wavelength range of the spectrum, and each simulated spectrum (below) is truncated in wavelength to match that of the data. Beware that the average SNR of simulated spectra match the data, but the λ-dependence of SNR is from the spectrograph table (Fig. 3) and may not match the data.

The SIMLIB is updated for events passing cuts in the &SNLCINP namelist, and a SIMLIB row is written for each epoch. The SIMLIB header for each event includes the redshift, peak-MJD, and spectra-SNR info. When using this SIMLIB in the simulation, header values are selected with following sim-input keys (§4.7.1):

```
USE_SIMLIB_PEAKMJD:  1  # use peakMJD in header (if there)
USE_SIMLIB_REDSHIFT: 1  # use redshift in header (if there)
USE_SIMLIB_SPECTRA:  1  # use TAKE_SPECTRUM key(s) in header
```

A few warnings: 1) sim-input keys GENRANGE_REDSHIFT and GENRANGE_PEAKMJD are applied and therefore some SIMLIB entries can be rejected, 2) the simulation may generate more than NGENTOT_LC events so that NGENTOT_LC events are generated within the GENRANGE_REDSHIFT[PEAKMJD] windows, 3) TAKE_SPECTRUM keys are used from either the SIMLIB header or sim-input file, but not both; the simulation will abort if both are requested.

The OPT_SIMLIB_OUT feature writes the true (simulated) mags in the last column of the SIMLIB file. Beware that overlapping fields are not treated properly, so it is recommended to set CUTWIN_NFIELD = 1,1. The "SIMLIB" model feature in the simulation (§9.10) reads these mags.

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

Errors and S/N can be fudged in the simulation as follows:

```
# 1. 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
  EXPOSURE_TIME_FILTER: g,r 20,30   # band-dependent increase

  FUDGESHIFT_ZPT:  0.03           # add 0.03 to ZPT in all bands
  FUDGESHIFT_ZPT_FILTER: g  0.03  # add 0.03 to g-band ZPT only
  FUDGESHIFT_ZPT_FILTER: gz 0.03  # add 0.03 to g and z-band ZPT
  FUDGESHIFT_ZPT_FILTER: g,r,i,z  0.03,-0.02,0.01,0.04  # band-depdendent ZPT

# 2. options to increase SKY-noise while leaving SN flux unchanged
  FUDGESCALE_PSF: 2            # increase PSF
  FUDGESCALE_NOISE_SKY: 3      # increases SKYNOISE (SKY *= 3^2)
  FUDGESCALE_NOISE_TEMPLATE 0  # remove template noise

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

# 4. force nearest-peak S/N in r-band; use same EXPOSURE time in other bands
  FUDGE_SNRMAX:  r=25      # adjust exposure time for r-band

# 5. add magErr to all epochs
  FUDGE_MAGERR: 0.03               # add .03 magErr to all epochs and bands
  FUDGE_MAGERR_FILTER: g     .01   # add .01 magErr to g-band
  FUDGE_MAGERR_FILTER: riz   .02   # add .02 magErr to riz bands
  FUDGE_MAGERR_FILTER: r,i,z .01,0.02,0.03   # band-dependent

# 6. add ZPTERR to all epochs
  FUDGE_ZPTERR: 0.03             # add .03 ZPTERR to all epochs and bands
  FUDGE_ZPTERR_FILTER: g   .01   # add .01 ZPTERR to g-band
  FUDGE_ZPTERR_FILTER: riz .02   # add .02 ZPTERR to riz bands
  FUDGE_ZPTERR_FILTER: r,i,z  0.02,0.01,0.005

# 7. shift filter wavelength
  FUDGESHIFT_LAM: 10.0                              # add 10 A to all bands
  FUDGESHIFT_LAM_FILTER: gr       7.2               # add 7.2A to g+r bands
  FUDGESHIFT_LAM_FILTER: g,r,i,z  2.2,-3.3,4.4,1.1  # band-dependent
```

There are six 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, (3) force nearest-peak S/N to be the same fixed value in all bands, (4) force nearest-peak S/N for one band, then use computed, `EXPOSURE_TIME` in all bands, (5) add arbitrary mag-error at each epoch, (6) add zero point error for each eppch. The last two options (`MAGERR` and `ZPTERR`) are similar, but have a subtle

difference. FUDGE_MAGERR is applied to the model magnitudes regardless of the instrumental parameters. FUDGE_ZPTERR replaces the ZPTERR column in the SIMLIB file, which applies measurement scatter based on the SMEARFLAG_ZEROPT key. The 1-bit applies scatter to the true fluxes, the 2-bit includes ZPTERR in the reported flux-uncertainties, and "SMEARFLAG_ZEROPT: 3" does both.

FUDGE2_SNRMAX is useful for setting a nearly fixed error at all epochs, in contrast to a fixed S/N, or 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 SNRMAX[35] is used to calculate the EXPOSURE_TIME for the second iteration. For option 3), the EXPOSURE_TIME is adjusted separately in each band to get the same fixed S/N, while in option 4) the EXPOSURE_TIME is the same in each band to fix S/N in just one band. The EXPOSURE_TIME value for each band is written to the header in each data file; see SIM_EXPOSURE key. Defining SNR_i to be the i'th-iteration S/N where SNR_2 is the requested FUDGE[2]_SNRMAX value, and $\mathcal{R} \equiv \mathrm{SNR}_2/\mathrm{SNR}_1$, the zeropoint and sky-noise are modified for each passband in the second iteration as follows:

$$\texttt{FUDGE\_SNRMAX}: \quad \mathrm{EXPOSURE\_TIME}_2 = \mathcal{R}^2 \tag{45}$$

$$\texttt{FUDGE\_SNRMAX}: \quad \mathrm{ZPT}_2 - \mathrm{ZPT}_1 = 5\log(\mathcal{R}) \tag{46}$$

$$\texttt{FUDGE\_SNRMAX}: \quad \sigma_{\mathrm{sky},2}/\sigma_{\mathrm{sky},1} = \mathcal{R} \tag{47}$$

$$\texttt{FUDGE2\_SNRMAX}: \quad \mathrm{ZPT}_2 - \mathrm{ZPT}_1 = 0 \tag{48}$$

$$\texttt{FUDGE2\_SNRMAX}: \quad \sigma_{\mathrm{sky},2}/\sigma_{\mathrm{sky},1} = \sqrt{[\ (1/\mathrm{SNR}_2)^2 - 1/\hat{F}_{\mathrm{SN}}\ ]\ /\ [\ (1/\mathrm{SNR}_1)^2 - 1/\hat{F}_{\mathrm{SN}}\ ]} \tag{49}$$

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

Finally, FUDGESHIFT_ZPT and FUDGESHIFT_LAM can be used for systematic calibation shifts.

[35] SNRMAX is the maximum S/N ratio among all observations within a passband.

12.3.5 DOCANA: Documenting Input Maps

The simulation reads up to 20 different inputs maps and libraries (see DASHBOARD utility in §12.3.1), and thus documenting is a challenge. To enforce/remind about documentation, each map is required to include a yaml-compliant documentation block at the top of the file, which is called DOCANA. The YAML format enables higher-level pipelines to extact the DOCANA snippets and glue them together in a summary file. Examples can be found using find command, e.g.,

```
cd $SNDATA_ROOT/simlib
grep -rnw ./ -e 'DOCUMENTATION:'
```

and here is a generic illustration,

```
DOCUMENTATION:
  PURPOSE: what is this for ?
  REF:
  - AUTHOR: Y. Astro et al 2017 (few-word reminder)
    ADS:    https://ui.adsabs.harvard.edu/abs/xxx
  - AUTHOR: C. Telescope et al 2014 (few-word reminder)
    ADS:    https://ui.adsabs.harvard.edu/abs/xxx
  INTENT:     Nominal or Test
  USAGE_KEY:  key name for input file
  USAGE_CODE: snlc_sim.exe
  VALIDATE_SCIENCE:
  - figure(s) in published paper
  - local/unpublished tests
  - None ?
  NOTES:
  - odd reminder
  - another odd reminder
  VERSIONS:
  - DATE:  2020-02
    AUTHORS: J. Cosmology, Y. Astro
DOCUMENTATION_END:
```

While the YAML block is required to avoid a simulation abort, specific keys are not required. Thus the REF key could be absent during development (or point to internal collaboration memo), while other relevant YAML keys can be added as needed. The INTENT key indicates nominal analysis, or a generic test such as debug, systematic, etc. VALIDATE_SCIENCE briefly indicates the level of validation; put NONE if there is no validation. For debugging or urgent/rare situations, the DOCANA requirement can be suppressed *interactively* with

```
snlc_sim.exe <myInputFile>  REQUIRE_DOCANA 0
```

As of July 19 2021, DOCANA is required for

```
HOSTLIB_FILE   HOSTLIB_WGTMAP_FILE  SIMLIB_FILE
```

Additional maps will be gradually added over the next few months.

12.4 Misc. Fitting Tools

12.4.1 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.

12.4.2 Translating TEXT data-files into FITS Format; and vice-versa

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
   PRIVATE_DATA_PATH     = 'MYDATAPATH'
   VERSION_PHOTOMETRY    = 'MYSURVEY_TEXT'
   VERSION_REFORMAT_FITS = 'MYSURVEY_FITS'
   OPT_SETPKMJD = 20   ! useful if there is no PEAKMJD estimate
   OPT_MWEBV    = 2    ! useful if there is no MWEBV in text files
 &END

or use NOFILE option with no input NML file:

snana.exe NOFILE  \
   PRIVATE_DATA_PATH      MYDATAPATH     \
   VERSION_PHOTOMETRY     MYSURVEY_TEXT  \
   VERSION_REFORMAT_FITS  MYSURVEY_FITS  \
   OPT_SETPKMJD 20                       \
   OPT_MWEBV    2
```

This translation should be used only for data since the simulation is written in FITS format by default. In addition to translating the data into FITS format, the auxiliary files are also created: `MYSURVEY.LIST`, `MYSURVEY.IGNORE`, `MYSURVEY.README`. The output files are created under a newly created folder (`/MYSURVEY_FITS`) in your current directory.

If there is no PEAKMJD estimate in the TEXT formatted files, `OPT_SETPKMJD>0` provides a PEAKMJD estimate for the FITS files. Similarly, `OPT_MWEBV>0` modifies `MWEBV` and `MWEBV_ERR` for

the FITS output. Finally, updated redshifts and v_{pec} are included in FITS output; e.g., from header override, etc ... Beware that content of TEXT and FITS may differ in these header variables.

While FITS formatted data is efficient for analysis, it is not so useful for visual inspection. To enable visual inspection, TEXT-formatted data files can be extracted from FITS-formatted files (data or sim) as follows:

```
snana.exe NOFILE  \
   VERSION_PHOTOMETRY     MYSURVEY_FITS  \
   VERSION_REFORMAT_TEXT  MYSURVEY_TEXT  \
   SNCID_LIST 8,1034,5944
```

where an explicit CID list or snana selection criteria can be applied. To avoid file-count quota issues, a max limit of 20 data files can be extracted.

Additional reformat options can be specified using `SNLCINP` input mask `OPT_REFORMAT_FITS` or `OPT_REFORMAT_TEXT`

```
   OPT_REFORMAT_FITS += 2   ! include rejected epochs
   OPT_REFORMAT_FITS += 32  ! rename band -> [SURVEY]-[band]
   OPT_REFORMAT_FITS += 64  ! exclude private variables
   OPT_REFORMAT_FITS += 128 ! include spectra
   OPT_REFORMAT_FITS += 256 ! exclude SIM truth to look like real data
```

Here is an example command to convert a simulation into a data-like sample that has no truth information:

```
snana.exe NOFILE \
  VERSION_PHOTOMETRY     CHALLENGE_SNIa  \
  VERSION_REFORMAT_FITS  CHALLENGE_SNIa_TRAIN \ # output data folder
  OPT_REFORMAT_FITS      256 \
  CUTWIN_SNTYPE 1 99

or the wildcard feature:

snana.exe NOFILE \
  VERSION_PHOTOMETRY_WILDCARD  'CHALLENGE_*'  \
  VERSION_REFORMAT_FITS  CHALLENGE_TRAIN \   # output data folder
  OPT_REFORMAT_FITS      256 \
  CUTWIN_SNTYPE 1 99
```

The last input `CUTWIN_SNTYPE` selects all spectroscopically-confirmed tags, and other `CUTWIN` options can be used as well. The output data folder contains no sim-truth which is useful for i) data challenges, and ii) test analysis codes to ensure they don't cheat using sim-truth.

12.4.3 Extract Spectra into TEXT Format

Below are spectral reformat options to re-write only the spectra (no photometry) to text tables, which can be used for spectroscopic analyzer codes or plotting. Each option writes a single text file per spectrum in the cwd (i.e., does not create an output folder); the calling code should organize a folder structure. For simulations, SIM_FLAM is automatically appended as last column.

```
OPT_REFORMAT_SPECTRA = 1 ! 3-column text file: lam Flam Flamerr
OPT_REFORMAT_SPECTRA = 2 ! write FLUX instead of FLam for astrodash
OPT_REFORMAT_SPECTRA = 4 ! write 'lam Flam Flamerr' using keyed format
OPT_REFORMAT_SPECTRA = 8 ! specialized output for SNID-SAGE code
```

Example to extract spectra for three CIDs:

```
snana.exe NOFILE \
  VERSION_PHOTOMETRY    MYSURVEY+gemini_SNIa \
  OPT_REFORMAT_SPECTRA  1   \
  SNCID_LIST 8,14,25
```

While there are no spectroscopic analysis codes in SNANA, the SNANA plot utility plot_table.py works on output with OPT_REFORMAT_SPECTRA=4.

12.4.4 Re-write Data Files with Flux Fudges

There are options in the analysis programs to modify the fluxes and flux-errors read from the data files. To re-write data files with these modifications,

```
 &SNLCINP
   VERSION_PHOTOMETRY = 'DES-SN3YR_DES'
   MAGCOR_FILE        = '$DES_ROOT/lcmerge/DES3YR_DES_MAGCOR.DAT'
   FLUXERRMODEL_FILE  = '$DES_ROOT/simlibs/DES3YR_FAKE_ERRORFUDGES.DAT'
   VERSION_REFORMAT_FITS = 'NEW_VERSION_NAME'  ! re-write in FITS format
&END
```

In this example, `MAGCOR_FILE` and `FLUXERRMODEL_FILE` are used to modify the fluxes and their uncertainties. `VERSION_REFORMAT_FITS` re-writes in FITS format with modified fluxes so that future analyses need not worry about using the `SNANA` flux-modify options. This option is particularly useful for data releases.

If the re-written data files include SNANA-computed flux/fluxErr fudges, a header variable in the data files is included,

```
MASK_FLUXCOR_SNANA:  <MASK>
```

where `MASK+=1` if the fluxes include a correction, and `MASK+=2` if the flux-errors have been correctd. `MASK=3` means that both the fluxes and uncertainties have been corrected. When analyzing the re-written data files, `MASK` is an instruction to ignore SNANA fudges, thus allowing the same input nameList file to be used on original and re-written data files. When analyzing re-written data files that already include a correction, an additional correction can be "FORCED" as follows,

```
MAGCOR_FILE  = '$DES_ROOT/lcmerge/DES3YR_DES_MAGCOR.DAT FORCE'
```

12.4.5 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.

12.4.6 1/Vmax Method: Post-Fit Calculations

The 1/Vmax method can be used to compute $Z_{\rm max}$ and the associated volumne for each SN. These calculation are controlled with the following `&FITINP` namelist options,

```
MAGLIM_Vmax = 'r 21.5  i 21.0'
OPT_Vmax    = 0                  ! valid options: 0,1,2,3
```

`MAGLIM_Vmax` defines a list of mag-limits and observer-frame filters to perform the 1/Vmax calculation. Only one mag-limit per filter can be defined, but a different mag-limit can be defined for each filter. `OPT_Vmax` is a bit mask as follows.

- Bit0 controls how the SN magnitude is computed: OFF is for the brightest observed epoch (using best-fit mag) and ON is for the peak magnitude.
- Bit1 controls how $Z_{\rm max}$ is computed: OFF is for the brightest observed epoch (using best-fit mag) and ON is for the peak magnitude.
- Bit7 (128) turns on verbose mode to see $Z_{\rm max}$ at each iteration.

For example, `OPT_Vmax`=0 means that both the SN magnitude and $Z_{\rm max}$ are computed at $T_{\rm rest}$ corresponding to the brightest observed epoch. To avoid picking an epcoh with an upward fluctuation, the best-fit fluxes are used to determine the brightest observed epoch rather than the observed fluxes. `OPT_Vmax`=3 means that both the SN magnitude and $Z_{\rm max}$ are computed at the best-fit epoch of peak brightness.

The following variables are automatically included in the output `FITRES` table file: $T_{\rm rest}$ where mag is computed, mag, $Z_{\rm max}$, volume.

12.4.7 FILTER_REPLACE

For filters that are very similar, such as multiple filters overlapping a patch of sky, it is sometimes convenient to replace filter names. As an example, consider the SDSS filters $ugrizUGRIZ$, where the $UGRIZ$ filters are defined for the small fraction of SN that land on overlapping CCD regions. To fit with the usual "`FILTLIST_FIT = 'ugriz'`" input, we can replace the upper-case filters with the following `&SNLCINP` namelist variable,

```
&SNLCINP
   FILTER_REPLACE = 'UGRIZ -> ugriz'
```

This replacement is equivalent to modifying the data files with `U->u`, `G->g`, etc.

WARNING(v10_28l): When there are no $UGRIZ$ bands, we expect that fitting $ugrizUGRIZ$ should give the same result as fitting $ugriz$ with the `FILTER_REPLACE` command above. This test works perfectly except when `OPT_COVAR`> 0; hence something about using a model-error covariance matrix with off-diagonal elements introduces a very small (few millimag) discrepancy. It is not yet clear if this discrepancy is due to a bug, or if it is expected when using `OPT_COVAR`.

12.4.8 SNTABLE_FILTER_REMAP

This option is for a survey with many similar bands (e.g., low-z), and is used to remap the output bands. For example, suppose we have the following bands: ab(U-like), cde(B-like), $fghi$(V-like). The '$fghi$' bands are all V-like, and correspond to very small changes in telescope transmission, such as from a filter replacement. The output tables by default would include filter-dependent values for each band: $abcdefghi$. To simplify some analyses, it may be more convenient to work with UBV rather than with $abcdefghi$. This convenience is achieved in the output tables by remapping the filters with the command

```
&SNLCINP
   SNTABLE_FILTER_REMAP = 'ab->U  cde->B  fghi->V'
```

This command only affects the output tables, and has no impact on selection cuts or fitting: i.e., all 9 bands ($abcdefghi$) are still used in the light curve fit.

12.4.9 Selecting SNe

The SNANA analysis programs (snana.exe, snlc_fit.exe, psnid.exe) include a common set of cuts via the &SNLCINP namelist. Unfortunately these cut parameters are documented only within the code, e.g.,

```
grep cutvar_name $SNANA_DIR/src/snana.F90 | grep CUTBIT
```

and for each CUTBIT_XXX there is an associated CUTWIN_XXX for &SNLCINP or command line override. There are other non-CUTWIN inputs to select based on CID or TYPE,

```
SNCID_LIST    = 4,88,105          ! explicit list of integer CIDs
SNCCID_LIST   = 'Joe,Henry,Sam'   ! same for char string names
SNTYPE_LIST   = 1,18              ! select ITYPE = 1 or 18
MXEVT_PROCESS = 10                ! process first 10 events
MXEVT_CUTS    = 4                 ! process until 4 events pass cuts
```

12.4.10 Selecting Observations

By default, all observations are selected. A subset of observations is selected with explicit CUTWIN_XXX inputs, and/or rejected via PHOTFLAG bits,

```
&SNLCINP
  CUTWIN_PSF   = 0.5,  2.75  ! FWHM, arcsec
  CUTWIN_ZPNPE = 30.5, 100.  ! cut on ZP in Npe

  PHOTFLAG_MSKREJ = 1016     ! reject if PHOTFLAG & 1016
      or
  PHOTFLAG_BITLIST_REJECT = 3,4,5,6,7,8,9  ! same as above (LSB=0)
```

If the PHOTFLAG_MSKREJ value is not stable (e.g., near start of analysis), the value can be included in the data [VERSION].IGNORE file as

```
# -------------
PHOTFLAG_MSKREJ: 1016    # reject if PHOTFLAG & 1016
    or
PHOTFLAG_BITLIST_REJECT: 3,4,5,6,7,8,9  # LSB=0

# and optional list of specific epochs to ignore
#         SNID    MJD      band
IGNORE:   1032  53668.387   g
IGNORE:   1286  53773.291   r
IGNORE:   1785  53792.898   i
# -------------
```

If PHOTFLAG_MSKREJ is defined in the &SNLCINP name-list or the command line, this value overrides the value in [VERSION].IGNORE.

12.4.11 Selecting Early Epochs

During a survey, light curves are fit with only a few epochs for monitoring or spectroscopic target selection. For testing on simulated data, however, the entire light curve exists because it is not practical to re-generate simulations that stop at each successive night. The following &SNLCNINP inputs illustrate how to fit light curves (data or sim) using only a few epochs as if the latter epochs had not yet been observed:

```
&SNLCINP
   EARLYLC_STRING = 'MAXOBS 4  FILTERS riz  SNRMIN  4.5'     !1
      or
   EARLYLC_STRING = 'MAXOBS 4  FILTERS riz  PHOTPROBMIN .5'  !2
      or
   EARLYLC_STRING = 'MAXOBS 4  FILTERS riz  PHOTMASK 4096'   !3
      or
   EARLYLC_STRING = 'MAXNIGHT 2  FILTERS riz  PHOTMASK 4096' !4
      or
   EARLYLC_STRING = 'MAXNIGHT 2  PHOTMASK 4096  NDAYADD 20'  !5
      or
   EARLYLC_STRING = 'MAXNIGHT 2  PHOTMASK_START 4096'        !6a
      or
   EARLYLC_STRING = 'MAXOBS 3  SNR_START 5'                  !6b
```

The MAXOBS key specifies the maximum number of EARLYC observations to keep that satisfy the logical AND of cuts on filters, signal-to-noise (SNRMIN), photometry-probability (PHOTPROBMIN) and photometry-mask (PHOTMASK). The first example above (!1) keeps epochs until there are 4 observations in r,i, or z that have SNR above 4.5. In the second example (!2), the SNR requirement is replaced with the PHOTPRPOBMIN requirement. Third example (!3) requires four observations to have the 4096 bit of the photometry mask (e.g., detection). Epochs satisfying the EARLYLC requirements may not occur until well after explosion; thus all pre-EARLYLC (pre-explosion) epochs are kept. Once MAXOBS=4 EARLYLC epochs are found, no additional epochs are kept.

The MAXNIGHT key (!4) includes all observations within a 12 hour period, or 1 night. For example, consider a night with observations in the g, r, i, z bands, and they all satisfy the "PHOTMASK & 4096" requirement. In the following night we have i, z observations which pass PHOTMASK. While this counts as 6 observations toward MAXOBS, it counts as 2 nights towards MAXNIGHT. Once the MAXNIGHT requirement is met with the first i-observation in the 2nd night, all observations in the 2nd night are included.

NDAYADD (!5) will add epochs for this many days after the last observation from the above keys.

Finally, PHOTMASK_START and SNR_START (!6a,6b) are requirements on when to start counting nights (for MAXNIGHT) or observations (for MAXOBS). In 6a, a single observation satisfying "PHOTMASK_START & 4096" starts the number-of-nights counter with no further requirements on successive observations. Similarly in 6b, a single observation with SNR>5 starts the number-of-observation counter. This feature is useful for rapidly fading transients that may be detected on only 1 night, but still have useful observations in successive nights.

The default cuts are wide open, except for the default NDAYADD=0. Not specifying FILTERS will count all filters, not specifying PHOTPROBMIN will ignore PHOTPROB in counting early epochs, etc. Thus setting EARLYLC_STRING='MAXOBS 4' will select the first 4 observations in the data file, regardless of what is measured.

To avoid mistakenly using the entire light curve in some part of an analysis, the removed epochs are not stored in any arrays so that the internal arrays are filled as if the late epochs did not exist.

12.4.12 Selecting Epoch Ranges for Fast Transients (REQUIRE_EPOCHS_STRING)

To help select transients whose duration is faster than Supernova, the time above threshold is a useful quantity that can be used with the following nameList input:

```
&SNLCINP
   REQUIRE_EPOCHS_STRING 'FILTERS riz SNRMIN 5 TOBS_RANGES 14 20 25'
```

SNRMIN defines the threshold. The three TOBS_RANGES values specify time-window 1) before threshold is satisfied, 2) while SNR is above threshold, and 3) after SNR falls below threshold. More specifically, events are selected whose time above SNRMIN is less than 20 days. To avoid getting fooled at the start or end of a survey, an observation is required within 14 days prior to the first epoch with SNR>5, and another observation is required within 25 days after the last epoch with SNR>5. The pre- and post observation ensure that an above-threshold epoch would have been seen, and was not hidden by a dormant survey.

12.4.13 Rejecting Outlier Fluxes for Fakes and SIM

If fakes overlaid on images have catastrophic outliers, outlier epochs can be rejected with

```
&SNLCINP
   CUTWIN_SIMPULL -3.0, +3.0  ! reject >3 sigma outliers
```

12.4.14 Miscellaneous &SNLCINP Options

```
&SNLCINP
   USESIM_SNCC = F    ! ignore simulated CC  (fit true Ia only)
       or
   USESIM_SNIA = F    ! ignore simulated IA (fit true CC only)

   SIM_PRESCALE = 4.0 ! fit 1/4 of sim sample, but still fit all data
```

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 use 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 (§9.6), 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/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.

12.6 Reading gzip'ed Input Files

To reduce disk usage and improve the speed in reading large input text files, SNANA will read any input file that is compressed with gzip. Always specify the uncompressed name (e.g., mystuff.text) and SNANA programs will check for both mystuff.text and mystuff.text.gz. If both the compressed and uncompressed files exist, the code will abort. Reading compressed files is faster, with the cost of a little added time ($\sim$ 0.1 sec) opening the file with popen instead of fopen. Gzipped inputs include the SIMLIB file (§4.7), HOSTLIB file (§4.23), SIMSED model files (§9.6), LCLIB model files, SALT2 model files, and text-formatted data files.

Be aware that two of these input files (SIMLIB & LCLIB) are rewound each time the EOF is reached. The "popen" command for gzipped files is not compatible with the "rewind" function, and thus rewinding a gzipped file requires an explcit close and re-open. For small input files that are re-read many times, the extra close-and-open overhead from a gzipped file may increase processing time. Thus it is recommended to gzip *only* the large unput files, and leave the small input files unzipped.

12.7 Program Return Codes

Program return codes are useful for higher-level pipelines. Each C and fortran program returns 0 on sucessfull completion. An abort or error results in the following error codes that can be examined with unix command 'echo $?' :

Program	return error code
kcor.exe	10
snlc_sim.exe	11
SALT2mu.exe	12
combine_fitres.exe	13
sntable_dump.exe	14
wfit.exe	15
merge_root.exe	16
merge_hbook.exe	17
snana.exe	21
snlc_fit.exe	22
psnid.exe	23

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. 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 SNDATA_ROOT tarball is updated less frequently; README_UPDATES will indicate if and why a new SNDATA_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.**

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] K. S. Mandel et al. A hierarchical Bayesian SED model for Type Ia supernovae in the optical to near-infrared. *MNRAS*, 510(3):3939–3966, March 2022.

[4] C. R. Burns et al. The Carnegie Supernova Project: Light-curve Fitting with SNooPy. *AJ*, 141:19, January 2011.

[5] G. Goldhaber et al. Timescale Stretch Parameterization of Type Ia Supernova B-band Light Curves. *Astrophys. J.*, 558:359–368, 2001.

[6] 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.

[7] R. Kessler et al. Results from the Supernova Photometric Classification Challenge. *PASP*, 122:1415–1431, December 2010.

[8] R. Kessler et al. `SNANA`: A public software package for supernova analysis. *PASP*, 121:1028, 2009.

[9] P. Nugent et al. K-Corrections and Extinction Corrections for Type Ia Supernovae. *PASP*, 114:803, 2002.

[10] J. Carretero et al. An algorithm to build mock galaxy catalogues using MICE simulations. *MNRAS*, 447:646–670, February 2015.

[11] V. Marra, M. Quartin, and L. Amendola. Accurate weak lensing of standard candles. I. Flexible cosmological fits. *Phys. Rev. D*, 88(6):063004, September 2013.

[12] 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.

[13] E. F. Schlafly and D. P. Finkbeiner. Measuring Reddening with Sloan Digital Sky Survey Stellar Spectra and Recalibrating SFD. *Astrophys. J.*, 737(2):103, August 2011.

[14] E. L. Fitzpatrick. Correcting for the Effects of Interstellar Extinction. *PASP*, 111(755):63–75, January 1999.

[15] E. L. Fitzpatrick. Interstellar Extinction in the Milky Way Galaxy. In A. N. Witt, G. C. Clayton, and B. T. Draine, editors, *Astrophysics of Dust*, volume 309 of *ASP Conf. Series*, page 33, May 2004.

[16] J. Maíz Apellániz et al. The VLT-FLAMES Tarantula Survey. XVI. The optical and NIR extinction laws in 30 Doradus and the photometric determination of the effective temperatures of OB stars. *A&A*, 564:A63, April 2014.

[17] E. L. Fitzpatrick et al. An Analysis of the Shapes of Interstellar Extinction Curves. VII. Milky Way Spectrophotometric Optical-through-ultraviolet Extinction and Its R-dependence. *Astrophys. J.*, 886(2):108, December 2019.

[18] K. D. Gordon et al. One Relation for All Wavelengths: The Far-ultraviolet to Mid-infrared Milky Way Spectroscopic R(V)-dependent Dust Extinction Relationship. *Astrophys. J.*, 950(2):86, June 2023.

[19] K. D. Gordon et al. The Panchromatic Hubble Andromeda Treasury. XV. The BEAST: Bayesian Extinction and Stellar Tool. *Astrophys. J.*, 826(2):104, August 2016.

[20] K. D. Gordon et al. A Quantitative Comparison of the Small Magellanic Cloud, Large Magellanic Cloud, and Milky Way Ultraviolet to Near-Infrared Extinction Curves. *Astrophys. J.*, 594(1):279–293, September 2003.

[21] K. Gordon. dust_extinction: Interstellar Dust Extinction Models. *J. Open Source Software*, 9(100):7023, August 2024.

[22] A. Goobar. Low R_V from Circumstellar Dust around Supernovae. *Astrophys. J.*, 686(2):L103–L106, October 2008.

[23] R. Amanullah et al. Diversity in extinction laws of Type Ia supernovae measured between 0.2 and 2 μm. *MNRAS*, 453(3):3300–3328, November 2015.

[24] J. C. Weingartner and B. T. Draine. Dust Grain-Size Distributions and Extinction in the Milky Way, Large Magellanic Cloud, and Small Magellanic Cloud. *Astrophys. J.*, 548(1):296–309, February 2001.

[25] B. T. Draine. Scattering by Interstellar Dust Grains. I. Optical and Ultraviolet. *Astrophys. J.*, 598(2):1017–1025, December 2003.

[26] L. Sommovigo et al. Learning the Universe: physically-motivated priors for dust attenuation curves. *pre-print*, arXiv:2502.13240, February 2025.

[27] Y. C. Pei. Interstellar Dust from the Milky Way to the Magellanic Clouds. *Astrophys. J.*, 395:130–139, August 1992.

[28] A. Li et al. On Dust Extinction of Gamma-Ray Burst Host Galaxies. *Astrophys. J.*, 685(2):1046–1051, October 2008.

[29] M. Sako et al. Photometric Type Ia Supernova Candidates from the Three-year SDSS-II SN Survey Data. *Astrophys. J.*, 738:162, September 2011.

[30] S. Thorp et al. Testing the consistency of dust laws in SN Ia host galaxies: a BAYESN examination of Foundation DR1. *MNRAS*, 508(3):4310–4331, December 2021.

[31] M. Grayling et al. Scalable hierarchical BayeSN inference: investigating dependence of SN Ia host galaxy dust properties on stellar mass and redshift. *MNRAS*, 531(1):953–976, June 2024.

[32] E. Y. Hsiao et al. K-Corrections and Spectral Templates of Type Ia Supernovae. *Astrophys. J.*, 663(2):1187–1200, July 2007.

[33] J. Marriner et al. A More General Model for the Intrinsic Scatter in Type Ia Supernova Distance Moduli. *Astrophys. J.*, 740:72, October 2011.

[34] R. Kessler and D. Scolnic. Correcting Type Ia Supernova Distances for Selection Biases and Contamination in Photometrically Identified Samples. *Astrophys. J.*, 836:56, February 2017.

[35] L. Hui and P. B. Greene. Correlated Fluctuations in Luminosity Distance and the Importance of Peculiar Motion in Supernova Surveys. *Phys. Rev. D*, 73(12):123526, 2006.

[36] M. Goliath et al. Supernovae and the Nature of the Dark Energy. *A&A*, 380:6–18, 2001.