The Astronomical Journal, 149:138 (14pp), 2015 April doi:10.1088/0004-6256/149/4/138

AUTONOMOUS GAUSSIAN DECOMPOSITION

Robert R. Lindner[1], Carlos Vera-Ciro[1], Claire E. Murray[1], Snežana Stanimirović[1], Brian Babler[1], Carl Heiles[2], Patrick Hennebelle[3], W. M. Goss[4], and John Dickey[5]
[1] Department of Astronomy, University of Wisconsin, 475 North Charter Street, Madison, WI 53706, USA; rlindner@astro.wisc.edu
[2] Radio Astronomy Lab, UC Berkeley, 601 Campbell Hall, Berkeley, CA 94720, USA
[3] Laboratoire AIM, Paris-Saclay, CEA/IRFU/SAp-CNRS-Université Paris Diderot, F-91191 Gif-sur Yvette Cedex, France
[4] National Radio Astronomy Observatory, P.O. Box O, 1003 Lopezville, Socorro, NM 87801, USA
[5] University of Tasmania, School of Maths and Physics, Private Bag 37, Hobart, TAS 7001, Australia
Received 2014 September 2; accepted 2015 February 13; published 2015 March 23

ABSTRACT

We present a new algorithm, named Autonomous Gaussian Decomposition (AGD), for automatically decomposing spectra into Gaussian components. AGD uses derivative spectroscopy and machine learning to provide optimized guesses for the number of Gaussian components in the data, and also their locations, widths, and amplitudes. We test AGD and find that it produces results comparable to human-derived solutions on 21 cm absorption spectra from the 21 cm SPectral line Observations of Neutral Gas with the EVLA (21-SPONGE) survey. We use AGD with Monte Carlo methods to derive the H I line completeness as a function of peak optical depth and velocity width for the 21-SPONGE data, and also show that the results of AGD are stable against varying observational noise intensity. The autonomy and computational efficiency of the method over traditional manual Gaussian fits allow for truly unbiased comparisons between observations and simulations, and for the ability to scale up and interpret the very large data volumes from the upcoming Square Kilometer Array and pathfinder telescopes.

Key words: ISM: atoms – ISM: clouds – ISM: lines and bands – line: identification – methods: data analysis – techniques: spectroscopic

1. INTRODUCTION: 21 CM GAUSSIAN FITS

Neutral hydrogen (H I) is the raw fuel for star formation in galaxies, and an important ingredient in understanding galaxy formation and evolution through cosmic time. In the interstellar medium (ISM) of the Milky Way, H I is predicted to exist in two thermally stable states: the cold neutral medium (CNM) with temperature between 40 and 200 K, and the warm neutral medium (WNM) with temperature between 4100 and 8800 K (Field et al. 1969; McKee & Ostriker 1977; Wolfire et al. 2003).

The 21 cm hyperfine transition of H I is a convenient tracer of neutral gas because (1) it is easily excited by random thermal motions due to its small transition energy ($\Delta E/k \simeq 0.068$ K), and (2) it is abundant in galaxies over a wide range of interstellar environments and scales, from dense molecular clouds to diffuse galactic halos. The 21 cm line can also be used the measure the H I excitation (or spin) temperature distribution, which can be used to constrain theoretical models of the neutral ISM. One technique for measuring the spin temperature of H I is to fit 21 cm emission and absorption data with a collection of independent isothermal Gaussian components. With this technique, H I spin temperatures have been measured in the range of ∼10–3000 K (e.g., Dickey et al. 1978, 2003; Crovisier et al. 1980; Mebold et al. 1982; Kalberla et al. 1985; Heiles 2001; Heiles & Troland 2003; Begum et al. 2010; Roy et al. 2013). Although Galactic H I surveys have constrained the cold end of the ISM spin temperature distribution with hundreds of detected CNM-temperature components, few components have been detected with temperatures consistent with the WNM. There are, however, indications of $\gtrsim$7000 K H I in damped L_α systems at high redshift (Kanekar et al. 2014). The missing WNM-temperature components leave the spin temperature distribution of the WNM, a phase that contains ∼50% of the mass in the neutral ISM (Draine 2010), unconstrained. Surveys also find a significant fraction of thermally *unstable* components (with temperatures between ∼200 and 4100 K), up to 47% of detections in Heiles & Troland (2003). Although the missing WNM components could be explained in terms of sub-thermal excitation of the 21 cm line in low density environments (e.g., Liszt 2001), recent results from Murray et al. (2014) point instead toward a lack of absorption observations with enough sensitivity to detect WNM-temperature gas, which has an absorption strength ∼100× less than CNM-temperature gas. Additionally, numerical simulations have shown that magnetic fields and non-equilibrium physics like bulk flows and turbulence can affect the expected relative fractions of WNM, CNM, and intermediate temperature (unstable) gas (Audit & Hennebelle 2005; Heitsch et al. 2005; mac Low et al. 2005; Clark et al. 2012; Hennebelle & Iffrig 2014), although observational data cannot yet distinguish between these scenarios. Aside from the missing constraints at high spin temperatures, the main reason it has been difficult to make progress in understanding the neutral ISM is that most observational surveys have sample sizes of only 10–100 sightlines, leaving large statistical errors in the measurements of the H I spin temperature distribution.

The Square Kilometer Array[6] (SKA) and its pathfinder telescopes, the Australian SKA Pathfinder[7] (ASKAP), the recently expanded Karl G. Jansky Very Large Array[8], and MeerKAT[9], will push radio astrophysics into a new era of "big

[6] www.skatelescope.org
[7] www.atnf.csiro.au/projects/askap/index.html
[8] https://science.nrao.edu/facilities/vla
[9] www.ska.ac.za/meerkat

spectral data" by providing scientists with millions of high spectral resolution, high-sensitivity radio emission and absorption spectra probing lines of sight through the Milky Way and neighboring galaxies. This infusion of data promises to revolutionize our understanding of the neutral ISM. However, these new data will bring new challenges in data interpretation. Modeling a 21 cm emission or absorption spectrum as a superposition of N independent Gaussian components requires solving a nonlinear optimization problem with 3 N parameters. Because Gaussian functions do not form an orthogonal basis (solutions are not unique), the parameter space is non-convex (contains local optima instead of a single, global optimum), and therefore the final solutions sensitively depend on the initial guesses of the components' positions, widths, and amplitudes, and especially on the total number of components. To minimize the chances of getting stuck in local optima during model fitting, researchers choose the initial parameter guesses to lie as close to the global optimum as possible. In previous and current surveys, these initial guesses are provided manually, effectively using the pattern-recognition skills of humans to identify the individual components within the blended spectra. This manual selection process is time consuming and subjective, rendering it ineffective for the large data volumes in the SKA era.

Automatic line finding and Gaussian decomposition algorithms can solve the problems above. Below we briefly review some available algorithms for automatic line detection that can help analyze the large data volumes in the SKA era. The Bayesian line finder by Allison et al. (2012) searches parameter space using the nested sampling algorithm (Skilling 2004), and uses Bayesian inference to decide on the optimal number of spectral components. Because the algorithm searches the entirety of parameter space for each value of N, the total number of components, it is computationally expensive. Procedural algorithms like those of Haud (2000) or Nidever et al. (2008) iteratively add, subtract, or merge components based on the effects these decisions have on the resulting residuals of least-squares fits, and have been used to interpret large data sets from, e.g., the Leiden–Argentina–Bonn (LAB) All-Sky H I survey (Kalberla et al. 2005). However, the initial parameters for each fit are adopted from previous solutions in adjacent sky positions, thereby limiting the use these algorithms to densely sampled emission surveys. Topology-based algorithms like Clumpfind (Williams et al. 1994) and Duchamp (Whiting 2012) take advantage of the 3D nature of position–position–velocity data, but are too limited for efficient Gaussian decomposition because they can only detect components that are strong enough to produce local maxima in their spectra, and do not allow for overlapping components. Similarly, GaussClumps (Stutzki & Guesten 1990) only locates strong components that produce local optima in 3D space.

In this paper, we present a new algorithm, called Autonomous Gaussian Decomposition (AGD), which provides a capability that is complementary to the algorithms described above by focusing on computational speed in Gaussian decomposition. AGD uses computer vision and machine learning to quickly provide optimized guesses for the initial parameters of a multi-component Gaussian model. AGD allows for the interpretation of large volumes of spectral data and for the ability to objectively compare observations to numerical simulations in a statistically robust way. While the development of AGD was motivated by radio astrophysics, specifically the 21 cm SPectral line Observations of Neutral Gas with the EVLA[10] (21-SPONGE) survey (Murray et al. 2014, 2015), the algorithm can be used to search for one-dimensional Gaussian (or any other single-peaked spectral profile)-shaped components in any data set.

In Section 2, we explain the algorithm; in Section 3, we show AGD's performance in decomposing real 21 cm absorption spectra; and in Section 4, we present a discussion of results and conclusions.

2. AUTONOMOUS GAUSSIAN DECOMPOSITION

AGD approaches the problem of Gaussian decomposition through least-squares minimization by focusing on the task of choosing the parameters' initial guesses, where human input is traditionally most needed. By quickly producing high quality initial guesses, most of the work in interpreting the spectrum has been done, and the resulting least-squares fit converges quickly to the global optimum. The strategy of separating the "guess" and "fit" steps of nonlinear least squares optimization into two automatic algorithms was also used by Barriault et al. (2010), who produced initial guesses using a genetic algorithm.

In the following, x and $f(x)$ represent an example spectrum. For example, x might have units of frequency and $f(x)$ units of flux density. Where relevant, all one-dimensional variables are to be interpreted as column vectors. The variables a, σ, and μ represent the amplitude, "1σ" width (hereafter referred to as just the "width"), and position of a Gaussian function G according to

$$G(x;\, a,\, \mu,\, \sigma) = a\, e^{-(x-\mu)^2/2\sigma^2}. \tag{1}$$

2.1. Derivative Spectroscopy

AGD uses derivative spectroscopy to decide how many Gaussian components a spectrum contains, and also to decide where they are located. Derivative spectroscopy is the technique of analyzing a spectrum's derivatives to gain understanding about the data. It is used in computer vision applications because derivatives can respond to shapes in images like gradients, curvature, and edges. It has a long history of use in biochemistry (see, e.g., Fell 1983), and has been recently used to analyze the spectral features of H I self-absorption in two Galactic molecular clouds (Krčo et al. 2008).

The algorithm places one guess at the location of every *local minimum of negative curvature* in the data, where the curvature of $f(x)$ is defined as the second derivative, $d^2f(x)/dx^2$. This criterion finds "bumps" in the data, and has the sensitivity to detect weak and blended components. Mathematically, this condition corresponds to locations in the data which satisfy the four conditions:

$$f > \epsilon_0 \tag{2a}$$

$$\mathrm{d}^2 f/\mathrm{d}x^2 < 0 \tag{2b}$$

$$\mathrm{d}^3 f/\mathrm{d}x^3 = 0 \tag{2c}$$

$$\mathrm{d}^4 f/\mathrm{d}x^4 > 0. \tag{2d}$$

[10] The Expanded Very Large Array, currently known as the Karl G. Jansky Very Large Array.

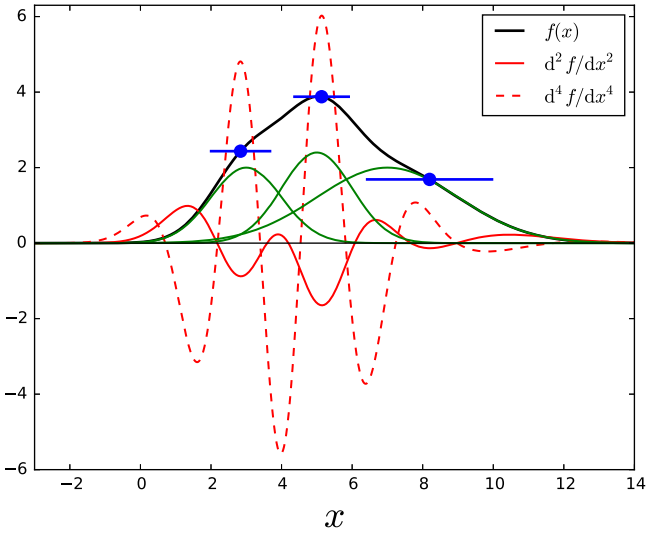

Figure 1. Derivative spectroscopy example. The green and black solid lines show the individual components and total signal, respectively, for a noise-free spectrum consisting of three Gaussian components. Overplotted are the 2nd (red solid) and 4th (red dash) numerical derivatives. The locations (i.e., μ) in the data satisfying the conditions from Equations (2a) to (2d) are identified with blue circles, with blue line segments showing the guessed $\pm 1\sigma$ widths from Equation (4). The positions and widths indicated by the blue circle and line segments represent the guesses that AGD would produce for this example spectrum.

In ideal, noise-free data, we could set $\epsilon_0 = 0$ in Equation (2a); however, observational noise produces random curvature fluctuations and a signal threshold should be applied to avoid placing guesses in signal-free regions. Equation (2b) enforces that the curvature is negative, while Equations (2c)–(2d) ensure the location is a local minimum of the curvature. The N values of x satisfying Equations (2a)–(2d) serve as the guesses for the component positions μ_n where $n \in \{1 \ldots N\}$. In practice, the allowed values of μ_n are restricted to the discrete set of channel centers in the data. Figure 1 shows an example of applying Equations (2a)–(2d) to find the component locations in an ideal noise-free spectrum.

Next, AGD guesses the components' widths by exploiting the relation between a component's width and the maximum of its second derivative. For an isolated component, the peak of the 2nd derivative is located at $x = \mu$, and has a value of

$$\left.\frac{\mathrm{d}^2}{\mathrm{d}x^2} G(x;\, a,\, \mu,\, \sigma)\right|_{x=\mu} = -\frac{a}{\sigma^2}. \tag{3}$$

AGD applies this single-component solution to provide estimates for the widths of all n components σ_n by approximating $a \approx f(x)$ to obtain

$$\sigma_n^2 = f(x)\left(\frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2}\right)^{-1}\Big|_{x=\mu_n}. \tag{4}$$

Finally, AGD guesses the components' amplitudes, a_n. Naive estimates for the amplitudes of the N components are simply the values of the original data evaluated at the component positions. However, if the components are highly blended, then the naive guesses can significantly over estimate the true amplitudes. AGD compensates for this overestimate by attempting to "de-blend" the amplitude guesses using the information in the already-produced position and width guesses (see Appendix A for details on the deblending process).

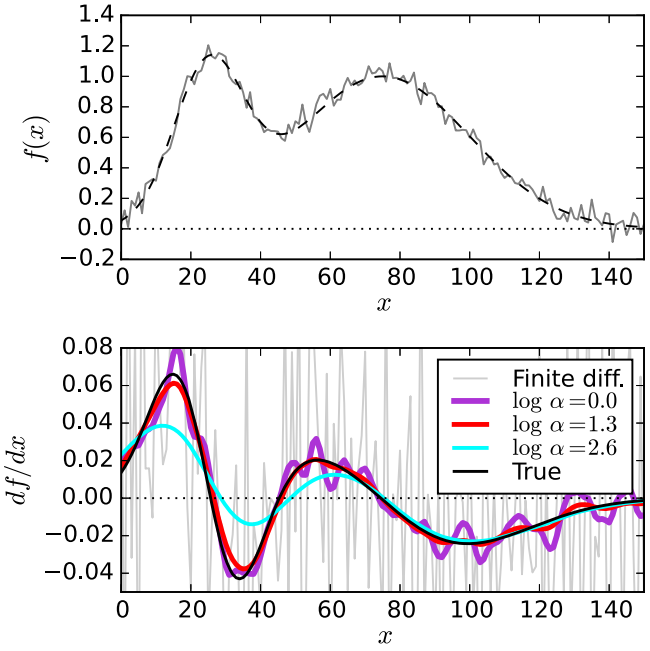

Figure 2. Regularized numerical derivatives. Above: the gray data show an example spectrum with two Gaussian components, one with $a = 1$, $\sigma = 10$ and $\mu = 25$ and one with $a = 1$, $\sigma = 25$, and $\mu = 75$, in Gaussian-distributed noise with rms = 0.05. The dashed black line represents the noise-free data. Below: the black solid line shows the ideal derivative of the underlying data. The result of a finite-difference-based numerical derivative applied to the noisy data is shown in gray. The amplified noise makes it impossible to locate local optima reliably. The remaining (purple, red, and cyan) curves show the regularized derivatives (Section 2.2) using different values of the regularization parameter $\log \alpha$. Larger or smaller values of α trade smoothness for data fidelity, respectively.

2.2. Regularized Differentiation

In order to identify components in $f(x)$ using Equations (2a)–(2d), the derivatives of $f(x)$ must be accurate and smoothly varying. Any noise in the derivatives of the spectra will produce spurious component guesses. Computing derivatives using finite-difference techniques greatly amplifies noise in the data (see, e.g., Figure 2), thereby rendering finite-difference techniques unusable for our needs of computing derivatives up to the fourth order. We regularize[11] the differentiation process using Tikhonov regularization (Tikhonov 1963), where the derivative is fit to the data under the constraint that it remains smooth by following the technique presented in Vogel (2002) and Chartrand (2011).

We define the regularized derivative of the data as $u_{\min} = \arg\min_u(R[u])$, where

$$R[u] = \alpha \int \sqrt{(D_x u)^2 + \beta^2} + \int |A_x u - f|^2, \tag{5}$$

the derivative operator $D_x u = \mathrm{d}u/\mathrm{d}x$ and the anti-derivative operator $A_x u = \int_{x_{\min}}^{x} u\, \mathrm{d}x$. The first term on the right-hand side (RHS) of Equation (5) is the regularization term, and is an

[11] Regularization techniques are also used in, e.g., the maximum entropy method of synthesis image deconvolution (Taylor et al. 1999), and in gravitational lens image inversion (Wallington et al. 1994).

approximation to Total-Variation (TV) regularization (Rudin et al. 1992). When β is zero, this term becomes the L_1 norm of du/dx, pushing u to be piecewise constant. When β is >0, the regularization term behaves more like the L_2 norm of du/dx, constraining u instead to be smoothly varying. To produce smoothly varying solutions for our derivatives, we (a) set $\beta = 0.1$, and (b) rescale the bin widths to unity and peak-normalize the data; these scale factors are remembered and reapplied when optimization has completed.

The second term of the RHS of Equation (5) is the data fidelity term, enforcing that the integral of u closely follows the data f. The parameter α controls the relative balance between smoothness and data fidelity in the solution, i.e., between variance and bias. When $\alpha = 0$, $u_{\rm min}$ is equal to the finite difference derivative. Because of the large range that α can span, we hereafter refer to the regularization parameter as $\log_{10} \alpha \equiv \log \alpha$.

Figure 2 shows how the parameter $\log \alpha$ affects the regularized derivative of synthetic data consisting of two Gaussian components within Gaussian distributed noise. The above panel shows the noisy synthetic data (gray), and the below panel compares the finite-difference derivative (gray) to the regularized derivative for different values of $\log \alpha$ (purple, red, and blue). Larger values of $\log \alpha$ ignore variations in the data on increasingly larger spectral scales. The true derivative (black line) is reproduced well using $\log \alpha = 1.3$.

The above algorithm is a noise suppression technique for numerical derivatives. Another common method for smoothing data is Gaussian convolution, or Gaussian filtering. In Appendix B, we show how our optimization-based methodology compares to convolution and find that optimization returns higher accuracy derivatives, especially in data containing a range of spectral scales.

2.3. Choosing $\log \alpha$ with Machine Learning

In supervised machine learning, the computer is given a collection of input/output pairs, known as a training set, and then "learns" a general rule for mapping inputs to outputs. After this "training" process is completed, the algorithm can be used to predict the output values for new inputs (see, e.g., Bishop 2006; Ivezic̀ et al. 2014).

The regularization procedure of Section 2.2 allows us to take smooth derivatives at the expense of introducing the free parameter $\log \alpha$, which controls the degree of regularization. We use supervised machine learning to train AGD and pick the optimal value of $\log \alpha$ which maximizes the accuracy of component guesses on a training set of spectra with known decompositions. One can obtain the training set by manually decomposing a subset of the data, or by generating new synthetic spectra using components that are drawn from the same distribution as the science data. In the latter case, there is a risk that the training data are different from the science data, but also the benefit that the decompositions are guaranteed to be "correct" while the manual decompositions are not.

Given N_g component guesses $\{a_i^g, \mu_i^g, \sigma_i^g\}_{i=1}^{N_g} \equiv g_\alpha$, produced by running AGD with fixed $\log \alpha$ on data containing N_t true components $\{a_i^t, \mu_i^t, \sigma_i^t\}_{i=1}^{N_t} \equiv t$, the "accuracy" $\mathcal{A}$ of the guesses is defined using the balanced F-score. The balanced F-score is a measure of classification accuracy that depends on both precision (fraction of guesses that are correct) and recall (fraction of true components that were found), thus penalizing component guesses which are incorrect, missing, or spurious. The accuracy is given by

$$\mathcal{A}\left(g_\alpha, t\right) = \frac{2N_c}{N_g + N_t} \equiv \mathcal{A}(\log \alpha), \tag{6}$$

where N_c represents the number of "correct" guesses. We consider a single guessed component (a^g, σ^g, μ^g) to be a "correct" match to a true component (a^t, σ^t, μ^t) if its amplitude, position, and width are all within the limits specified by the following equations:

$$\begin{aligned} c_1 &< \frac{a^g}{a^t} < c_2 \\ \left|\frac{\mu^g - \mu^t}{\sigma^t}\right| &< c_3 \\ c_4 &< \frac{\sigma^g}{\sigma^t} < c_5. \end{aligned} \tag{7}$$

The analysis in Section 3 uses $(c_1 \ldots c_5) = (0, 10, 1, 0.3, 2.5)$. The final solution is least sensitive to the initial amplitudes, so we choose the values c_1 and c_2 to bracket a large relative range; it is more sensitive to the guessed widths, so we chose a narrower relative range in c_4 and c_5; finally, we find that the positions are the most important parameters for fitting the data in the end, motivating the relatively strict value of c_3. We impose the additional restriction that matches between guessed and true components must be one-to-one, and therefore match consideration proceeds in order of decreasing amplitude.

The optimal value of $\log \alpha$ is that which maximizes the accuracy (Equation (6)) between AGD's guessed components and the true answers in the training data. This nonlinear optimization process is performed using a modified version of gradient descent and is described in detail in Appendix C.

3. PERFORMANCE: 21 CM ABSORPTION

We test AGD by comparing its results to human-derived answers for the first 21 spectra from the 21-SPONGE survey (Murray et al. 2015) using GaussPy (Appendix D), the Python implementation of AGD. GaussPy extends the AGD algorithm (i.e., Section 2) to search for components at multiple different scales by offering who modes, referred to as "one-phase" (using one regularization parameter, $\log \alpha$) and "two-phase" (using two regularization parameters, $\log \alpha_1$ and $\log \alpha_2$) decomposition. 21-SPONGE spectra cover a velocity range from -100 to $+100$ $\mathrm{km\,s^{-1}}$), tracing Galactic H I gas. 21-SPONGE's 21 cm absorption spectra are among the most sensitive ever observed with typical optical-depth rms sensitivities of $\sigma_\tau \lesssim 10^{-3}$ per $0.4\,\mathrm{km\,s^{-1}}$channel (Murray et al. 2015). This combination of sensitivity and spectral resolution will stay among the best obtainable through the SKA era. The survey data come natively in units of fractional absorption (I/I_0), and we transform the data into optical depth units ($\tau = -\ln(I/I_0)$) for the AGD analysis because only in τ-space will a single component produce a single peak in curvature (i.e., absorption signals with $\tau \gtrsim 1$ will produce *dual* peaks in the curvature of I/I_0).

We begin by constructing the training data set, which is based on independent 21 cm absorption observations from the Millennium Arecibo 21 cm Absorption-Line Survey (Heiles & Troland 2003). We produce 20 synthetic spectra by randomly

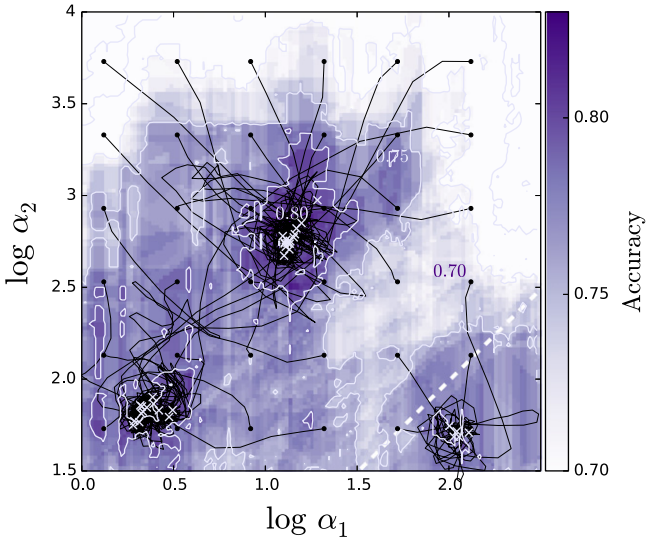

Figure 3. Training AGD using gradient descent. Starting locations, tracks, and convergence locations of the parameters ($\log \alpha_1$, $\log \alpha_2$) during AGD's two-phase training process (Appendix C) are represented by black circles, black lines, and white "X"s, respectively. The dashed white line marks the $\log \alpha_1 = \log \alpha_2$ boundary. Tracks that begin too far away from the global best solution ($\log \alpha_1 = 1.12$, $\log \alpha_2 = 2.73$) can converge into local optima with lower resulting accuracy. Multiple training runs with different starting positions are therefore recommended to find the global optimum. Additionally, physical considerations like the expected width of components can help guide the choice of starting value. The background image shows a densely sampled representation of the underlying parameter space, and was generated using the HTCondor cluster at the University of Wisconsin's Center for High-Throughput computing.

selecting Gaussian components from the Heiles & Troland (2003) catalog. The number of components per spectrum is chosen to be a uniform random integer between the mean value (three) and the maximum value (eight) from the observations. Only components with peak optical depth $\tau < 3.0$ are included in the training data because beyond this, the absorption signal saturates and the component properties are poorly constrained. We next add Gaussian-distributed noise with rms $= 10^{-2}$ per 0.4 km s^{-1} channel to the spectra (in observed I/I_0 space) to mimic real observational noise from the Millenium survey (Heiles & Troland 2003), and re-sample the data at 0.1 km s^{-1}/channel to avoid aliasing the narrowest components (with FWHMs of $\sim$1 km s^{-1}) in the training set. We set the global threshold, (parameter ϵ_0 in Equation (2*a*)), to be 5$\times$ the rms for individual spectra.

We next train AGD for both one- and two-phase decompositions and compare their performances. For one-phase AGD we use the initial value $\log \alpha_1 = 3.00$ and AGD converged to $\log \alpha_1 = 1.29$. The resulting accuracy was 0.78 on the training data, and 0.71 on an independent test-set of 100 newly generated (out-of-sample) synthetic spectra. Testing the performance on similar but independent out-of-sample "test" data prevents against "over-fitting" the training data. For two-phase AGD, we use initial values of $\log \alpha_1 = 1.3$ and $\log \alpha_2 = 3.0$ and AGD converged to $\log \alpha_1 = 1.12$, $\log \alpha_2 = 2.73$, returning 0.81 on the training data and 0.79 on the independent test data from above. Figure 3 shows the convergence tracks of ($\log \alpha_1$, $\log \alpha_2$) when the two-phase training process is initialized with different initial values for $\log \alpha_1$ and $\log \alpha_2$. The $\log \alpha$ values between one- and two-phase decompositions generally follow the trend $\log \alpha_1^{\text{two phase}} < \log \alpha^{\text{one phase}} < \log \alpha_2^{\text{two phase}}$, and this property can be used to help choose initial values during training.

We next apply the trained algorithm to the 21-SPONGE data. We find that two-phase AGD performs better than one-phase in decomposing the 21-SPONGE data, which contain absorption signatures from two distinct populations of ISM clouds: cold clouds with narrow absorption features and warm clouds with broad absorption features. We compare the performance of AGD to human decompositions using the average difference in the number of modeled components ΔN:

$$\Delta N = \left\langle N_{\text{AGD}} - N_{\text{Human}} \right\rangle, \tag{8}$$

and the average fractional change in the residual rms, f_{rms}:

$$f_{\text{rms}} = \left\langle \frac{\text{rms}_{\text{AGD}} - \text{rms}_{\text{Human}}}{\text{rms}_{\text{Human}}} \right\rangle. \tag{9}$$

We find that $\Delta N = -0.14$ and $f_{\text{rms}} = +29\%$ for one-phase AGD and $\Delta N = +0.1$ and $f_{\text{rms}} = -2.2\%$ for two phase AGD. Both one-phase and two-phase AGD guessed comparable numbers of components, but two-phase AGD resulted in lower residual errors compared to human-decomposed spectra, consistent with two-phase AGD's higher accuracy (i.e., 0.79 versus 0.71, for two and one-phase AGD, respectively). A comparison between the resulting number of components and rms residuals between two-phase AGD and human results for the individual spectra is shown in Figure 4.

In Figure 5, we show a scatter plot of the best-fit FWHMs and peak amplitudes for all AGD and human-derived Gaussian components for the 21-SPONGE data. There are 120 and 118 components detected by AGD and human, respectively. We performed a 2-sample Kolmogorov–Smirnov test on the amplitudes ($p = 0.997$), FWHMs ($p = 0.64$), and derived equivalent widths (EW $= \int \tau(v) dv$; $p = 0.9995$) of the resulting components from AGD versus human results and find that in each case, the AGD and human distributions are consistent with being drawn from identical distributions. Thus, AGD results are statistically indistinguishable to human-derived decompositions in terms of the numbers on components, the residual rms values, and the component shapes. Figure 6 shows the AGD guesses, AGD-best fits, and human-derived best fits for all 21 spectra in our data set.

3.1. Completeness and Spurious Detections

Observational noise can scatter the measured signals of weak spectral lines below a survey's detection threshold, effectively modifying the measured component distribution by a "completeness" function. The effect of completeness needs to be taken into account in order to make high-precision comparisons between the measured distributions of H I absorption/emission profiles and the predictions of physical models. AGD's speed and autonomy allows for easy reconstruction of a survey's completeness function, and this information can be used to correct the number counts of observed line components so that one can infer the true component distribution to lower column densities.

We demonstrate this procedure by measuring AGD's line completeness of 21-SPONGE H I absorption profiles as a

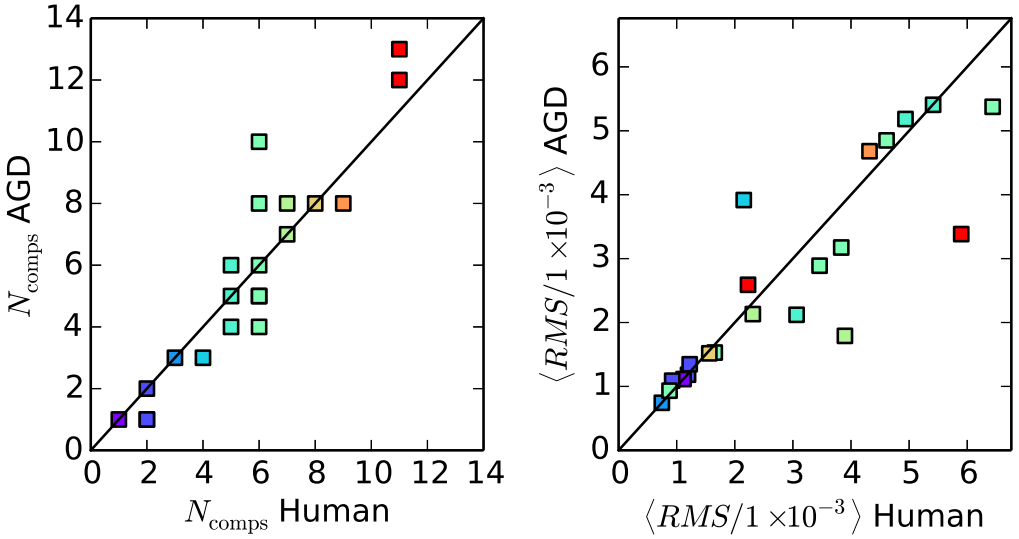

Figure 4. AGD vs. human results for the number of Gaussian components (left) and rms residuals in optical depth (right) for guess + final fit Gaussian decompositions to 21 spectra from the 21-SPONGE survey. The color scale represents the number of human-selected components (corresponding to the *x*-axis of the left panel).

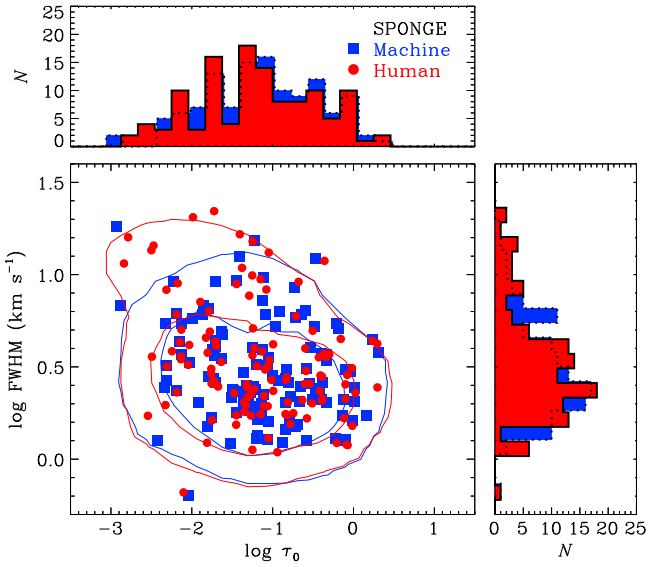

Figure 5. AGD (squares) vs. human (circles) Gaussian decomposition results for 21-SPONGE spectra. The central panel shows peak optical depth (τ_0) and velocity FWHM for each recovered Gaussian component. The contours represent 68 and 95% containment regions. The side panels show marginalized histograms of peak optical depth (top) and velocity FWHM (right) for AGD (dashed) and human (solid) results. There are 118 human-detected components, and 120 AGD-detected components in the 21 spectra.

function of amplitude and velocity width using a Monte Carlo simulation. We inject a single Gaussian component with fixed parameters into synthetic spectra containing realistic observational noise (rms $= 10^{-3}$ per 0.4 km s^{-1} channel) and then run AGD to measure the completeness, which we define as the fraction of successfully detected components out of 50 trials. AGD's completeness function for the 21-SPONGE data is shown in Figure 7. AGD obtains $\simeq$100% completeness for components with FWHM > 1 km s^{-1} and $\tau_0 > 7 \times 10^{-3}$.

Noise fluctuations can also scatter data *above* a survey's detection threshold and produce false-positive detections, also known as "spurious detections." AGD will produce spurious Gaussian component guesses if the threshold in Equation (2*a*) is set too low. The expected number of spurious detections in a spectrum with Gaussian-distributed noise is given by

$$N_{\rm spurious}(S/N) \simeq \frac{V}{{\rm d}V}\left[\frac{1}{2} - \frac{1}{2}\,{\rm erf}\left(\frac{S/N}{\sqrt{2}}\right)\right], \quad (10)$$

where V is the total velocity range, dV is the velocity resolution, and erf is the Gauss error function,

$${\rm erf}\,(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,{\rm d}t. \quad (11)$$

In Figure 8, we perform a Monte Carlo simulation by running AGD on noise-only spectra that mimic the 21-SPONGE data while varying the signal-to-noise ratio (S/N) threshold. Spurious detections increase rapidly below S/N = 3.0.

3.2. Robustness to Varying Observational Noise

Regularized derivatives (Section 2.2) are insensitive to noise on scales less than that set by the regularization parameter $\log\alpha$ (e.g., Equation (B1)). Because the observational sensitivity of 21-SPONGE data is uniform and very high, we next demonstrate that AGD is robust to varying noise properties by characterizing the guessed initial position and initial FWHM of a Gaussian component with fixed true shape in data with increasing noise intensity. Figure 9 shows that $\sim$100% of component guesses remain within $\pm1\sigma$, where $\sigma = {\rm FWHM_{true}}/2\sqrt{2\ln 2}$, of the true component positions for noise intensities ranging from 1–16 $\times$ RMS. Over the same range in noise, the guesses' FWHMs varied by $\pm$20%. Therefore, varying the noise properties has little effect on AGD's performance, making AGD a robust tool to analyze heterogeneous data sets with varying sensitivities.

4. DISCUSSION AND CONCLUSIONS

4.1. Summary

We have presented an algorithm, named Autonomous Gaussian Decomposition (AGD), which produces optimized

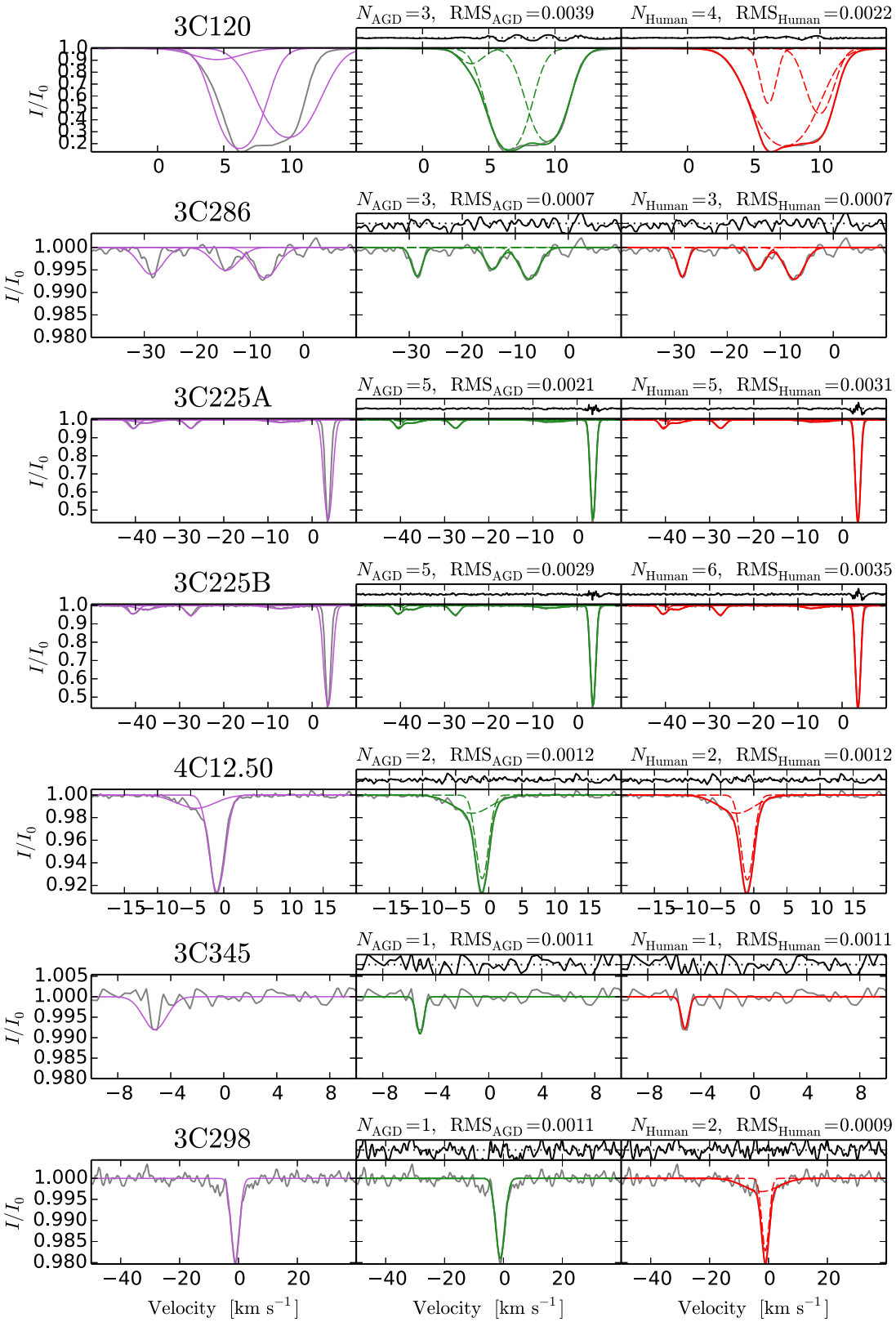

Figure 6. AGD vs. human Gaussian decompositions for 21-SPONGE absorption spectra. The left panels show AGD's initial guesses (purple), the center panels show the resulting best-fit Gaussian components (thin green dashed) and total model (thick green) found by initializing a least-squares fit with these initial guesses, and the right panel shows the human-derived best-fit components (thin red dashed; Murray et al. 2015) and resulting models (thick red). The residual errors between the best-fit total models and the data are shown above the center (AGD) and right panels (human). The number of components in each fit, the source names, and the residual rms values are indicated in the panels.

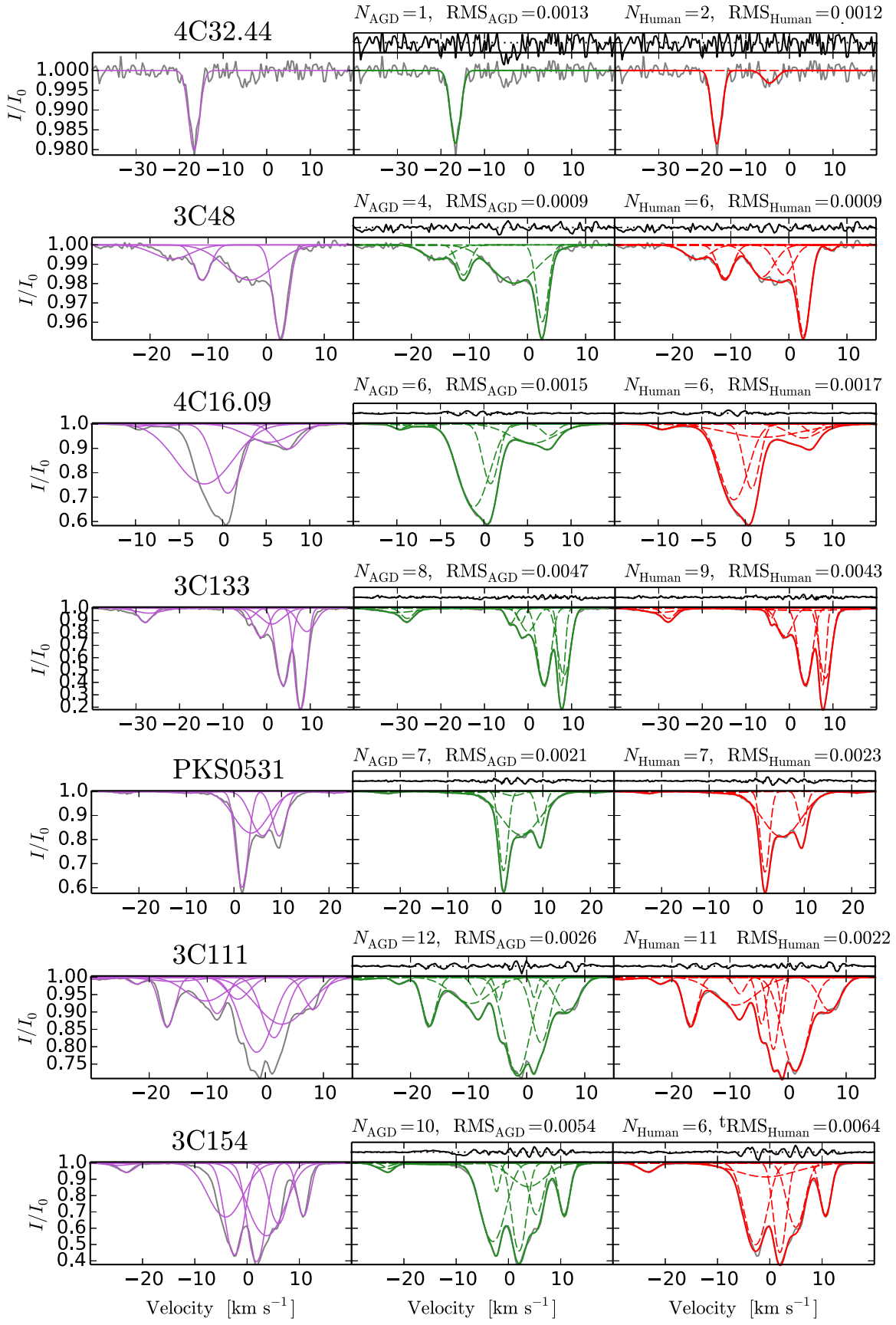

Figure 6. (Continued.)

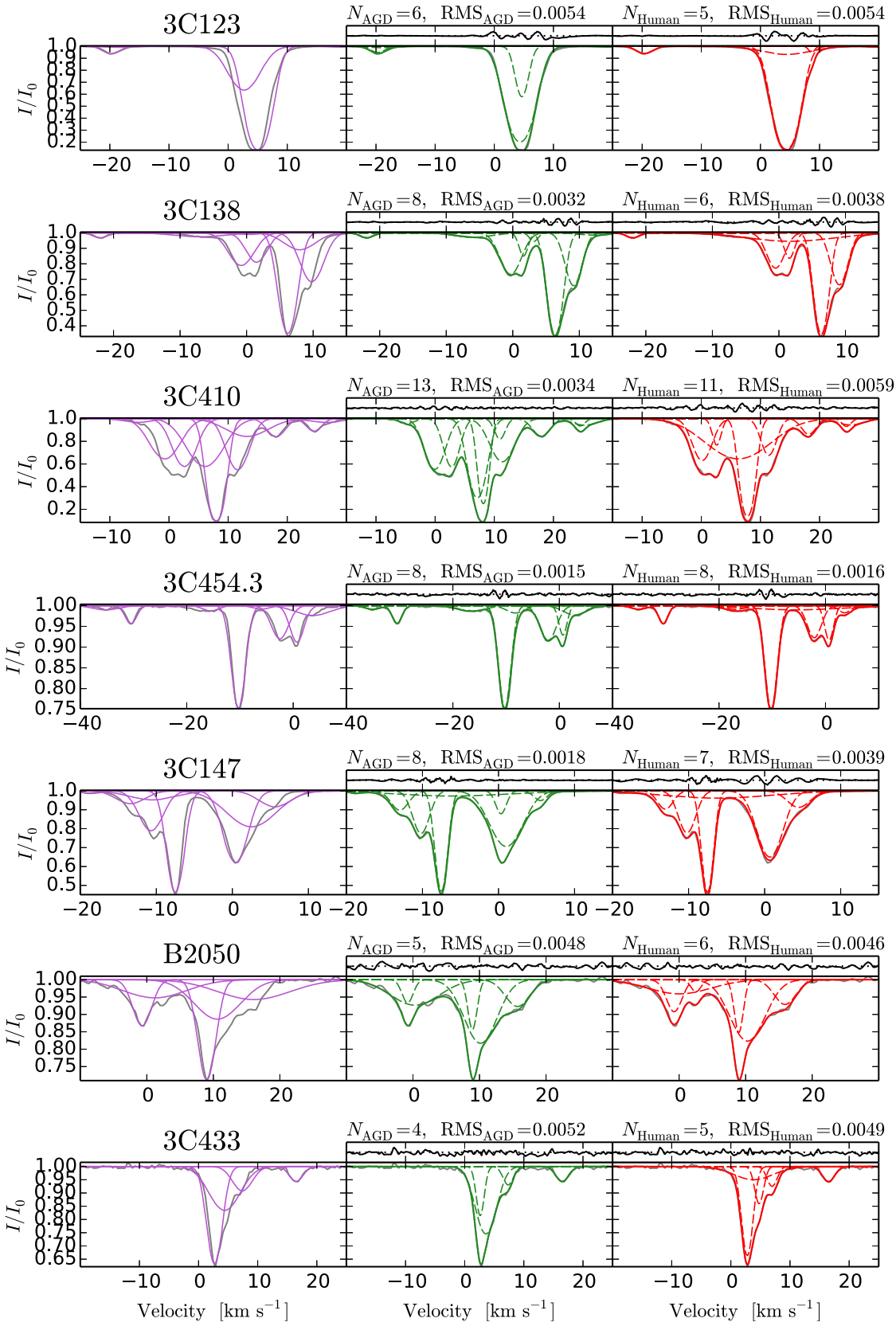

Figure 6. (Continued.)

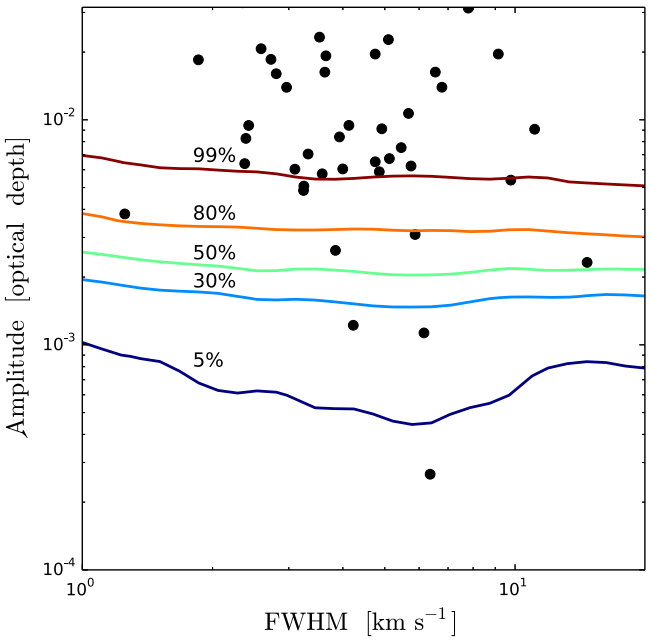

Figure 7. Completeness as a function of component amplitude and FWHM. The labeled contours represent the probability of detecting a component of a given shape, and is equal to the ratio of successful detections in a Monte Carlo simulation where single components were injected one at a time into spectra containing realistic noise of rms $= 10^{-3}$ per 0.4 km s^{-1} channel. The black circles represent the 21-SPONGE detections from AGD. Detected components with amplitudes $\tau_0 > 7 \times 10^{-3}$ have $\simeq$100% completeness.

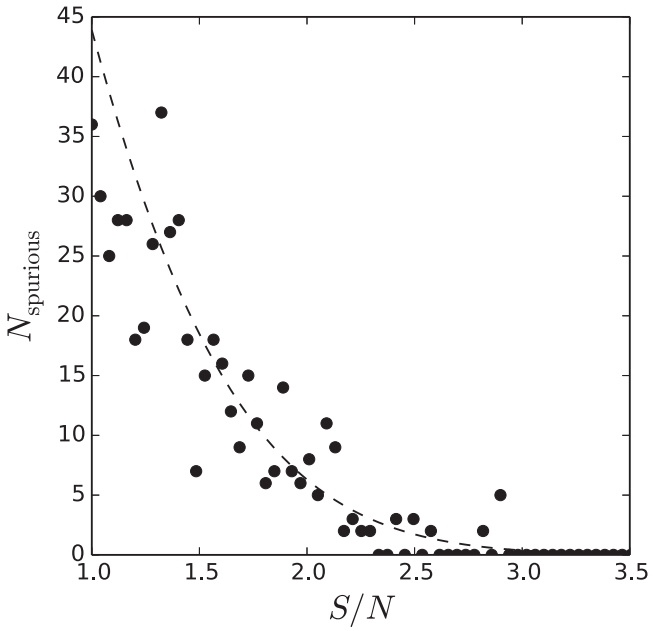

Figure 8. Number of spurious component guesses as a function of S/N threshold when running the trained AGD algorithm on noise-only synthetic spectra. The dashed curve shows the ideal expectation from Equation (10) with the effective velocity resolution d$V = 7.2$, computed using Equation (B1).

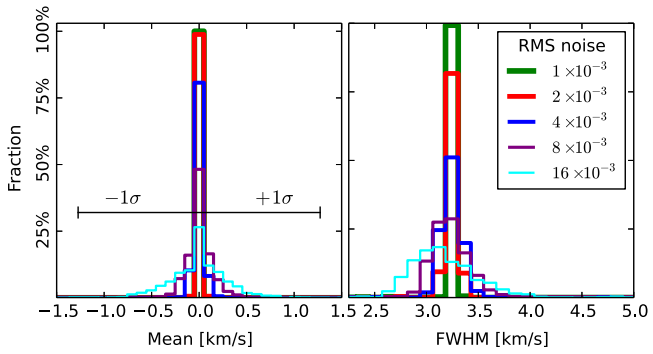

Figure 9. Monte Carlo test of AGD robustness to increasing noise. The left and right panels show the distribution of guessed positions and FWHMs, respectively, for injected components which all have a fixed shape of $a = 0.1$, $\mu = 0$ km s^{-1}, and FWHM $= 3$ km s^{-1}. Different line thicknesses and colors represent different rms noise values, ranging from 1×10^{-3} to 16×10^{-3}. The horizontal bracket displays the $\pm 1\sigma$ width of the injected component.

initial guesses for the parameters of a multi-component Gaussian fit to spectral data. AGD uses derivative spectroscopy and bases its guesses on the properties of the first four numerical derivatives of the data. The numerical derivatives are calculated using regularized optimization, and the smoothness of the derivatives is controlled by the regularization parameter $\log \alpha$. Supervised machine learning is then used to train the algorithm to choose the optimal value of $\log \alpha$ which maximizes the accuracy of component identification on a given training set of data with known decompositions.

We test AGD by comparing its results to human-derived Gaussian decompositions for 21 spectra from the 21-SPONGE survey (Murray et al. 2015). For this test, we train the algorithm on results from the independent Millennium survey (Heiles & Troland 2003). We find that AGD performs comparably to humans when decomposing spectra in terms of number of components guessed, the residuals in the resulting fit, and the shape parameters of the resulting components. AGD's performance is affected little by varying observational noise intensity until the point where components fall below the S/N threshold (i.e., completeness). Combined with a Monte Carlo simulation, we use AGD to measure the H I line completeness of 21-SPONGE data as a function of H I peak optical depth and velocity width. Thus, AGD is well suited for helping to interpret the big spectral data incoming from the SKA and SKA-pathfinder radio telescopes.

AGD is distinct from Bayesian spectral line finding algorithms (e.g., Allison et al. 2012) in terms of the criteria used in deciding the number of components. Where the Bayesian approach chooses the number based on the Bayesian evidence, AGD uses machine learning and is motivated by the answers in the training set. This machine learning approach requires one to produce a training set, but allows for more flexibility in telling the algorithm how spectra should be decomposed.

4.2. Other Applications and Future Work

In Section 3, we used AGD to decompose spectra into Gaussian components which correspond to physical clouds in the ISM. However, AGD can provide a useful parametrization of spectral data even when there is no *physical* motivation to represent the data as independent Gaussian functions. For example, AGD could potentially be used to compress the data volume of wide-bandwidth spectra for easy data transportation, or on-the-fly viewing. For example, If a 16×10^3 channel spectrum contains signals which can be represented by $\sim$10 Gaussian components, then by recasting the data[12] into

[12] The ASKAP spectrometer provides a total of 16,384 channels.

Gaussian component lists one could achieve a data compression factor of ~500.

With our approach to computing smooth derivatives, AGD is constrained to finding populations of components with similar widths (GaussPy currently allows for two populations, i.e., "phases"). A significant improvement would be made if AGD could reliably find components of *any* width without retraining. There are at least two numerical algorithms that may be able to provide this functionality for AGD. The first is Wavelet–Vaguelette Decomposition (Donoho 1995), which builds a smooth representation of data out of a finite collection of smooth template functions. The second is Total Generalized Variation (Bredies et al. 2010), which is a regularized optimization algorithm that can preserve all scales in the data while suppressing noise. Further research is needed to understand the performance (accuracy) and computational cost of these algorithms when applied to AGD.

4.3. Considerations for Real Observational Data

Observational data are often limited in dynamic range by artifacts. In radio astronomy, artifacts can be caused by non-ideal bandpass calibration, radio-frequency interference, or contamination by sources in the telescope's side lobes. AGD is relatively robust to artifacts that have characteristic widths much narrower or much broader than those of the components in the adopted training set. To avoid artifact signals with characteristic widths comparable to the components in the training set, one should increase the signal threshold parameter, ϵ_1 from Equation (2a), enough to ignore the known sources of spectral interference.

One should also take into consideration the channel spacing in the data compared to the expected size of the Gaussian components. There will be additional systematic uncertainty, and potentially numerical instability, in the best-fit parameters of components with widths that are comparable to or *less* than the channel spacing. For example, in the 21-SPONGE data of Section 3, some of the narrowest components have widths that are comparable to the original channel spacing of 0.4 km s^{-1}, so we over-sample the data by $\times 4$ to improve numerical stability of the fitting process.

A Gaussian profile is assumed for the final least-squares fit provided by GaussPy (Appendix D). This assumption is empirically motivated by the observation that most isolated 21 cm line profiles are well-modeled by Gaussian functions, despite the fact that the velocity dispersion of the gas is turbulence dominated (e.g., see discussion in Heiles & Troland 2003). However, even with this built-in assumption, the *initial guesses* from AGD are only weakly dependent on the specific Gaussian shape—much more important is the assumption of a single-peaked profile. Therefore, AGD could be used to provide reasonable initial guesses for the center, width, and peak amplitude of any well behaved, singly peaked profile. For example, the Voigt profile becomes relevant in pressure-broadened emission and absorption lines.

This work was supported by the National Science Foundation (NSF) Early Career Development (CAREER) Award AST-1056780, by NSF Grants No. 1211258 and No. PHYS-1066293, and by the hospitality of the Aspen Center for Physics. We thank the anonymous referee, whose comments have helped to significantly improve this manuscript and the AGD algorithm. C.M. acknowledges support by the NSF Graduate Research Fellowship and the Wisconsin Space Grant Institution. We thank Elijah Bernstein-Cooper, Matthew Turk, James Foster, Jeff Linderoth, James Luedtke, and Robert Nowak for useful discussions. We thank Lauren Michael and the University of Wisconsin's Center for High-Throughput computing for their help and support with HTCondor. The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc. The Arecibo Observatory is operated by SRI International under a cooperative agreement with the National Science Foundation (AST-1100968), and in alliance with Ana G. Méndez-Universidad Metropolitana, and the Universities Space Research Association.

APPENDIX A
DEBLENDED AMPLITUDE GUESSES

AGD "de-blends" the naive amplitude guesses using the fact that when the parameters σ_n and μ_n are fixed, the multi-component Gaussian model becomes a *linear* function of the component amplitudes. Therefore, the naive amplitude estimates can be written as a linear combination of true deblended amplitudes $a^{\rm true}$, weighted by the overlap from each neighboring component. This system of linear equations is expressed in matrix form (see, e.g., Kurczynski & Gawiser 2010) as

$$\begin{pmatrix} B_{11} & \cdots & B_{1N} \\ \vdots & \ddots & \vdots \\ B_{N1} & \cdots & B_{NN} \end{pmatrix} \begin{pmatrix} a_1^{\rm true} \\ \vdots \\ a_N^{\rm true} \end{pmatrix} = \begin{pmatrix} a_1^{\rm naive} \\ \vdots \\ a_N^{\rm naive} \end{pmatrix} \tag{A1}$$

where

$$B_{ij} = e^{\frac{-\left(\mu_i-\mu_j\right)^2}{2\sigma_j^2}}. \tag{A2}$$

The elements of matrix B_{ij} represent the overlap of component j onto the center of component i. When components are negligibly blended, B_{ij} is equal to the identity matrix and $a_n^{\rm true} = a_n^{\rm naive}$. The "true" de-blended amplitude estimates $a_n^{\rm true}$ are then found using the normal equations of linear least squares minimization to be

$$a^{\rm true} = \left(B^T B\right)^{-1} B^T a^{\rm naive}. \tag{A3}$$

In practice, we compute the solution for $a^{\rm true}$ through numerical optimization to avoid inverting a possibly singular matrix B. If all the de-blended amplitude estimates are greater than zero (i.e., physically valid), then they are adopted as the amplitude guesses; if any are $\leqslant 0$ (caused by errors in the estimates of μ_n, σ_n, or the number of components), the naive amplitudes are retained. Therefore,

$$a_n = \begin{cases} a_n^{\rm true} & \text{if all } a_n^{\rm true} > 0 \\ a_n^{\rm naive} & \text{otherwise.} \end{cases} \tag{A4}$$

APPENDIX B
REGULARIZATION VERSUS GAUSSIAN SCALE SPACE

A useful concept in the field of computer vision is the collection of all convolutions between a dataset and Gaussian

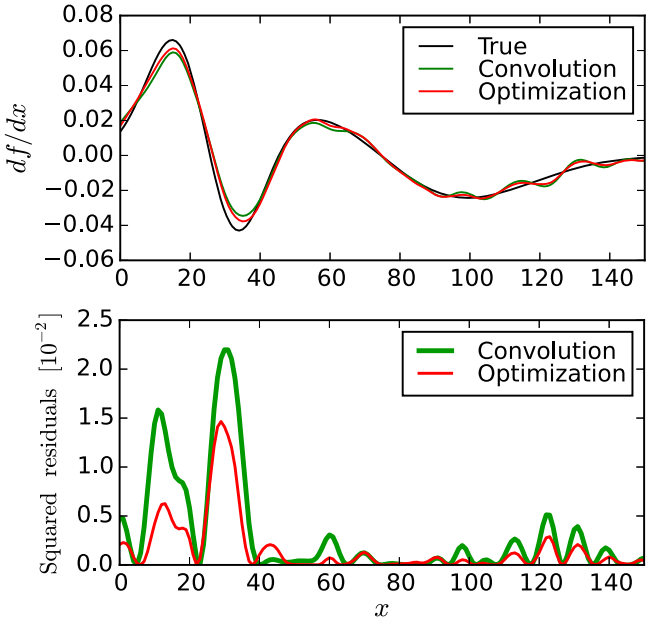

Figure A1. Regularized optimization vs. convolution in computing numerical derivatives. Above panel: the black line shows the true derivative of the data from Figure 2. The best-fitting, de-noised numerical derivatives, computed using convolution (green) and regularized optimization (red) are overplotted. Below panel: squared residuals between each derivative result and the true derivative. Optimization consistently has lower residuals than convolution.

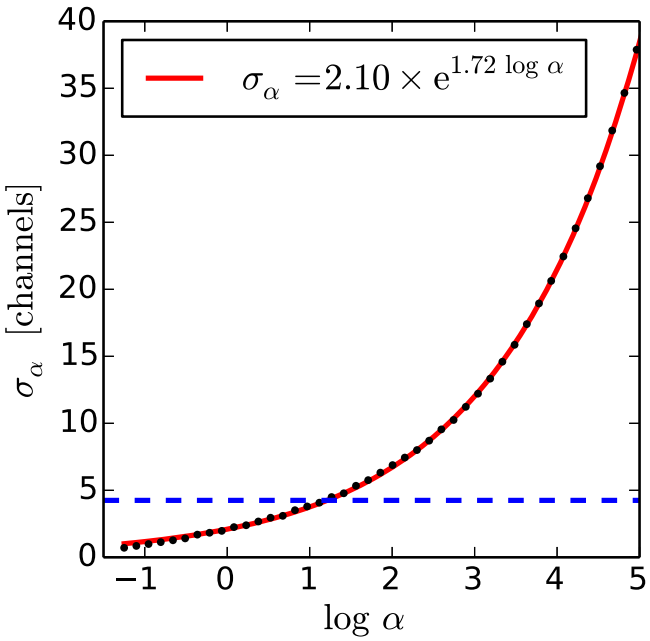

Figure A2. Empirical relation between convolution smoothing parameter σ_α and the regularization parameter $\log \alpha$. The solid red line shows the best-fitting exponential function (Equation (B1)), and the dashed blue line represents the σ_α value corresponding to the width of a single channel of 21-SPONGE data (before oversampling; see Section 3).

kernels of all possible widths (spectral scales); it is known as the data's Gaussian "scale space" representation (e.g., Witkin 1983). In Section 2.2, we solve for the regularized derivative of the data, u, through Tikonov regularization with one regularization term containing the first derivative of u. It can be shown using Fourier analysis that the resulting solution for u is equivalent (in the case of pure L_2 regularization) to a negative exponential-weighted average over the scale space of u' (Florack et al. 2004), where u' is the naive finite-difference based derivative. If we instead include *all* derivatives up to infinite order in the regularization, then the resulting solution is identical to a convolution of u' with a Gaussian kernel of some particular width (Nielsen et al. 1996).

Motivated by this mathematical similarity, in Figure A1 we next compared the smoothed derivatives obtained through regularized optimization (Section 2.2) to those obtained through Gaussian convolution (smoothing). We find that optimization provides a better fit to the derivative of multi-scale data than convolution. This is expected given that the former is a weighted combination of *all* spectral scales in the data, while the latter isolates information on only a single spectral scale (i.e., σ_α). Although the convolution technique is more limited in this sense, it is also more computationally efficient than regularized optimization. Therefore, the two techniques are optimized for different uses, and both are included in the Python implementation of AGD, GaussPy, described in Appendix D.

In Figure A2, we characterize the relation between the optimization-based regularization parameter $\log \alpha$ and the convolution-based smoothing scale σ_α. For each value of $\log \alpha$, we found the matching value of σ_α by minimizing the residuals between the convolution and optimization-based derivatives. The empirically derived scaling relation,

$$\sigma_\alpha \simeq 2.10 \times \ e^{1.72 \log \alpha}, \tag{B1}$$

where σ_α is the spectral scale in channels, is plotted as the solid red curve in Figure A2.

APPENDIX C
MOMENTUM-DRIVEN GRADIENT DESCENT

The regularization parameter $\log \alpha$ (which is generally a multi-dimensional vector; see, e.g., Appendix E) is tuned to maximize the accuracy of component guesses (Equation (6)) using gradient descent with momentum. We define the cost function J that we wish to minimize in order to find this solution as

$$J(\log \alpha) = -\ln \mathcal{A}(\log \alpha). \tag{C1}$$

In traditional gradient descent, updates to the parameter vector $\log \alpha$ are made by moving in the direction of greatest decrease in the cost function, i.e., $\Delta \log \alpha = -\lambda \nabla J(\log \alpha)$, and the learning rate λ controls the step size. Our cost function $J(\log \alpha)$ is highly non-convex, so we use gradient descent (see, e.g., Press et al. 1992) with added momentum to push through local noise valleys. Therefore, at the nth iteration, our parameter update is given by

$$\Delta \log \alpha_n = -\lambda \nabla J\left(\log \alpha_n\right) + \phi \Delta \log \alpha_{n-1}, \tag{C2}$$

where the "momentum" ϕ controls the degree to which the previous update influences the current update.

Because the decision function (i.e., Equation (7)) representing the success or failure for individual component guesses is binary in nature, the cost function $J(\log \alpha)$ is a piecewise-constant surface on small scales (see Figure 3). Therefore, in order to probe the large-scale slope of the cost function surface, we use a relatively large value for the finite difference step size when computing the gradient. For example, the $i_{\rm th}$ component

of the gradient in Equation (C2) is defined according to

$$\nabla_i J(\log \alpha) = \frac{J(\log \alpha_i + \epsilon) - J(\log \alpha_i - \epsilon)}{2\epsilon}, \quad \text{(C3)}$$

where ϵ is the finite-difference step size which we set to $\epsilon = 0.25$. Figure 3 shows example tracks of $\log \alpha = (\log \alpha_1, \log \alpha_2)$ when using gradient descent with momentum during AGD's two-phase training on the 21-SPONGE data. We find that small-scale local optima are ignored effectively during the search for large-scale optima.

APPENDIX D
GAUSSPY: THE PYTHON IMPLEMENTATION OF AGD

GaussPy is the name of our Python[13]/C implementation of the AGD algorithm. The computational bottleneck in performing full Gaussian decompositions is not AGD's production of initial guesses, but the computation time required for the final nonlinear least-squares fit, which takes typically $\sim$1 s. Therefore, a single machine with $N_{\rm cpu}$ cores can decompose $\sim$10 k spectra in $\sim 3/N_{\rm cpu}$ hours after the algorithm is trained. Because GaussPy depends only on freely available open-source packages, it is also easy to deploy on high-throughput computing solutions like HTCondor[14] (see Figure 3) or Apache Spark[15] (Zaharia et al. 2010), allowing for rapid decomposition of very large ($>1\,M$) spectral data sets, e.g., the spectral data products of the SKA. AGD may also be suitable for deployment on the Scalable Source Finding Framework (Westerlund & Harris 2014). GaussPy is maintained by the author and will be publicly available through the Python Package Index[16] upon publication of this manuscript.

The AGD algorithm as explained in Section 2 is optimized for finding components spanning only a modest range in width. This is the cost we pay for the ability to compute smooth derivatives using regularization. In order to search for Gaussian components on widely different spectral scales, e.g., to search for components with widths near 1–3 and 20–30 km s^{-1} in the same spectra, we can iteratively apply AGD to search for components with widths at each of these scales. This capability is included in GaussPy and is referred to as "two-phase" decomposition (for details, see Appendix E). The recently developed algorithm Total Generalized Variation (Bredies et al. 2010) could potentially be used to improve AGD by providing smooth derivatives without any preferred scale, although at a significantly increased computational cost.

GaussPy uses AGD to produce the initial guesses for parameters in a multi-component Gaussian fit, and also carries out the final least-squares fit on the data. In this final optimization, GaussPy uses the Levenberg–Marquardt (Levenberg 1944) algorithm, which has been used in previous 21 cm surveys (e.g., Heiles & Troland 2003), implemented using the Python package LMFIT[17] which allows for non-negativity constraints on the component amplitudes.

In GaussPy, we minimize the functional $R[u]$ (Equation (5)) using the quasi-Newton algorithm "BFGS2" from the GNU Scientific Library[18] `Multimin` package and achieve computation-time scalings of $\mathcal{O}\left(n^{1.95}\right)$, where n is the number of channels in the data, and $\mathcal{O}\left(\alpha^{-0.4}\right)$. The relative scaling between any $\log \alpha$ and the corresponding minimum preserved scale in the data is given by Equation (B1). By inserting an estimate of the expected component widths to Equation (B1), one obtains a rough estimate of the appropriate regularization parameter $\log \alpha$. However, to find the value which maximizes the accuracy of the decompositions, one should solve for $\log \alpha$ using the machine learning technique of Section 2.3.

APPENDIX E
TWO-PHASE GAUSSIAN DECOMPOSITION

Two-phase decompositions allow researchers to decompose spectra which contain components that are drawn from two distributions with very different widths. GaussPy performs two-phase decomposition by first applying the usual AGD algorithm but with a non-zero threshold used in Equation (2b): $df^2/dx^2 < e_2$, which locates only the narrowest components in the data so that they can be removed. The parameters of just these narrow components are next found by minimizing the sum of squared residuals K between the second derivative of the data and the second derivative of a model consisting of only these narrow components, $\{a_n^{\mathcal{N}}, \mu_n^{\mathcal{N}}, \sigma_n^{\mathcal{N}}\}_{n=1}^{N} \equiv \mathcal{N}$, given by

$$K(\mathcal{N}) = \sum_x \left| \frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) - \frac{\mathrm{d}^2}{\mathrm{d}x^2} \sum_n G\left(x; a_n^{\mathcal{N}}, \mu_n^{\mathcal{N}}, \sigma_n^{\mathcal{N}}\right) \right|^2. \quad \text{(E1)}$$

The narrow components are fit to the data on the basis of their second derivatives so that the signals from wider components, which they may be superposed on, are attenuated by a factor $\sim \sigma^2_{\rm narrow}/\sigma^2_{\rm broad}$. The residual spectrum is then fed back into AGD to search for broader components using a larger value of $\log \alpha$ and setting $e_2 = 0$.

[13] GaussPy uses the NumPy (Walt 2011), SciPy (Jones et al. 2001), and matplotlib (Hunter 2007) packages.
[14] http://research.cs.wisc.edu/htcondor/
[15] https://spark.apache.org/
[16] https://pypi.python.org/pypi
[17] http://cars9.uchicago.edu/software/python/lmfit/contents.html
[18] http://gnu.org/software/gsl/

REFERENCES

Allison, J. R., Sadler, E. M., & Whiting, M. T. 2012, PASA, 29, 221
Audit, E., & Hennebelle, P. 2005, A&A, 433, 1
Barriault, L., Joncas, G., Falgarone, E., et al. 2010, MNRAS, 406, 2713
Begum, A., Stanimirović, S., Goss, W. M., et al. 2010, ApJ, 725, 1779
Bishop, C. M. 2006, Pattern Recognition and Machine Learning (Information Science and Statistics) (Secaucus, NJ: Springer-Verlag)
Bredies, K., Kunisch, K., & Pock, T. 2010, SIAM Journal on Imaging Sciences, 3, 492
Chartrand, R. 2011, ISRN Appl. Math., 2011, 164564
Clark, P. C., Glover, S. C. O., Klessen, R. S., & Bonnell, I. A. 2012, MNRAS, 424, 2599
Crovisier, J., Kazes, I., & Aubry, D. 1980, A&AS, 41, 229
Dickey, J. M., McClure-Griffiths, N. M., Gaensler, B. M., & Green, A. J. 2003, ApJ, 585, 801
Dickey, J. M., Terzian, Y., & Salpeter, E. E. 1978, ApJS, 36, 77
Donoho, D. L. 1995, Appl. Comput. Harmon. Anal., 2, 101
Draine, B. 2010, Physics of the Interstellar and Intergalactic Medium, Princeton Series in Astrophysics (Princeton, NJ: Princeton Univ. Press)
Fell, A. F. 1983, TrAC Trends in Analytical Chemistry, 2, 63
Field, G. B., Goldsmith, D. W., & Habing, H. J. 1969, ApJL, 155, L149
Florack, L., Duits, R., & Bierkens, J. 2004, in 2004 International Conference on Image Processing
Haud, U. 2000, A&A, 364, 83

Heiles, C. 2001, ApJL, 551, L105
Heiles, C., & Troland, T. H. 2003, ApJS, 145, 329
Heitsch, F., Burkert, A., Hartmann, L. W., Slyz, A. D., & Devriendt, J. E. G. 2005, ApJL, 633, L113
Hennebelle, P., & Iffrig, O. 2014, arXiv:1405.7819
Hunter, J. D. 2007, CSE, 9, 90
Ivezič, V., Connolly, A. J., VanderPlas, J. T., & Gray, A. 2014, Statistics, Data Mining, and Machine Learning in Astronomy (student ed.; Princeton, NJ: Princeton Univ. Press)
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open source scientific tools for Python, [Online; accessed 2014-08-20]
Kalberla, P. M. W., Burton, W. B., Hartmann, D., et al. 2005, A&A, 440, 775
Kalberla, P. M. W., Schwarz, U. J., & Goss, W. M. 1985, A&A, 144, 27
Kanekar, N., Prochaska, J. X., Smette, A., et al. 2014, MNRAS, 438, 2131
Krčo, M., Goldsmith, P. F., Brown, R. L., & Li, D. 2008, ApJ, 689, 276
Kurczynski, P., & Gawiser, E. 2010, AJ, 139, 1592
Levenberg, K. 1944, QApMa, 2, 164
Liszt, H. 2001, A&A, 371, 698
Mac Low, M.-M., Balsara, D. S., Kim, J., & de Avillez, M. A. 2005, ApJ, 626, 864
McKee, C. F., & Ostriker, J. P. 1977, ApJ, 218, 148
Mebold, U., Winnberg, A., Kalberla, P. M. W., & Goss, W. M. 1982, A&A, 115, 223
Murray, C. E., Lindner, R. R., Stanimirović, S., et al. 2014, ApJL, 781, L41
Murray, C. E., Stanimirović, S., Goss, W. M., et al. 2015, ApJ, in press
Nidever, D. L., Majewski, S. R., & Burton, W. B. 2008, ApJ, 679, 432
Nielsen, M., Florack, L., & Deriche, R. 1996, in Lecture Notes in Computer Science, Vol. 1065, ed. B. Buxton, & R. Cipolla (Berlin, Heidelberg: Springer), 70
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in FORTRAN in: The Art of Scientific Computing (2nd ed.; Cambridge: Cambridge Univ. Press)
Roy, N., Kanekar, N., & Chengalur, J. N. 2013, MNRAS, 436, 2366
Rudin, L. I., Osher, S., & Fatemi, E. 1992, PhyD, 60, 259
Skilling, J. 2004, in AIP Conf. Proc. 735, 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, ed. R. Fischer, R. Preuss, & U. von Toussaint (Melville, NY: AIP), 395
Stutzki, J., & Guesten, R. 1990, ApJ, 356, 513
Taylor, G. B., Carilli, C. L., & Perley, R. A. (ed.) 1999, ASP Conf. Ser. 180, Synthesis Imaging in Radio Astronomy II, ed. G. B. Taylor, C. L. Carilli, & R. A. Perley (San Francisco, CA: ASP)
Tikhonov, A. N. 1963, Soviet Mathematics-Doklady, 4, 1624
Vogel, C. R. 2002, Computational Methods for Inverse Problems, Vol. 23 (Philadelphia: SIAM)
Wallington, S., Narayan, R., & Kochanek, C. S. 1994, ApJ, 426, 60
Walt, S. V. D., Colbert, S. C., & Varoquaux, G 2011, Computing in Science & Engineering, 13, 22
Westerlund, S., & Harris, C. 2014, arXiv:1407.4958
Whiting, M. T. 2012, MNRAS, 421, 3242
Williams, J. P., de Geus, E. J., & Blitz, L. 1994, ApJ, 428, 693
Witkin, A. P. 1983, in Proc. Eighth International Joint Conference on Artificial Intelligence, Vol. 2 (San Francisco, CA: Morgan Kaufmann Publishers), 1019
Wolfire, M. G., McKee, C. F., Hollenbach, D., & Tielens, A. G. G. M. 2003, ApJ, 587, 278
Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. 2010, in Proc 2nd USENIX Conf., Hot Topics in Cloud Computing, 10