PUBLICATIONS OF THE ASTRONOMICAL SOCIETY OF THE PACIFIC, **123**:237–248, 2011 February

SYNAPPS: Data-Driven Analysis for Supernova Spectroscopy

R. C. THOMAS,[1] P. E. NUGENT,[1] AND J. C. MEZA[2]

Received 2010 October 29; accepted 2010 December 14; published 2011 February 10

ABSTRACT. We introduce a new computer program, SYNAPPS, for forward-modeling of supernova (SN) spectroscopy data sets. SYNAPPS is a spectrum fitter embedding a highly parameterized synthetic SN spectrum calculation within a parallel asynchronous optimizer. This open-source code is primarily aimed at the problem of systematically interpreting large sets of SN spectroscopy data. While SYNAPPS should be useful to current SN spectroscopy efforts like the Nearby Supernova Factory, Lick Observatory Supernova Search, Palomar Transient Factory, Harvard Center for Astrophysics SN program, and so on, it could also benefit future similar efforts connected to the Dark Energy Survey, Panoramic Survey Telescope and Rapid Response System, or the Large Synoptic Survey Telescope. Smaller programs are also potential users. SYNAPPS illustrates the potential for data-driven discovery enabled by high-performance computing, where complex physical systems are directly constrained by large information-rich sets of scientific observations. Here, we discuss the motivation of our approach, outline the structure of the code, present some example calculations, and describe a few enhancements in terms of physics modeling, optimization, and computing that we will be pursuing for the future.

Online material: color figures

1. INTRODUCTION

A rough physical description of a supernova (SN) applicable from a few days to months after explosion (phases easily accessible to observation) is an optically thick, continuum-emitting pseudophotosphere surrounded by an extended line-forming region (see, e.g., Branch et al. 2003 for a detailed review). Hydrodynamical forces are important in the first moments after ignition, but the rapid unbinding of the progenitor puts the stellar ejecta into homologous free expansion. As the SN expands, the density of the gas drops as t^{-3} and the optically thick region shrinks. Photons originating in the decay of freshly synthesized radioactive nuclei and energy deposited by shocks are reprocessed by the SN atmosphere. The elemental abundances, densities, and temperature structures in the ejecta influence the emerging spectrum. As the SN evolves, its spectral energy distribution changes, reflecting the effect of newly exposed layers of SN ejecta. The spectral features are line blends, broad overlapping absorption and emission features that form from complex nonlinear radiative transfer processes. A set of SN spectral time series encodes a picture like a computed tomography (CT) scan (Fig. 1). With radiative transfer calculations to interpret them, spectral time series reveal the structure of the ejecta, telling the story of the explosion and hinting at the history of the pre-SN progenitor star.

In the SN field, radiative transfer calculations are the interface through which observations confront theoretical predictions. Computed photometry and spectra from detailed theoretical high-performance computing (HPC) SN simulations are compared with multiwavelength time-series observations to verify or invalidate proposed explosion models. Many other astrophysical problems are linked in one way or another to our HPC-enabled understanding of SN explosions, such as the life cycles of stars and the interstellar medium and the formation of compact objects. One notable example is SN cosmology. Observations of Type Ia SNe (SNe Ia) led to the discovery that the universe is not only expanding as Hubble (1929) found, but that the rate of expansion itself is *accelerating* (Riess et al. 1998; Perlmutter et al. 1999). The need to better understand SNe Ia for future cosmology experiments is a key driver for the use of HPC to construct ever-more-realistic numerical SN models and test them against high-statistics, high-dimensionality data sets.

Grids of model SN explosions sampling initial and attendant conditions such as progenitor metallicity, distribution of ignition points, envelope mass, and so on have been constructed in the literature with varying levels of detail (e.g., 2D SN Ia explosion model grids have just recently been calculated by Kasen et al. 2009). Often, models rely on simplified nucleosynthesis reaction networks and do not always include detailed radiation transfer postprocessing. Extreme-scale computing may one day close this gap, enabling the construction of detailed

[1] Computational Cosmology Center, Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road Mail Stop 50B4206, Berkeley, CA 94720-8151.

[2] Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road Mail Stop 50B4230, Berkeley, CA 94720-8150.

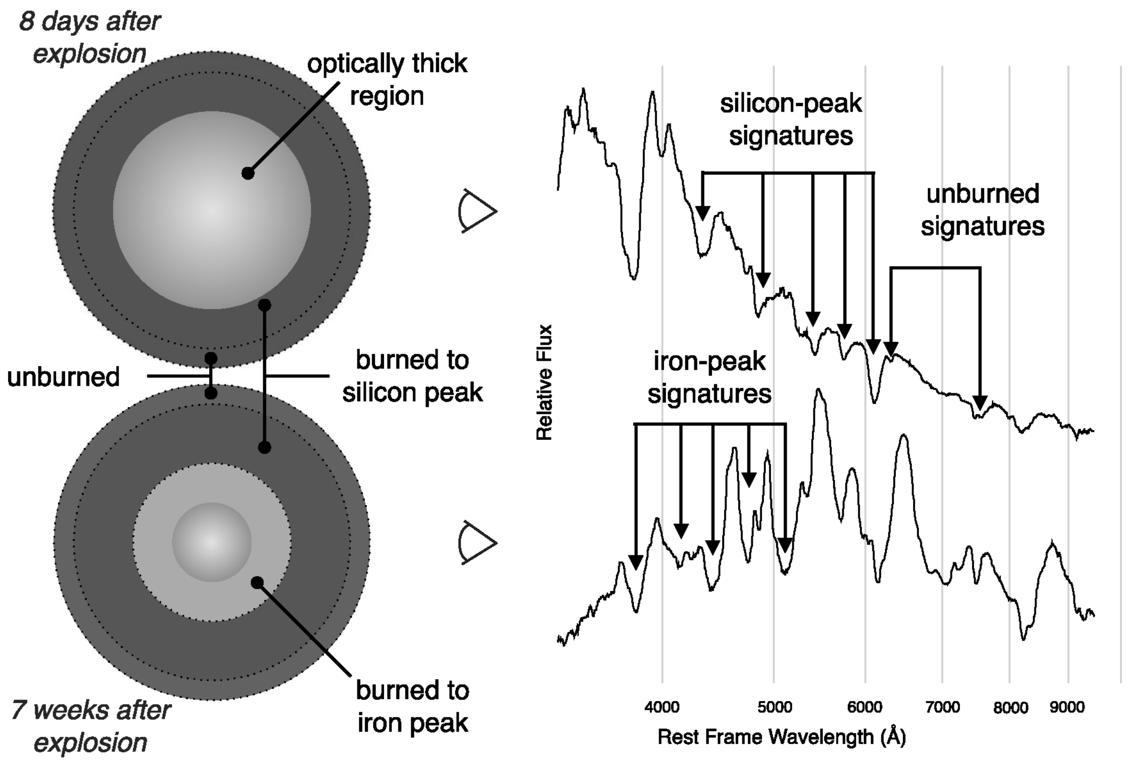

FIG. 1.—SN spectroscopy as a scan of a stellar explosion. The reprocessing of deposited energy by the ejecta imprints signatures that appear as spectroscopic features. These features (strong absorptions, indicated by arrows, and separated by emission humps) encode atomic abundances, their distributions in velocity space, temperature structure, and ejecta density. As the optically thick core region recedes deeper into the ejecta, inner layers are exposed and influence the appearance of the spectrum, as depicted on the right. A week after explosion, the SN Ia is optically thick up to the outer layers, which are rich in silicon peak and unburned elements. Later, more completely burned products (iron peak) are exposed and change the character of the spectrum markedly from blue to red. As the density drops, the outer layers also become more optically thin, and the signatures previously apparent at early times weaken and the SN fades. See the electronic edition of the *PASP* for a color version of this figure.

petabyte model data sets originating at first principles through multidimensional time-dependent hydrodynamics and radiative transfer simulations. The hope is that within these model data sets is the right CT scan for every SN, or at least something very close. Getting to that point has been, and will continue to be, a slow process.

On the other hand, constraining explosion models from data-driven SN analysis will narrow down the range of model grids so that future HPC simulations can be effectively focused at the right questions. For much of the history of the study of SN spectroscopy, fitting parameterized synthetic spectra to observations has produced physics requirements for successful HPC explosion simulations. This approach is the context of this article, describing work that developed from the question of how to systematically interpret giant samples of SN Ia spectroscopy like the forthcoming time-series spectrophotometry set from the Nearby Supernova Factory (SNfactory; Aldering et al. 2002). SNe Ia are the original problem domain, but our results are applicable to other types of SNe. Also, new wide-field surveys such as the Palomar Transient Factory (Rau et al. 2009) are uncovering exotic classes of explosive stellar transients for which no established chain of modeling literature yet exists. The need to physically account for these new objects is justification enough for developing and using parameterized spectrum synthesis codes instead of waiting for ab initio models to catch up.

A familiar example of such a code is SYNOW (Synthesis Now; Fisher 2000; Branch et al. 2009). SYNOW treats line-blending explicitly, enabling inferences about the presence or absence of atomic species at various ejection velocity intervals and their ionization states. These results play an important role in constraining more detailed explosion models. Many of the inferences are tentative until confirmed by more detailed codes, but the idea is to point those efforts in the right direction—guidance that is especially indispensable for understanding unusual new objects.

SYNOW and similar codes are interactive and run iteratively—a reasonable design when the amount of data to examine is relatively small. Users adjust approximately 50 input parameters and, in about a minute on a desktop or laptop, can compare a synthesized spectrum with an observed one. Investigation proceeds in an exploratory manner as the user makes adjustments needed to identify lines and obtain a good fit. The ability to explore is a strength of the approach; as ions can be individually switched on or off, a user can see how a spectrum is put together. Line identifications are more confident when multiple ion signatures can be fit simultaneously, and fitting a feature at one wavelength results in predictions for lines from the same parent ion at others (fits in the optical predict features in the ultraviolet or infrared). Users are guided by experience and the accumulated wisdom of previous generations of experts (e.g., Hatano et al. 1999). Experienced users can make sense of a spectrum, through "chi-by-eye," on timescales of hours to days through iterative fitting.

In the face of newer, larger, and more accurate SN spectroscopic data sets, the prospect of performing hundreds of tedious fits to thousands of spectra by human trial and error is more than just daunting—it seems to be the wrong approach. Given the volume of existing spectroscopic data sets, such as those accumulated by the Harvard Center for Astrophysics (Matheson et al. 2008), the Lick Observatory Supernova Survey (Silverman et al. 2011, in preparation; Ganeshalingam et al. 2010), and, in particular, the SNfactory, we have undertaken the task of designing and implementing an automated direct spectroscopic analysis code.

This article describes a new code, SYNAPPS, for the automated analysis of SN spectra.[3] SYNAPPS is an application combining a highly parameterized model SN atmosphere calculation with a general parallel numerical optimization framework, facilitating direct spectroscopic SN analysis. This combination relieves the user from tedious iterative adjustment of a large number of parameters to make gains in fit agreement and ensures more exhaustive searching of the parameter space. This article is organized into five sections. In § 2 we outline our approach and describe the code. Section 3 includes sample calculation results. Section 4 contains detailed instructions on running SYNAPPS and its companion code SYN++. Section 5 concludes this article, discussing directions for future development.

2. TECHNIQUE

To solve the SYNOW automation problem, we cast the problem of fitting a SN spectrum as a constrained nonlinear optimization problem. Bound constraints, linear inequality constraints, and linear equality constraints are all in play. The basic optimization problem is of the standard form:

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) \\ &\text{subject to } \boldsymbol{c_L} \leq \boldsymbol{A_I x} \leq \boldsymbol{c_U} \\ &\qquad\qquad \boldsymbol{A_E x} = \boldsymbol{b} \\ &\qquad\qquad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned} \tag{1}$$

[3]SYNAPPS is available to anyone under the terms of the BSD license as a part of the ES software package at http://c3.lbl.gov/es.

The function $f(\boldsymbol{x})$ is a scalar objective function of the parameterization input vector $\boldsymbol{x}$ and measures the difference between a synthetic spectrum and an observed one. Components of $\boldsymbol{x}$ are the ingredients of a simplified model SN atmosphere: velocities at the lower and upper boundaries of the line-forming region, reference values and gradients for opacity profiles, ion excitation temperatures, etc. Linear inequality constraints (the matrix $\boldsymbol{A_I}$) may be used, for example, to confine iron opacity to ejecta velocities below a floating cutoff that also serves as the minimum velocity for some other species. Equality constraints (matrix $\boldsymbol{A_E}$) are used to control what is called "detaching" ions in SYNOW parlance: If the constraint is enforced, then the minimum ejection velocity for an opacity profile is pegged to the photospheric velocity, but if deactivated is allowed to float (i.e., the opacity profile is "detached" from the photosphere). Bound constraints (vectors $\boldsymbol{l}$ and $\boldsymbol{u}$) provide a simple means to restrict values to allowable ranges of interest. The vectors $\boldsymbol{c_L}$, $\boldsymbol{c_u}$, and $\boldsymbol{b}$ define constraint surfaces in the parameter space.

The approach seems straightforward, but in practice, there are a number of design challenges. Analytical derivatives with respect to the inputs of the objective function are not readily calculable, and different numbers of parameters (entire ions) may be used at different times. Though an L^2-norm objective function is attractive from maximum-likelihood considerations, the option of using an L^1-norm instead to avoid overpenalizing bad fits to spurious data or certain shortcomings of the parameterization may be desirable. This choice could even be user-defined and quite general. Setup costs like atomic data management and memory allocation should be amortized over large numbers of objective function evaluations. We also recognized the utility of focusing exploratory spectrum synthesis on individual ions, features, or wavelength regions of interest, so we considered this capability a design requirement.

We have reimplemented SYNOW as our fitting function in modern C++ with some modifications and enhancements. Our immediate aim is to keep the physics assumptions, basic user interface, and workflow as familiar as possible for SYNOW users—all of the usual SYNOW assumptions and caveats apply. Line opacity is parameterized spatially (e.g., an exponential falloff with velocity) and as a function of wavelength with line relative strengths set by Boltzmann excitation temperatures. Line transfer is computed under the Sobolev approximation (e.g., Rybicki & Hummer 1978; Jeffery & Branch 1990) and assuming a pure resonance-scattering source function. The photosphere is simulated as a sharply defined blackbody continuum-emitting surface. One difference from SYNOW is that since there is no remote self-coupling for a given wavelength bin in the computation of line source functions, they can be computed using OpenMP loop-level parallelism over radius and angle. This yields a small performance boost, limited by the barrier at the end of each wavelength step: red line source functions depend on bluer ones. An individual calculation takes less than a minute on a typical desktop or laptop computer, depending on the number of atomic lines needed.

The spectrum synthesis calculation may be run as a single-shot standalone executable (SYN++) or may be linked into another application through a library interface. This library interface is the means by which the calculation can be coupled to an optimization framework and (using the notation of eq. [1]) is designed to take a parameterization input vector $\boldsymbol{x}$ and return the objective function value $f(\boldsymbol{x})$ computed by comparing the synthesized spectrum with an observed one.

The lack of analytic derivatives for the automation component of our problem made something conceptually similar to the derivative-free Nelder & Mead (1965) search attractive. Variations in the complexity of candidate evaluations could lead to latency if they are launched in parallel, so we sought an asynchronous search method. After some experimentation, the APPSPACK (asynchronous parallel pattern search; Kolda 2005; Gray & Kolda 2005; Griffin & Kolda 2006) optimizer package was selected. APPSPACK operates a parallel generating set search to help solve generic problems with upward of 100 variables on distributed architectures. It is a fault-tolerant framework that is especially suitable to problems where analytic derivatives are unavailable or the objective function is noisy. Using APPSPACK, our new code, SYNAPPS, implements a master-worker scheme—the master CPU generates trial points $\boldsymbol{x}$ in the parameter space and communicates them to the worker CPUs, which in turn synthesize the resulting spectra, compute the objective function values $f(\boldsymbol{x})$, and report them back to the master. Depending on criteria such as sufficient decrease in the objective function, step size, contraction factor, etc., the master CPU evaluates whether the trial point is an improved fit and shifts, contracts, or otherwise adjusts the distribution of trial points accordingly. The master CPU workload is light compared with that of the workers, so no serial bottleneck through the master CPU has been observed. APPSPACK manages the trial-point list and communication between the master and workers and keeps a cache of evaluated points for later reference, in lieu of recomputation. In practice, we observe that about 10% of evaluations are recycled from the cache, and the cache itself can be reused by subsequent runs in many cases. Communication between APPSPACK-managed components uses the Message-Passing Interface (MPI); so since individual workers utilize OpenMP for the objective function, SYNAPPS is a multilevel parallel code. The general architecture is depicted in Figure 2.

The objective function is based on the Euclidean distance between the synthesized spectrum and observed target spectrum, weighted by the flux uncertainties on the observed spectrum. To overcome some of the systematics introduced by the approximate radiative transfer technique (for instance, due to the treatment of the lower boundary as a sharp photosphere) and also any low-frequency calibration artifacts in the data, a low-order polynomial is multiplied against the synthesized

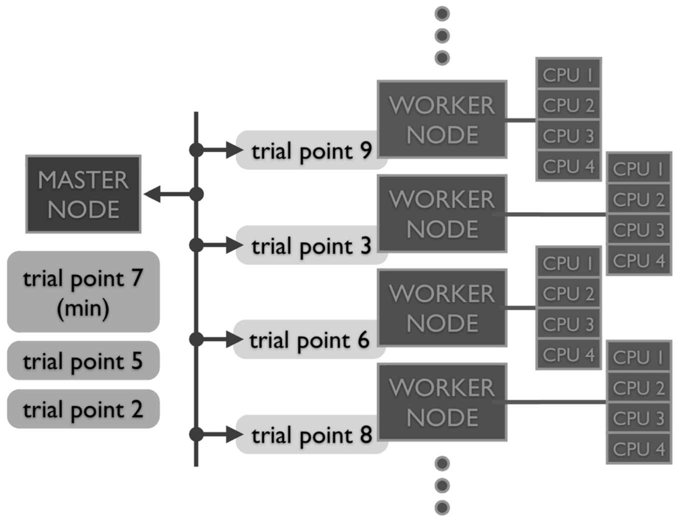

FIG. 2.—General architecture of the SYNAPPS program. The master node, under APPSPACK control, maintains a queue of unevaluated trail points and exchanges them with worker nodes for function values. This level of communication is asynchronous and managed through MPI. Each worker node runs a synthetic spectrum calculation, using OpenMP loop-level parallelism to speed up the calculation of line source functions. See the electronic edition of the *PASP* for a color version of this figure.

spectrum when it is compared with the observed spectrum. Coefficients of the polynomial are themselves included as fit parameters. There are goodness-of-fit measures engineered for comparing SN spectra that could be used instead (DIFF1 and DIFF2; Jeffery et al. 2007), but these do not readily incorporate flux uncertainty information and, moreover, we prefer the well-defined polynomial fit coefficients. In addition, telluric features or parts of the spectrum to be ignored by the fit may be masked out. Given the well-defined restricted goals (feature identification and measuring ejecta velocity intervals for ions) and model systematics, this simple warping and masking are adequate and are merely an attempt to robustly account for the filtering that SYNOW users mentally apply when interactively fitting spectra.

SYNAPPS has been tested and used on a number of architectures of different sizes. A typical SYNAPPS run to fit a spectrum using a 16-core AMD Opteron Linux cluster without OpenMP activated (each objective function evaluation takes longer, but more are evaluated at once) takes about 12–18 wall-clock hours or less, depending on initial guess values. This performance is probably attractive to researchers and laboratories with access to such clusters and moderate amounts of data. Larger systems, such as the Cray XT4 supercomputer Franklin at the National Energy Research Scientific Computing Center (NERSC) allow SYNAPPS to scale up to its full potential of a few hundred cores per spectrum and return results in a half-hour or hour. Scaling (in terms of number of evaluations per unit time) is linear up to a number of workers equal to twice the problem dimensionality and then levels off—the

APPSPACK generating set search cannot take advantage of more workers (see § 5). Users other than the developer have also used SYNAPPS on NERSC machines.

3. SAMPLE CALCULATIONS

Figure 3 contains an array of SYNAPPS fits to some preliminary data from the SNfactory performed with the Franklin supercomputer at NERSC. In fact, this figure may represent the largest number of forward-modeling fits to individual SNe Ia anywhere in the literature, and it includes only a fraction of SNfactory SN Ia spectra obtained within 2.5 days of maximum brightness. Most of the major ion features are faithfully reproduced. Spectra sampling a range of signal-to-noise ratios (S/N), SN Ia luminosity class (less luminous than normal, to overluminous, and a possible super-Chandrasekhar-mass SN Ia), and spectroscopic subtype are represented, all fit with relative ease, but with different ions and ion strengths, as necessary. When completed, SNfactory's full analysis of its more than 2000 spectra will extend from the earliest available epochs (1–2 weeks before maximum) to around 40 days past maximum; the radiative transfer technique is most reliable across this window. The ability to scale to derive constraints on explosion models from such large detailed data sets as from the SNfactory means that we can finally conduct statistical, cross-sectional, and longitudinal studies of the relationship of spectral-feature evolution to other observables like color and luminosity and, especially, that we can produce constraints for detailed explosion models.

One capability we have carried forward from SYNOW is the ability to "take apart" SN spectra with SYNAPPS, demonstrated by the fits to the S II W blend in Figure 4. Faithfully fitting this feature with SYNOW requires tedious iterative fine-tuning, but SYNAPPS eliminates the need to do this manually. Here, we have performed fits to a restricted portion of the spectra with just one ion to see where it contributes to features at other wavelengths. Such exploration allows us to identify which ions influence the formation of the spectrum within blends, predict features elsewhere to look for, and more confidently make line identifications.

SYNAPPS has recently been used to interpret several unusual SNe and to place constraints on models of novel transient phenomena. SYNAPPS has been used to study the spectrum of SN 2007if at peak brightness—this peculiar SN Ia shows clear spectroscopic signatures indicating a super-Chandrasekhar mass progenitor, possibly from the coalescence of two white dwarfs (Scalzo et al. 2010). SYNAPPS was instrumental in the identification of SN 2007bi as a pair-instability SN, the product of an extremely massive (~100 $M_\odot$) progenitor star (Gal-Yam et al. 2009). Studies of other unusual stellar explosions discovered by the Nearby Supernova Factory and Palomar Transient Factory are underway.

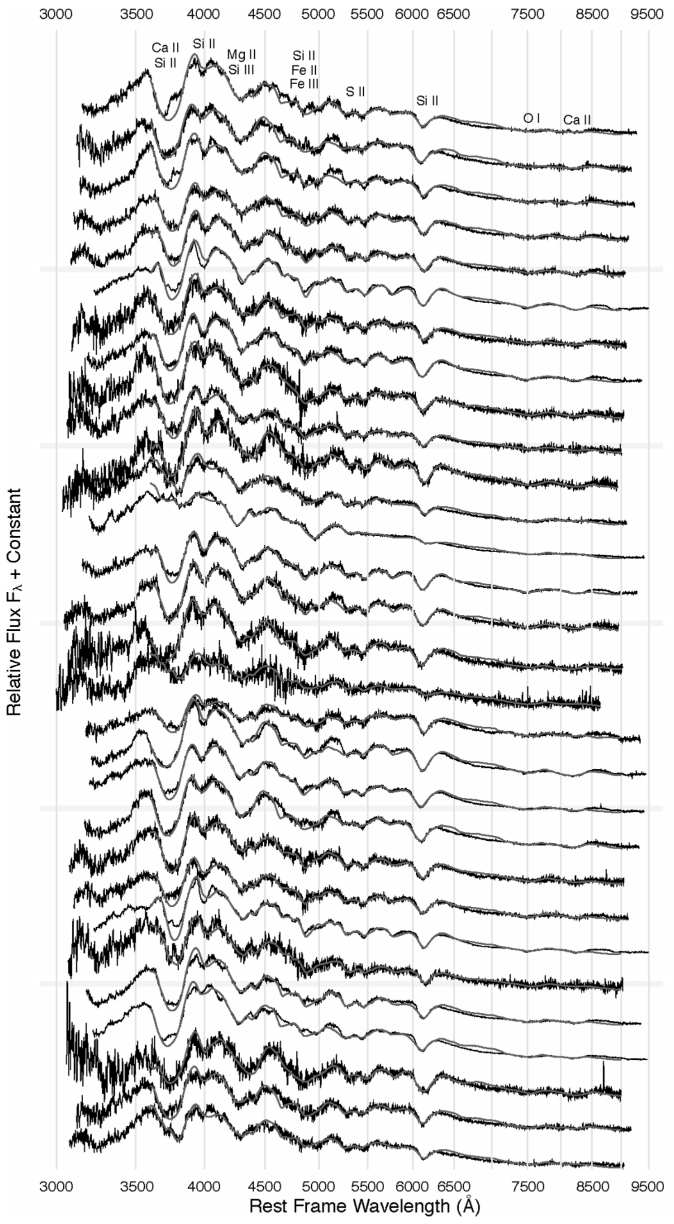

FIG. 3.—Result of automated analysis of several SN Ia spectra from the SNfactory around maximum light with the SYNAPPS code. A range of S/N and SN Ia luminosity and spectroscopic subclasses are represented. Parameterized model SN Ia atmospheres are fit to the observed data (dark curve) by comparing them with synthetic spectra (light/red). These models can be compared across the entire sample and across phase to trace out developmental similarities and differences within the population. See the electronic edition of the *PASP* for a color version of this figure.

4. USING THE PROGRAMS

We have released the SYNAPPS and SYN++ source code together in a package called "ES: Elementary Supernova Spectrum Synthesis" under the BSD open-source license. In our design we have strived to ensure that both individual researchers

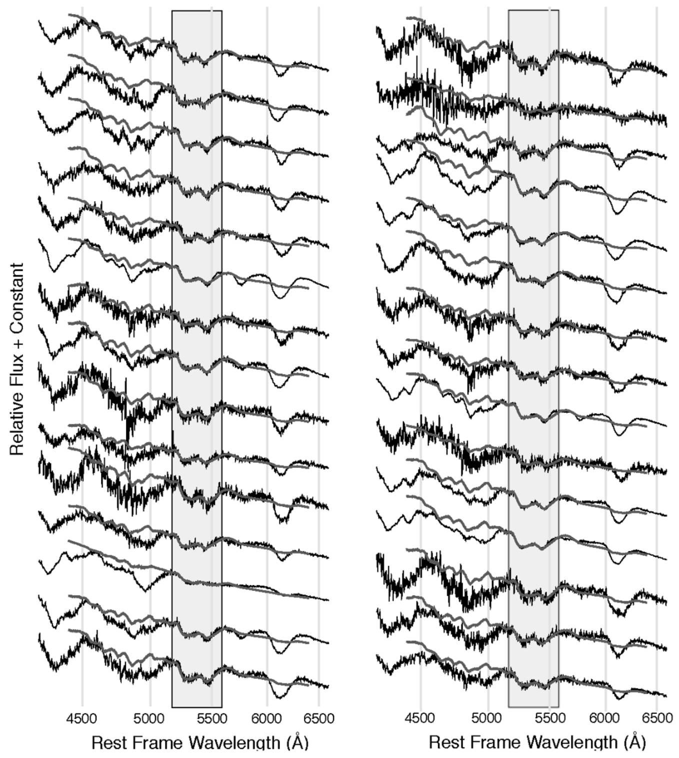

FIG. 4.—Single-ion fits to SNfactory spectra with S II only. SYNAPPS can be constrained to fit individual features or sections of spectra to allow prediction of features at other wavelengths; SYNOW can accomplish the same task, but only through lengthy manual iteration. Here, the S II "W" feature is fit (boxed region), but the synthetic spectrum at other wavelengths predicts where other S II features influence the formation of the spectrum. In the higher S/N spectra, small notches predicted by SYNAPPS around 4600 Å due to S II are easily visible. See the electronic edition of the *PASP* for a color version of this figure.

with small data sets and projects with larger ones find the codes easy to work with. ES is equipped with a familiar GNU autotools build system (`configure`, `make`) that has been tested on a variety of Linux and Mac OS X systems of varying sizes and types. Our intent is that SYNAPPS be useful not only for our own research, but for that of others as well.

There is one minimum external software dependency that must be satisfied to build both programs—atomic-line-list data are stored in binary FITS files, so both programs must link against the CFITSIO[4] library. In addition, to build SYNAPPS, BLAS,[5] LAPACK,[6] and APPSPACK[7] software dependencies must be satisfied. We strongly recommend installing the version of APPSPACK designated specifically for linking with SYNAPPS (version 5.0.1-C3), as it has a more robust build system than the others. The dependencies may be detected automatically by `configure`, but some or all may need to be provided on the `configure` command line. If no dependency other than CFITSIO is satisfied, then only SYN++ will be built. Enabling OpenMP to accelerate line source-function calculations is optional for both codes, but building SYNAPPS requires an MPI compiler (e.g., `mpicxx`).

A typical workflow with SYNAPPS and SYN++ can range from highly exploratory to routine pipelined fittings. An exploratory fit begins with a few initial passes with one or a few ions applied to a subset of wavelengths focused on any better-known features. The results of these fits can be combined and more ions and opacity components can be added until the major features are captured in the fit. The input files for both scripts are designed to ease the activation/deactivation of ions and opacity components for comparing fits with or without them. If the target is of a known type, fit results from similar objects at the same phase make for a good starting point.

Both programs use a structured-input YAML[8] control file; examples are included with the source code. The example files are organized into named blocks or sections, but in practice, any valid YAML file containing all the required data is sufficient. This makes creation and management of control files trivial to script in Python, Perl, Ruby, or other languages supporting YAML. There are sections common to the control files for both programs—we discuss these common sections first and then follow with program-specific settings for SYN++ and SYNAPPS. The units conventions for all physical quantities expressed in the control files are as follows: angstrom for wavelength quantities, 10^3 km s^{-1} for velocity quantities, and 10^3 K for temperatures. Finally, all command examples use a `$` symbol to indicate a UNIX command-line prompt.

4.1. Common Control-File Components

Both SYN++ and SYNAPPS operate the same type of spectrum synthesis calculation, so the control files for both programs have some parts in common. In principle, these settings require only occasional adjustment—the variables most likely to require adjustment specify the location of line-list files, line-forming-region maximum-velocity limit, and whether or not output spectra will be "flattened." Some performance increase can be achieved by making various grids smaller, but this may incur an accuracy loss.

The `grid` control-file section governs radial velocity, line opacity, and line source-function grids:

```
grid:
  bin_width: 0.3
  v_size: 100
  v_outer_max: 30.0
```

[4] See http://heasarc.gsfc.nasa.gov/fitsio/.
[5] See http://www.netlib.org/blas/.
[6] See http://www.netlib.org/lapack/.
[7] See https://software.sandia.gov/appspack/downloads/.
[8] See http://www.yaml.org.

The parameters, in order, are the velocity width of the opacity bins, the number of radial velocity grid points in the line-forming region, and the maximum velocity allowed during the simulation at the outer edge of the line-forming region. The last parameter dictates the amount of wavelength overshoot on either end of the synthetic spectrum wavelength grid to use when loading atomic line data from disk.

The section `opacity` contains parameters for managing atomic-line-list data and how Sobolev line opacity is distributed spatially:

```
opacity:
  line_dir: /usr/local/share/es/lines
  ref_file: /usr/local/share/es/refs.dat
  form: exp
  v_ref: 10.0
  log_tau_min: -2.0
```

For each ion, a relatively strong reference line has been selected (in fact, these lines are the same as those used by SYNOW). The opacity in this particular line is directly controlled by the user, with radial parameterization dictated by some functional form. Other lines from the same ion are scaled according to Boltzmann level excitation. The `line_dir` and `ref_file` variables in the preceding example point to a location on disk where the atomic line data files are stored and the path to the reference line-list file. The functional form of the Sobolev opacity parameterization is specified by the `form` variable, with the reference-line opacity normalized at `v_ref`. For the exponential functional form, the reference-line Sobolev opacity for a given ion is given by

$$\tau_{\text{ref}}(v) = \tau_{\text{ref}}(v_{\text{ref}}) \exp\left(\frac{v_{\text{ref}} - v}{v_e}\right). \tag{2}$$

In the initial release of ES, we have included only this radial exponential functional form, but additional forms may be added in the future. Parameter `log_tau_min` specifies an opacity threshold ($\log_{10}$) below which total binned opacity at a given grid point will be ignored.

The two short sections `source` and `spectrum` are mainly concerned with resolution for integrating line source functions and fluxes. The example values listed here are usually adequate:

```
source:
  mu_size: 10
spectrum:
  p_size: 60
  flatten: No
```

At each radial grid point and for each wavelength, the mean intensity integral includes 2×`mu_size` direction cosines, with half subtending the photosphere and the other half subtending the sky. At each wavelength of the synthesized spectrum, `p_size` impact rays subtend the photosphere—additional rays are added as needed to intersect the projected surface of the emission lobe out to the edge of the line-forming region. The `flatten` variable automatically removes the continuum level from the output spectrum, which is mainly useful for visualization.

4.2. Running SYN++

SYN++ is a standalone SN spectrum synthesis program. To run it, one need only supply the path to a SYN++ YAML control file. Assuming that the program has been installed into the user's UNIX executable path, an invocation could be:

```
 $ syn++ syn++.yaml
```

where `syn++.yaml` is the name of the control file used. If OpenMP was enabled at compile time, it may be activated at runtime to parallelize line source-function calculations by setting the `OMP_NUM_THREADS` environment variable to the number of threads desired.

Synthesized output spectra are written to standard output in multicolumn ASCII format. SYN++ is capable of synthesizing multiple SN spectra in succession; the output spectra are separated by a blank line. The output may be written to a file by shell redirect:

```
 $ syn++ syn++.yaml > output_spectrum.dat
```

or piped to another command such as `XMGRACE`:[9]

```
 $ syn++ syn++.yaml | xmgrace obs_spectrum.dat -
```

In the preceding example, SYN++ is used to compute a spectrum or spectra, which are graphically overlaid on an observed one. The program may easily be invoked by a system call from a scripting language like Python, and the output may be captured and postprocessed in some way. In the case that several spectrum calculations are to be performed by one SYN++ run, progress will be reported to standard error if the `--verbose` command-line flag is specified.

The `output` section of the SYN++ control file governs the wavelength grid of the synthetic spectrum:

```
output:
  min_wl: 2500.0
  max_wl: 10000.0
  wl_step: 5.0
```

[9] See http://plasma-gate.weizmann.ac.il/Grace/.

The output spectrum is computed on a regularly spaced grid between the specified limits. Alternatively, the user may want the output spectrum be computed on a wavelength grid taken from another (possibly observed) spectrum. In that case, the `--wl-from` option will override the control-file output section:

```
$ syn++ --wl-from=obs_spectrum.dat syn++.yaml
```

The argument of the `--wl-from` option must be a multi-column ASCII file, with the first column representing wavelengths in angstrom units. This option is useful for direct wavelength-for-wavelength comparison without interpolation.

Each synthetic spectrum computation is controlled by a setup. One or more setups may be listed in a SYN++ control file. Setups are expressed as a YAML list, with each entry preceded on its first line by a dash character. Each setup is represented by a dictionary of key-value pairs. Figure 5 illustrates an example setup in a SYN++ control file.

The first three parameters listed in the example setup are the warping function coefficients. The warping function applied to the synthetic spectrum is

$$W(\lambda) = a_0 + a_1\left(\frac{\lambda - 6500}{6500}\right) + a_2\left(\frac{\lambda - 6500}{6500}\right)^2. \quad (3)$$

Hence, setting both `a1` and `a2` to zero means that, effectively, no warp is applied to the synthetic spectrum, and it will be output as computed.

Parameters `v_phot` and `v_outer` are the inner and outer bounds of the line-forming region, respectively. The value of `v_outer` should never exceed the value of `v_outer_max`

```
setups :
    -   a0      :  1.0
        a1      :  0.0
        a2      :  0.0
        v_phot  :  8.0
        v_outer : 30.0
        t_phot  : 12.0
        ions    : [ 1401, 2001 ]
        active  : [  Yes,  Yes ]
        log_tau : [  0.1,  1.0 ]
        v_min   : [ 10.0, 10.0 ]
        v_max   : [ 30.0, 30.0 ]
        aux     : [  1.0, 10.0 ]
        temp    : [ 10.0, 10.0 ]
    -   a0      :  1.0
        a1      :  0.0
        ...
```

FIG. 5.—Example SYN++ control-file `setups` section. A first full setup is shown, followed by the start of a second.

defined in the `grid` section, to ensure that enough extra blue and red wavelength coverage is included to build up the line source functions and spectrum. A sharply defined Lambert radiator surface is placed at `v_phot` with blackbody temperature `t_phot`.

Each opacity component includes an ion label and a Boolean switch to activate or deactivate the component. In the preceding setup listing, each opacity component corresponds to a column, starting with an entry in the `ions` row at the top and ending with the `temp` row at the bottom. Ion labels consist of two digits for the atomic number and two digits for the ionization state (neutral is 00, so H I is 100, Si II is 1401, and Fe III is 2602). The same ion may be used more than once to set up an opacity component. The `active` switch conveniently toggles an opacity component on or off, which can be helpful for examining the overall effect that an ion has on a synthetic spectrum.

In addition to the ion label and active switch, each opacity component includes five parameters. The first is `log_tau` (base 10), corresponding to the value of $\tau_{\text{ref}}(v_{\text{ref}})$ in equation 2, the Sobolev opacity in the ion's reference line at v_{ref}. The `v_min` and `v_max` parameters are velocity bounds for the opacity component. The `aux` parameter corresponds to an additional parameter that needs to be specified for the spatial dependence of the reference-line opacity—here, it corresponds to v_e in equation 2. Finally, the `temp` parameter is a Boltzmann excitation temperature that sets the strengths of other lines from the same ion relative to the ion reference line.

Opacity profiles for reference lines are computed in the order they are specified in the setup. A consequence is that if two opacity components specify reference-line optical depth for the same ion, and they overlap in velocity space, the rightmost component takes precedence between its `v_min` and `v_max`. If there is no overlap in velocity space, then no such overwriting occurs. Similarly, for a given ion, the rightmost excitation temperature `temp` takes precedence.

Between execution of setups, the atomic line list may be refreshed. If the list of ions between two successive setups is different, then lines from ions used in the first but not in the second are dropped, and lines from new ions used in the second but not in the first are loaded. When two successive setups use the same list of ions, no dropping or loading of lines is triggered. In a case where all setups use the same list of ions, the line list is only loaded once.

4.3. Running SYNAPPS

Assuming that SYNAPPS has been installed into the UNIX executable path, the program may be run as follows:

```
$ mpirun -np 16 synapps synapps.yaml
```

This will start SYNAPPS up with one master and 15 worker MPI processes. The exact command line may depend on the version of MPI and the job-launcher (`mpirun`, `aprun`, etc.) used. As with SYN++, if OpenMP was enabled at build time, it may be activated at runtime by setting the `OMP_NUM_THREADS` environment variable to the number of desired threads, *but per MPI process*. In the preceding example, if the number of OpenMP threads is set to four, a total of 64 OpenMP threads would be used, divided among MPI processes into 16 teams of four. This would be an optimal allocation for a 16-node cluster with four shared-memory cores per node.

The SYNAPPS YAML control file is similar to the SYN++ control file, but instead of the `output` and `setups` section, it includes an `evaluator` and `config` section. The `config` section can be thought of as describing a parameter space from which trial SYN++ setups will be generated. Such trial setups correspond to trial $\boldsymbol{x}$ points indicated in equation (1).

Figure 6 shows an example `evaluator` section governing how the objective function values are computed. The `target_file` setting is the path to the SN spectrum to be fit. This file should be a three-column white-space-delimited ASCII file, with columns wavelength (angstrom), flux (F_λ units), and flux error (also F_λ units). The `vector_norm` parameter controls the fit objective function:

$$f(\boldsymbol{x}) = \left(\sum_{n=1}^{N} \left|\frac{S_\lambda(\lambda_n;\boldsymbol{x}) - F_\lambda(\lambda_n)}{\sigma_n}\right|^{L}\right)^{1/L}. \tag{4}$$

Here, $S_\lambda(\lambda_n;\boldsymbol{x})$ is the synthetic spectrum, $F_\lambda(\lambda_n)$ is the target spectrum, σ_n are the observed flux errors, and L is the value of `vector_norm`. Symbols $\boldsymbol{x}$ and $f(\boldsymbol{x})$ have the same meanings as in equation (1).

The `regions` subsection of the evaluator block specifies what parts of the target spectrum SYNAPPS will fit and which parts it will ignore. Each wavelength in the target spectrum is assigned an additional `weight`, with the value of this weight determined by an “or” operation among the regions specified columnwise. In the preceding listing, only the region between 5400 and 6400 Å will be fit—all other wavelengths below 1 μm will be ignored, because they are assigned zero weight. The second region specification is ignored altogether, since `apply` is set to `No`.

Figure 7 represents the `config` section, which defines the variables to be used in the fit, any imposed constraints, and cache and output files. The `fit_file` and `cache_file` variables specify where the final output fit spectrum will be placed and the path to a cache file. If the cache file already exists, it will be ingested by the program at startup and added to as the fit progresses. If the cache file listed in the `config` section is not present, it will be created.

The parameters listed in the `config` section are the same as in the `setups` section of the SYN++ control file. However, each parameter gets five variables that control the parameter constraints in the fit. A parameter may be held constant during

```
evaluator :
    target_file : "target.dat"
    vector_norm : 2
    regions     :
        apply   : [   Yes,   No,  Yes ]
        weight  : [     0,    1,    1 ]
        lower   : [     0, 3900, 5600 ]
        upper   : [ 10000, 4100, 6400 ]
```

FIG. 6—Example SYNAPPS control-file `evaluator` section.

a fit if `fixed` is set to `Yes`. The initial parameter value is specified by the `start` variable. The lower and upper bounds for the parameter are given by the correspondingly named variables. The start value should be within the lower and upper bounds. Finally, the `scale` variable provides a normalization value; in effect, this is akin to setting the initial step size for this dimension. While SYNAPPS runs, it outputs a step size for which all of these parameter normalizations have been applied.

Finally, the `detach` Boolean implements a linear equality constraint if set to `No`. In this case, the value selected for `v_min` for trial points will be exactly equal to `v_phot`. When not detached, the starting value of `v_min` and `v_phot` should be set equally. If an opacity component is detached, the values of `v_min` and `v_phot` are allowed to differ.

4.4. Limitations and Strengths

It is important to bear in mind what SYNAPPS can and cannot do when deciding on an analysis path for SN spectra. Extracting quantitative abundances from SN spectra requires a more detailed treatment of the ejecta gas equation of state and self-consistent radiation-field solution. However, even for more sophisticated codes, it has been noted that such extraction is ultimately still a very ill-posed inverse problem (e.g., Mazzali et al. 2008). More generally, we emphasize that directly interpreting the objective function value as a χ^2 fit measurement is misleading, because the underlying model of this code includes biases from parameterization and cannot account for all radiative processes operating at all wavelengths in a real SN. Hence, direct quantitative comparison of synthetic spectra with observed ones should, in principle, be weighted somehow; our masking and weighting approach works well in practice. Most practitioners are aware of the tendency of some codes (like ours) to overestimate flux at redder wavelengths in postmaximum SN Ia spectra, when a detailed fit to the blue can be made otherwise. Our solution is to deweight, or mask, the red continuum portions—in hand-driven fits, this is simply ignored by the user.

SYNAPPS is designed to provide the same kinds of results as the older SYNOW code, but with more automation to make systematic analysis of larger data sets more practical and accessible

```
config :
    fit_file    : "target.fit"
    cache_file  : "target.cache"

    a0          :  { fixed:  No, start:    1, lower:   0, upper: 10, scale: 10 }
    a1          :  { fixed:  No, start: -2.6, lower: -10, upper: 10, scale: 20 }
    a2          :  { fixed:  No, start: -5.0, lower: -10, upper: 10, scale: 20 }
    v_phot      :  { fixed:  No, start: 10.7, lower:   5, upper: 15, scale: 10 }
    v_outer     :  { fixed: Yes, start:   30, lower:  15, upper: 30, scale:  1 }
    t_phot      :  { fixed:  No, start: 11.4, lower:   5, upper: 25, scale: 20 }

    ions        :  [ 1401, 2001, ]
    active      :  [  Yes,   No, ]
    detach      :  [   No,   No, ]

    log_tau     :
        fixed   :  [   No,   No, ]
        start   :  [ 1.14,    0, ]
        lower   :  [   -2,   -2, ]
        upper   :  [    2,    2, ]
        scale   :  [    1,    1, ]
    v_min       :
        fixed   :  [   No,   No, ]
        start   :  [ 10.7,   10, ]
        lower   :  [    5,    5, ]
        upper   :  [   15,   15, ]
        scale   :  [    1,    1, ]
    v_max       :
        fixed   :  [  Yes,  Yes, ]
        start   :  [   30,   30, ]
        lower   :  [   15,   15, ]
        upper   :  [   30,   30, ]
        scale   :  [    1,    1, ]
    aux         :
        fixed   :  [   No,   No, ]
        start   :  [ 1.35,  1.6, ]
        lower   :  [  0.1,  0.1, ]
        upper   :  [    5,    5, ]
        scale   :  [    1,    1, ]
    temp        :
        fixed   :  [   No,   No, ]
        start   :  [ 20.2,   10, ]
        lower   :  [    5,    5, ]
        upper   :  [   25,   25, ]
        scale   :  [    1,    1, ]
```

FIG. 7.—Example SYNAPPS control-file `config` section.

to the SN community. Where SYNOW excels, SYNAPPS can be expected to do likewise. The most reliable results from both codes come in the form of information about what ion species are present in the ejecta and what their distribution is in velocity space. The parameterized model these codes use, and the assumptions about radiative transfer made that trade in favor of speed, make the use of the code most applicable during the photospheric phase of SN evolution.

5. CONCLUSION AND FUTURE WORK

We have introduced a new code, SYNAPPS, which fuses a valued tool for empirical analysis of SN spectra with a parallel multidimensional optimization framework. The code is especially useful for systematic analysis of large data sets of SN spectra, to place constraints on explosion models and to inform more detailed HPC modeling codes. In the future, we anticipate exploring three axes of SYNAPPS code development. One is to investigate alternative parallel optimization frameworks: in particular, to possibly improve scaling and convergence. Another goal is to improve performance of the objective function through many-core computing. Finally, we are interested in implementing a version of the code replacing the parameterized spectrum synthesis calculation with a radiative equilibrium model to improve constraints on explosion models. We briefly outline these efforts here.

APPSPACK's generating set search will not scale to an arbitrary number of MPI processes, and it is not guaranteed to return a global minimum, but its recently released successor package HOPSPACK (Griffin & Kolda 2008) addresses both of these issues. It allows multiple optimization solvers of even multiple types to run simultaneously, distributing function evaluations to a pool of worker nodes. The optimizers can share information about the objective function as the calculation progresses, enabling cooperation across algorithms with different strengths—a generating set search may learn of a new minimum from a global optimization search and then jump to it. This suggests that better, but also more efficient, sampling of the parameter space can be achieved than with APPSPACK. We have ensured that the objective function that SYNAPPS uses is compatible with the HOPSPACK framework interface, and we have run some experiments with it. It is our intent to make APPSPACK easily swapped out with HOPSPACK when the latter is mature enough, with minimal impact to users. Part of the appeal of APPSPACK is its lack of need for analytic derivatives of the objective function. While these are not readily calculable, numerical derivatives may be useful if the objective function is not particularly noisy. There do exist frameworks of parallel gradient-based optimizers that can incorporate numerical derivatives (for example, OPT++; Meza et al. 2007). The main expectation from using gradient information is improved rates of convergence.

The SYNAPPS objective function takes about a minute to return a result—further parallelism through accelerator hardware (such as graphics processor units [GPUs]) may promise substantial performance improvements. Our first experiment was to move the calculation of the main bottleneck code, the line source function, to GPUs. A direct port in OpenCL (a proposed standard for heterogeneous computing) results in a speedup of a factor of 10 on an NVIDIA Tesla C1060 card, including the cost of transferring data to the GPU. With more effort to capitalize on the GPU memory hierarchy, a factor of 40 has been achieved in the test code. We have begun development of an experimental branch of SYNAPPS using GPUs.

Finally, it would be very interesting to replace the highly parameterized SYNOW-style objective function in SYNAPPS with a self-consistent radiative equilibrium model, such as that employed in abundance tomography (e.g., Stehle et al. 2005). This calculation is more complex, but could capitalize on findings from the preceding studies. Monte Carlo methods can ensure very fast convergence for SN problems (Lucy 1999; Kasen et al. 2006), as they keep the radiation-field calculation divergence-free throughout the atmosphere at each iteration, and they parallelize trivially. This capability could allow the construction of model SN ejecta profiles from a time-series set of spectroscopic observations through a *simultaneous* fit. The goal would be to maximize the potential of well-calibrated spectral time series for rigorous constraints on explosion models in the form of derived composition, density, and temperature structures—the type of scan referred to earlier. This type of calculation is more complex, and its implementation would likely benefit from the application of results in the other future software and GPU efforts.

We conclude by suggesting that SYNAPPS could provide high-volume transient experiments with rapid turnaround analyses of a different sort from automated spectroscopic SN classifier codes (e.g., SNID; Blondin & Tonry 2007). These types of analysis tools may prove increasingly important to organize transient surveys in order to efficiently marshal more expensive follow-up resources in real time.

The authors thank Tamara Kolda and Todd Plantenga of Sandia National Laboratories for assistance with APPSPACK and HOPSPACK, Ted Kisner at the Lawrence Berkeley National Laboratory for GNU build tools support and graphics processor unit experiments, the Nearby Supernova Factory for permission to include plots incorporating some of their data prior to publication, and the anonymous referee whose comments greatly improved the paper. R. C. T. thanks fellow participants of the Aspen Center for Physics Summer 2010 workshop, "Taking Supernova Cosmology into the Next Decade," for their input. This work was supported by the Director, Office of Science, Office of High Energy Physics, of the US Department of Energy under contract no. DE-AC02-05CH11231 and by a grant from the Gordon & Betty Moore Foundation. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the US Department of Energy under contract no. DE-AC02-05CH11231. This research was also supported by the US Department of Energy Scientific Discovery through Advanced Computing program under contract no. DE-FG02-06ER06-04.

REFERENCES

Aldering, G., et al. 2002, Proc. SPIE, 4836, 61
Blondin, S., & Tonry, J. L. 2007, ApJ, 666, 1024
Branch, D., Baron, E., & Jeffery, D. J. 2003, Lect. Notes Phys., 598, 47
Branch, D., Dang, L., & Baron, E. 2009, PASP, 121, 238
Fisher, A. 2000, Ph.D. thesis, Univ. Oklahoma
Gal-Yam, A. 2009, Nature, 462, 624
Ganeshalingam, M., et al. 2010, ApJS, 190, 418
Gray, G., & Kolda, T. 2005, ACM Trans. Math. Software, 32, 485
Griffin, J., & Kolda, T. 2006 (SAND2006-4621; Albuquerque: Sandia)
———. 2008 (SAND2008-6553; Albuquerque: Sandia)
Hatano, K., et al. 1999, ApJS, 121, 233
Hubble, E. 1929, PNAS, 15, 168
Jeffery, D. J., & Branch, D. 1990, in Winter School for Theoretical Physics, Supernovae, (Jerusalem: IAS), 149
Jeffery, D. J., et al. 2007, ApJS, 171, 493
Kasen, D., Thomas, R. C., & Nugent, P. 2006, ApJ, 651, 366
Kasen, D., Röpke, F., & Woosley, S. 2009, Nature, 460, 869
Kolda, T. 2005, SIAM J. Optim., 16, 563
Lucy, L. B. 1999, A&A, 344, 282
Matheson, T., et al. 2008, AJ, 135, 1598
Mazzali, P. A., et al. 2008, MNRAS, 386, 1897
Meza, J. C., et al. 2007, ACM Trans. Math. Software, 33, 12
Nelder, J., & Mead, R. 1965, Comput. J., 7, 308
Perlmutter, S., et al. 1999, ApJ, 517, 565
Rau, A., et al. 2009, PASP, 121, 133
Riess, A., et al. 1998, AJ, 116, 1009
Rybicki, G. B., & Hummer, D. G. 1978, ApJ, 219, 654
Scalzo, R., et al. 2010, ApJ, 713, 1073
Stehle, M., et al. 2005, in ASP Conf. Ser. 343, 1604–2004: Supernovae as Cosmological Lighthouses (San Francisco: ASP), 393