GAUSSPY+: A fully automated Gaussian decomposition package for emission line spectra

Downloaded from: https://research.chalmers.se, 2026-10-14 17:14 UTC

Citation for the original published paper (version of record):

Riener, M., Kainulainen, J., Henshaw, J. et al (2019). GAUSSPY+: A fully automated Gaussian decomposition package for emission line spectra. Astronomy and Astrophysics, 628. http://dx.doi.org/10.1051/0004-6361/201935519

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

A&A 628, A78 (2019)
https://doi.org/10.1051/0004-6361/201935519
© M. Riener et al. 2019

Astronomy & Astrophysics

GaussPy+: A fully automated Gaussian decomposition package for emission line spectra*

M. Riener[1,**], J. Kainulainen[1,2], J. D. Henshaw[1], J. H. Orkisz[2], C. E. Murray[3,***], and H. Beuther[1]

[1] Max-Planck Institute for Astronomy, Königstuhl 17, 69117 Heidelberg, Germany
e-mail: riener@mpia.de
[2] Chalmers University of Technology, Department of Space, Earth and Environment, 412 93 Gothenburg, Sweden
[3] Department of Physics & Astronomy, Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218, USA

Received 22 March 2019 / Accepted 25 June 2019

ABSTRACT

Our understanding of the dynamics of the interstellar medium is informed by the study of the detailed velocity structure of emission line observations. One approach to study the velocity structure is to decompose the spectra into individual velocity components; this leads to a description of the data set that is significantly reduced in complexity. However, this decomposition requires full automation lest it become prohibitive for large data sets, such as Galactic plane surveys. We developed GaussPy+, a fully automated Gaussian decomposition package that can be applied to emission line data sets, especially large surveys of HI and isotopologues of CO. We built our package upon the existing GaussPy algorithm and significantly improved its performance for noisy data. New functionalities of GaussPy+ include: (i) automated preparatory steps, such as an accurate noise estimation, which can also be used as stand-alone applications; (ii) an improved fitting routine; (iii) an automated spatial refitting routine that can add spatial coherence to the decomposition results by refitting spectra based on neighbouring fit solutions. We thoroughly tested the performance of GaussPy+ on synthetic spectra and a test field from the Galactic Ring Survey. We found that GaussPy+ can deal with cases of complex emission and even low to moderate signal-to-noise values.

Key words. methods: data analysis – radio lines: general – ISM: kinematics and dynamics – ISM: lines and bands

1. Introduction

Observations of emission lines are of fundamental importance in radio astronomy. Starting with the first detections of neutral hydrogen (HI) via the 21 cm line at 1420.4 MHz by Ewen & Purcell (1951) and the first detection of interstellar carbon monoxide (CO) in the Orion nebula by Wilson et al. (1970), the study of emission lines at radio wavelengths has led to groundbreaking astrophysical insights. Our knowledge about the interstellar medium (ISM) is in large part shaped by observations of the emission of its gas molecules. In particular, we can use the radial velocity – corresponding to Doppler shifts of the emission line with respect to its rest frequency – to gain information about the kinematics and dynamics of the gas.

In our Milky Way, large Galactic surveys of HI (e.g. Stil et al. 2006; Murray et al. 2015; Beuther et al. 2016) and isotopologues of CO (e.g. Dame et al. 2001; Jackson et al. 2006; Dempsey et al. 2013; Barnes et al. 2015; Rigby et al. 2016; Umemoto et al. 2017; Schuller et al. 2017; Su et al. 2019) have been used, for example, to study Galactic structure (e.g. Dame et al. 2001; Nakanishi & Sofue 2006) and construct catalogues of molecular clouds and clumps (e.g. Rathborne et al. 2009; Miville-Deschênes et al. 2017; Colombo et al. 2019). Such studies are usually more focused on the average properties of the gas on Galactic scales or on the scales of molecular clouds or clumps. However, there is a tremendous wealth of physically interesting information that can be gleaned from studying the detailed velocity structure of the gas, among them fundamental insights about turbulence properties in the ISM and molecular clouds (e.g. Larson 1981; Ossenkopf & Mac Low 2002; Heyer & Brunt 2004; Burkhart et al. 2010; Orkisz et al. 2017; for reviews, see Elmegreen & Scalo 2004 and Hennebelle & Falgarone 2012) and dense cores (e.g. Falgarone et al. 2009; Pineda et al. 2010; Keto et al. 2015; Chen et al. 2019), inference about imprints of shear in molecular clouds (e.g. Hily-Blant & Falgarone 2009), and the internal velocity structure of filaments (e.g. Arzoumanian et al. 2013, 2018; Hacar et al. 2013; Henshaw et al. 2014; Orkisz et al. 2019).

While the gas dynamics on smaller scales has been already well studied, the detailed velocity structure of the gas on Galactic scales remains as yet unexplored. We currently do not know whether the velocity structure across large scales shows properties that could serve as diagnostics of phenomena such as molecular cloud formation and evolution or the impact of the Galactic structure on the ISM. To facilitate such analyses, we would ideally like to apply the methods and techniques of the small-scale studies to the large surveys of the Galactic plane.

One approach that has substantial potential is quantifying and analysing the complex spectra taken through the Galactic plane by decomposing them into velocity components and then analysing the properties and statistics of these components. In such analyses, the components are usually assumed to have Gaussian shapes, as random thermal and non-thermal motions in the gas lead to Doppler motions with a Gaussian distribution of gas velocities. Moreover, adopting the Gaussian shape is

* https://github.com/mriener/gausspyplus
** Member of the International Max-Planck Research School for Astronomy and Cosmic Physics at the University of Heidelberg (IMPRS-HD), Germany.
*** NSF Astronomy & Astrophysics Postdoctoral Fellow.

Open Access article, published by EDP Sciences, under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
Open Access funding provided by Max Planck Society.

mathematically simple and leads to a significant reduction in complexity and enables easier post-analysis steps through a rich set of available Gaussian statistics tools.

Recently, several semi-automatic (e.g. Ginsburg & Mirocha 2011; Hacar et al. 2013; Henshaw et al. 2016, 2019) and fully automated (e.g. Haud 2000; Lindner et al. 2015; Miville-Deschênes et al. 2017; Clarke et al. 2018; Marchal et al. 2019) spectral fitting techniques have been introduced. The semi-automated techniques require user interaction, usually in deciding how many velocity components to fit. This can be achieved, for instance, by using spatially smoothed spectra to inform the fit. However, the user-dependent decisions introduce subjectivity to the fitting procedure that reduces reproducibility of the results. The required interactivity with the user can also make it difficult to distribute the analysis to multiple processors. Therefore, while semi-automated approaches are well-suited for small data sets (individual molecular clouds or nearby galaxies at high or low spatial resolution, respectively), they can become prohibitively time-consuming for the analysis of big surveys with millions of spectra and components.

The automated methods overcome these drawbacks by removing the user interaction. The initial number of components can either be a guess (Miville-Deschênes et al. 2017; Marchal et al. 2019) or based on the derivatives of the spectrum (Lindner et al. 2015; Clarke et al. 2018). However, currently these automated routines either: fit the spectra independently from each other (Lindner et al. 2015; Clarke et al. 2018), which might introduce unphysical differences between the fit results of neighbouring spectra; use a fixed number of velocity components as initial guesses (Miville-Deschênes et al. 2017; Marchal et al. 2019), which can be computationally expensive; or are not freely available to the community. Also, the current versions of the automated methods listed above are of the "first generation"; there is still potential to improve the decomposition techniques and their applicability to different data sets.

In this work, we present GaussPy+, an automated decomposition package that is based on the existing GaussPy algorithm (Lindner et al. 2015), but with physically-motivated developments specifically designed for analysing the dynamics of the ISM. We developed GaussPy+ with the specific aim of analysing CO surveys of the Galactic plane, such as the Galactic Ring Survey (GRS; Jackson et al. 2006) and SEDIGISM (Schuller et al. 2017). However, GaussPy+ should be easily adaptable to other emission line surveys for which Gaussian shapes provide a good approximation of the line shapes. Some of the line-analysis tasks of GaussPy+, such as the estimation of noise and the identification of signal peaks, can also be used as independent stand-alone modules to serve more specific purposes.

In this paper, we present the algorithm and test it thoroughly on synthetic spectra and a GRS test field. A full application of GaussPy+ on the entire GRS data set is in preparation and will be presented in a subsequent paper.

2. Archival data and methods

2.1. The GaussPy algorithm

In this work we extend and modify the GaussPy algorithm (Lindner et al. 2015), which is an autonomous Gaussian decomposition technique for automatically decomposing spectra into Gaussian components. While GaussPy was developed for the decomposition of HI spectra (e.g. Murray et al. 2018; Dénes et al. 2018) it can in principle be used for the decomposition of any spectra that can be approximated well by Gaussian functions (e.g. CO).

One of the strengths of the GaussPy algorithm is that it automatically determines the initial guesses for Gaussian fit components for each spectrum with a technique called derivative spectroscopy. This technique is based on finding functional maxima and minima in the spectrum to gauge which of the features are real signal peaks. Since the estimation of maxima and minima requires the calculation of higher derivatives (up to the fourth order), an essential preparatory step in GaussPy is to smooth the spectra in such a way as to get rid of the noise peaks without smoothing over signal peaks (cf. Fig. 2 in Lindner et al. 2015). If the data set contains signal peaks that show a limited range in widths, smoothing with a single parameter α_1 may already lead to good results in the fitting. In the original GaussPy algorithm users can choose between two different versions of denoising the spectrum before derivatives of the data are calculated: a total variation regularisation algorithm and filtering with a Gaussian kernel. We use exclusively the latter approach, in which the parameter α_1 refers to the size of the Gaussian kernel that is used to Gaussian-filter the spectrum. The decomposition of data sets that show a mix of both narrow and broad linewidths likely requires an additional smoothing parameter α_2 to yield good fitting outcomes. The fitting procedure using a single or two smoothing parameters is referred to as one-phase or two-phase decomposition, respectively.

It is essential for the best performance of the derivate spectroscopy technique to find the optimal smoothing parameters for the original spectra. The GaussPy algorithm achieves this via an incorporated supervised machine learning technique, for which the user has to supply the algorithm with a couple of hundred well-fit spectra, from which the algorithm then deduces the best smoothing parameters.

More specifically, GaussPy uses the gradient descent technique – a first-order iterative optimisation algorithm – to find values for α_1 and α_2 that yield the most accurate decomposition of the training set. This accuracy is measured via the F_1 score, which is defined as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{1}$$

where precision refers to the fraction of fit components that are correct and recall refers to the fraction of true components that were found in the decomposition of the training set with guesses for α_1 and α_2. See Lindner et al. (2015) for more details on how the training set is evaluated.

2.2. ^{13}CO data

We test GaussPy+ on data from the Boston University–Five College Radio Astronomy Observatory GRS (Jackson et al. 2006) that we downloaded from the online repository of the Boston University Astronomy Department[1]. This survey covered the lowest rotational transition of the ^{13}CO isotopologue with an angular resolution of 46″, a pixel sampling of 22″, and a spectral resolution of 0.21 km s^{-1}. The values in the GRS data set are given in antenna temperatures, which we converted to main beam temperatures by dividing them with the main beam efficiency of $\eta_{\rm mb} = 0.48$[1].

The lowest rotational transition of ^{12}CO can show strong self-absorption that can severely affect the lineshape (e.g. Hacar et al. 2016). A decomposition of the spectrum can therefore

[1] https://www.bu.edu/galacticring/new_data.html

lead to incorrect results, as strong self-absorption features can be erroneously fit with multiple components. We do not expect such strong opacity effects for ^{13}CO observations, but it can still become optically thick in very bright regions (e.g. Hacar et al. 2016). Optical depth effects are also expected for ^{13}CO observations of nearby regions or observations with high spatial resolution, for which the opacity effects are not smoothed out as in a larger physical beam. For the moderate spatial resolution of the GRS survey one would thus not expect severe optical depth effects, even though the analysis by Roman-Duval et al. (2010) suggests that opacity effects do indeed have to be taken into account for the GRS data set. In this work we will not address the potential problems of optical depth effects or self-absorption on the decomposition results, but we caution that fitting ^{13}CO peaks with Gaussian components might lead to incorrect fits of multiple components for a single self-absorbed emission line in case regions of optically thick ^{13}CO are expected to be present in the data set.

Even though in this paper we demonstrate the functionality of GaussPy+ only for a small GRS test field, we used the entire data set in testing and developing the algorithm. A forthcoming paper will present and discuss the decomposition results of GaussPy+ for the whole GRS data set and will also discuss the effects and implications of possible optical depth effects for the ^{13}CO emission and the fitting results.

3. New decomposition package: GaussPy+

The methods and procedures described in this section are all either new preparatory steps for, or extensions to, the original GaussPy algorithm. They aim at either improving the performance of GaussPy or automating required preparatory steps. Figure 1 presents a schematic outline of the GaussPy+ algorithm.

The main shortcomings of the original GaussPy algorithm that we aim at improving are: (i) the noise values are calculated from a fixed fraction of channels in the spectrum, which is not ideal in cases where signal peaks might occur at all spectral channels; (ii) the user has to supply the training set; (iii) there is no in-built quality control of the fit results; (iv) the fit of each spectrum is treated independently of its neighbours. The last point might lead to drastic jumps between the number of Gaussian components between neighbouring spectra. From a physical point of view we would not expect such component jumps for resolved extended objects with sizes larger than the beam. Moreover, observations are often Nyquist sampled, in which case the beam size or resolution element is larger than the pixel size. Therefore neighbouring pixels will contain part of the same emission, which also introduces coherence between the number of components between neighbouring spectra.

To develop a fitting algorithm that improves on the above points, we have included in GaussPy+: (i) automated preparatory steps for the noise calculation and creation of the training set (see Sect. 3.1); (ii) automated quality checks for the decomposition, some of which can be customised by the user and are used to flag and refit unphysical or unwanted fit solutions (see Sect. 3.2); (iii) automated routines that check the spatial coherence of the decomposition and in case of conflicting results try to refit the spectrum based on neighbouring fits (see Sect. 3.3).

In the following, the GaussPy+ algorithm is described in detail, following the outline presented in Fig. 1. A description of GaussPy+ keywords including their default values and other symbols used throughout the paper can be found in the Appendix F.2.

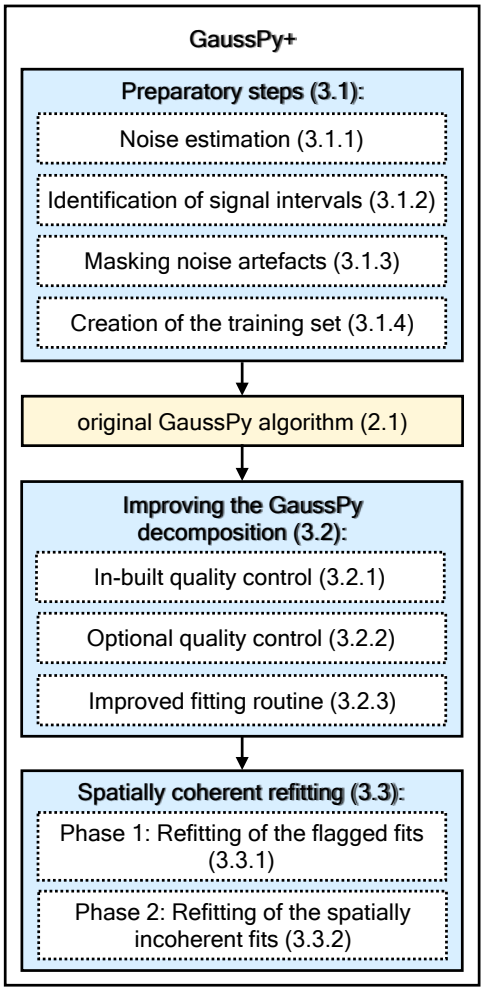

Fig. 1. Schematic outline describing new automated methods and procedures included in GaussPy+, along with corresponding sections in this paper.

3.1. Preparatory steps

3.1.1. Noise estimation

The original GaussPy algorithm either requires the user to supply noise estimates or uses a certain fraction of the spectral channels, assumed to contain no signal, for the noise estimation. However, the latter approach only leads to correct noise estimates if one can exclude the presence of signal peaks in the spectral channels used to calculate the noise.

A reliable noise estimation is of fundamental importance for the decomposition – key steps of GaussPy depend on the noise value, and also the new procedures in GaussPy+ rely on accurate noise estimation: the signal-to-noise (S/N) threshold is used for the initial guesses for the number of components in GaussPy and the noise estimate is needed for the quality assessments of the fit components in GaussPy+. Because of the key role of the noise, we developed a new, automated noise estimation routine as a preparatory step for the decomposition.

The fundamental, underlying assumptions in our noise estimation process are: i) the noise statistics are Gaussian, meaning "white noise"; ii) the spectral channels are uncorrelated; and iii) the noise is fluctuating around a baseline of zero. These assumptions enable us to make use of the number statistics of negative and positive channels in the noise estimation process (elaborated further in item 1 below).

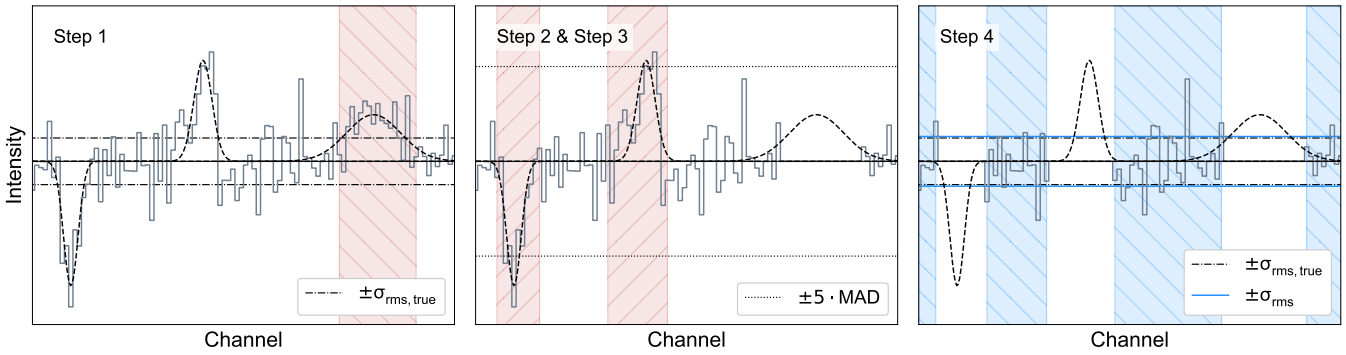

Fig. 2. Illustration of our automated noise estimation routine for a mock spectrum containing two signal peaks and a negative noise spike. Hatched red areas indicate spectral channels that are masked out and hatched blue areas indicate all remaining spectral channels used in the noise calculation. *Right panel*: Comparison of the true noise value ($\sigma_{\rm rms,\,true}$; black dash-dotted lines) with the noise value estimated by our automated routine ($\sigma_{\rm rms}$, blue solid lines). See Sect. 3.1.1 for more details.

In the following, we describe how our automated noise estimation proceeds. The overall idea is to identify the spectral channels that can be used for noise estimation and maximise their number. To do so, the routine has to identify as many channels as possible that are free from signal and instrumental effects. We demonstrate the steps of the process for a mock spectrum in Fig. 2. The spectrum has 100 channels and contains two challenging features for the noise estimation: a negative noise spike in the first few channels and a broad signal feature with a maximum amplitude of two times the root-mean-square noise $\sigma_{\rm rms}$.

The steps to estimate the noise are the following:

1. Mask out broad features in the spectrum; such features are likely to be either positive signal or instrumental artefacts due to, for instance, insufficient baseline corrections. Given our basic assumptions (see above), spectra containing (only) noise have the same number of positive and negative spectral channels on average. We can use this fact to determine the probability of having a number of consecutive positive or negative channels in the spectrum, meaning the probability that a given feature is noise (instead of a signal peak or an artefact). This provides a mean to mask out features that are likely not noise. We estimate the probability that a consecutive number of positive or negative channels is due to noise with a Markov chain (see Appendix A for more details). We then mask out all features whose probability to be caused by noise is below a user-defined threshold $P_{\rm Limit}$. For the example spectrum in Fig. 2 we used the default value of $P_{\rm Limit} = 2\%$[2]. From the Markov chain calculations for a spectrum with 100 spectral channels we get that all features with more than twelve consecutive positive or negative channels have a probability less than $P_{\rm Limit} = 2\%$ to be the result of random noise fluctuations and are thus masked out (one feature; see left panel in Fig. 2). In many cases, peaks will still continue on both sides of the identified consecutive channels. To take this into account, the user can specify how many additional channels $N_{\rm pad}$ will be masked out on both sides of the identified feature. In the example spectrum (Fig. 2) we set $N_{\rm pad} = 2$, so two additional channels on both sides of the identified features got masked out.

2. Use the unmasked negative channels to calculate their median absolute deviation (MAD). We use the MAD statistic because it is very robust against outliers in the data set, such as noise spikes. The relationship of MAD to the standard deviation σ is MAD $\approx 0.67\sigma$. We restrict the calculation of the MAD to spectral channels with negative values, since the positive channels can still contain multiple narrow high signal peaks that were not identified in the previous step. Narrow negative spikes will still be included in this calculation but we assume that their presence is sufficiently uncommon so that they will not significantly affect the estimation of the MAD.

3. Identify intensity values with absolute value higher than $5 \times$ MAD. We then mask out all consecutively negative or positive channels of all features that contain an intensity value higher than $\pm 5 \times$ MAD[3]. The mask is extended again on both sides by the user-defined number of channels $N_{\rm pad}$. In the example spectrum, two regions are masked out in this step (middle panel in Fig. 2), corresponding to the second positive signal and the negative noise spike in the spectrum.

4. Use all remaining unmasked channels to calculate the rms noise. The example spectrum is left with 51 unmasked channels (blue hatched areas in the right panel of Fig. 2) from which the noise is estimated.

The right panel of Fig. 2 shows the determined $\sigma_{\rm rms}$ value (blue solid line), which is very close to the true value $\sigma_{\rm rms,\,true}$ (black dash-dotted line) that was used to generate the noise. This example represents a case in which estimating the noise from a fixed fraction of channels in the beginning or the end of the spectrum would obviously not work well. Had we estimated the noise with the first or last 20% of spectral channels, we would have overestimated the noise by factors of 2.3 and 1.3, respectively.

In case of residual continuum in the spectrum or signal peaks covering almost all of the spectral channels, the noise estimation can be skewed and biased towards low values. To circumvent this problem, the user can supply an average noise value $\langle\sigma_{\rm rms}\rangle$ or calculate $\langle\sigma_{\rm rms}\rangle$ directly from the datacube by randomly sampling a specified number of spectra throughout the cube. This $\langle\sigma_{\rm rms}\rangle$ value is adopted instead of the value resulting from steps 1–4 above, if (1) the fraction of spectral channels available for noise calculation from steps 1–4 is less than a user-defined value (default: 10%), and (2) the noise value resulting from steps 1–4 is less than a user-defined fraction of $\langle\sigma_{\rm rms}\rangle$ (default: 10%)[4]. If no $\langle\sigma_{\rm rms}\rangle$ value is supplied or calculated, the spectra that do not

[2] $P_{\rm Limit} = 2\%$ yielded good results in our tests and represents a good compromise between excluding signal peaks with low-amplitude values from the noise estimation without masking out too many noise features.

[3] We choose $\pm 5 \times$ MAD as our threshold because it is a good trade-off: lower thresholds would remove too many valid noise peaks and higher thresholds could miss too many narrow signal peaks with low amplitude values.

[4] The default values are deliberately set to low values to target only spectra with anomalies such as severe baseline effects.

reach the required minimum fraction of spectral channels for the noise calculation are masked out.

We performed thorough testing of the effects of random noise fluctuations on our noise estimation routine. A detailed description of the tests is given in Appendix B.2. The tests showed that the routine is robust in typical situations (pure white noise, white noise with signal, white noise with signal and negative noise spikes, white noise with weak signal and negative noise spikes).

3.1.2. Identification of signal intervals

If a spectrum contains a high fraction of signal-free spectral channels, goodness of fit calculations can be completely dominated by noise and their value thus may decrease to acceptable numbers even in cases for which the fit did not work out. Therefore, we added a routine to GaussPy+ that automatically identifies intervals of spectral channels that contain signal; goodness of fit calculations are subsequently restricted to these channels[5]. However, the fitting itself is still performed on all spectral channels.

As part of our automated noise estimation routine (outlined in Sect. 3.1.1) we already identify consecutive positive spectral channels that can potentially contain signal (see Fig. 2). We identify these features as signal intervals using a criterion that takes both the S/N and the extent of the feature into account (this criterion is described in more detail in Sect. 3.2.1). For spectra that contain a single narrow peak, only a small fraction of the spectrum might be identified as signal interval. To ensure that for such cases the goodness of fit values are not artificially increased by a too small number of spectral channels, the user can require that a minimum number of spectral channels be adopted as signal intervals ($N_{\rm min}$; default value: 100). If the signal intervals identified in the spectrum contain fewer channels than required by $N_{\rm min}$, the size of all individual signal intervals identified in the spectrum is incrementally increased on both sides by $N_{\rm pad}$, until $N_{\rm min}$ is reached. This incremental padding will not include regions masked out as negative noise spikes (see next section). If no signal intervals could be identified in the spectrum, all channels are used for goodness of fit calculations, even though it is unlikely in this case that there are peaks in the spectrum that will be fit. We tested the performance of the signal interval identification on synthetic spectra and found that it is able to reliably determine weak and strong signal peaks without being sensitive to smaller peaks caused by random noise fluctuations (see Appendix B.3).

3.1.3. Masking noise artefacts

Spectra can sometimes contain negative noise spikes, which can bias the goodness of fit calculations. In principle, candidate regions with negative noise spikes are already identified in the automated noise estimation routine (Sect. 3.1.1). However, since the MAD-based threshold is set to a conservative value to exclude most of the narrow signal peaks from the noise estimation, it will also incorrectly remove an increased fraction of regular noise peaks or false positives (see the distribution for sample A of our synthetic spectra in Fig. B.2). To avoid such contamination of identified noise artefacts by regular noise peaks, the user can decide below which negative value features get masked out by supplying the value in terms of the S/N ($S/N_{\rm spike}$; default value: 5). Setting $S/N_{\rm spike} = 5$ means that any region of consecutive negative channels that contains at least one channel with a value lower than $-5 \times \sigma_{\rm rms}$ will get masked out. We tested the performance of the identification of noise spikes on synthetic spectra and found that we are able to reliably mask such features out (see Appendix B.4).

3.1.4. Creation of the training set

As described in Sect. 2.1, GaussPy needs a sample of already decomposed spectra to determine the smoothing parameters used in the decomposition. In principle, this training set can be composed of synthetic spectra whose noise and emission properties are similar to the data set the user wants to analyse. Another approach is to use actual spectra from the data set for which the user can supply a reliable decomposition. We added a routine to GaussPy+ that adopts the latter approach and automatically decomposes a user-defined number of spectra from the data set. These decomposition results are then supplied to GaussPy, which uses its machine learning functionality to infer the most appropriate smoothing parameters for the data set.

In principle, we could use GaussPy itself to construct decompositions for this training sample by first guessing the smoothing parameters and correcting them accordingly to get good fitting results. However, since it can be tricky and time-consuming to guess the correct smoothing parameters for a data set we added a routine to GaussPy+ that decomposes spectra for a training set.

Our key requirement for this decomposition routine was that it should be able to produce high quality fits for a small subset of the data set. We recommend to use training set sizes of about 200−500 decomposed spectra, as these should already give very good values for the smoothing parameter. In principle also larger training sets can be created, but users should be aware that in this case it can become time-consuming to train GaussPy, as it might be necessary to use different starting values for the smoothing parameters α_1 and α_2 to make sure that the search for optimal smoothing parameters explored the parameter space properly and did not get stuck in a local minimum (see Fig. 3 in Lindner et al. 2015). Training sets containing <200 spectra bear the risk of higher uncertainties for the resulting smoothing parameter values, as incorrectly fitted features in the training set may have a large negative impact on the F_1 score. While deviations of the smoothing parameters from the optimal values will impact the decomposition with GaussPy, the improved fitting (Sect. 3.2.3) and spatially coherent refitting (Sect. 3.3) routines in GaussPy+ should be able to mitigate such incorrect or insufficient decomposition results. Thus the decomposition of GaussPy+ also has a bigger margin for deviations of the smoothing parameters from their optimal values than the decomposition with GaussPy, which allows the use of smaller training set sizes.

For the decomposition of the spectra for the training set we use the SLSQP optimisation algorithm and least squares statistic (SLSQPLSQFitter) of the astropy.modeling package, which produced good fits to the spectra in our tests of the routine. We have to supply the SLSQPLSQFitter routine with initial guesses for possible Gaussian fit components. We determine the number of Gaussian fit component candidates and their initial guesses by estimating how many local positive extreme values or maxima are present in the spectrum. To find these local extreme values, we first set all values to zero that are below a user defined S/N threshold ($S/N_{\rm min}$; default value: 3). The remaining positive values are then searched for local maxima. We define a local maximum as a peak that exceeds all values for a minimum number of neighbouring spectral channels on either side of the peak. This required minimum number of spectral channels on either side

[5] With the exception of one normality test that we perform over the whole channel range. See Sect. 3.2.1 and Appendix D.

can be defined by the user with the ξ parameter (default value: 6). To infer a good value for ξ, users are advised to check the shape of the components present in the spectra or make a test run for a small training set size and check the decomposition results (routines for plotting the spectra, decomposition results, and residuals are contained in our method).

Our routine then tries to fit a number of Gaussian components according to the inferred peaks of local positive maxima present in the spectrum. We therefore likely start out with the maximum possible number of Gaussian fit components for the spectrum. The individual fit parameters of each Gaussian parameter (amplitude a_i, mean position μ_i, standard deviation σ_i) are then checked for the following criteria:

- amplitude $a_i \geq S/N_{\rm min} \times \sigma_{\rm rms}$
- significance $\mathcal{S}_{\rm fit} \geq \mathcal{S}_{\rm min}$. See Sect. 3.2.1 for more information about this criterion.
- the standard deviation σ_i is between user defined limits: $\sigma_{\rm min} \leq \sigma_i \leq \sigma_{\rm max}$, where the limits for the standard deviation can be specified in terms of the full width at half maximum (FWHM) given as fraction of channels ($\Theta_{\rm min}$ and $\Theta_{\rm max}$; default values: 1. and `None`, respectively).

We do not check if components are blended in the creation of the training set. If any of the individual Gaussian components do not satisfy all these requirements, their values are removed from the list of initial guess values and a new fit is performed. These checks and the subsequent refitting is performed as long as some of the individual Gaussians are not satisfying all the criteria or there are no more Gaussian parameters remaining. In the process of refitting a spectrum we do not add any new fit component candidates.

We thoroughly tested the routine outlined in this section on samples of synthetic spectra and found that it is able to create reliable training sets that allow inferring optimal smoothing parameters with GaussPy (see Appendix B.5). However, we did not optimise the SLSQPLSQFitter decomposition routine for speed, which is why we recommend to only use this fitting technique for the creation of training sets. See Appendix C.1 for a quantitative comparison between the SLSQPLSQFitter fitting routine and the improved fitting routine of GaussPy+ (Sect. 3.2.3) in terms of execution time and performance of the decomposition.

3.2. Improving the GaussPy decomposition

3.2.1. In-built quality control

In this section, we describe the automated quality checks for the decomposition results we implemented in GaussPy+. Figure 3 illustrates how the in-built quality controls for individual fit components are used to improve the fit results for a spectrum. If individual Gaussian components do not satisfy one of the criteria outlined in Fig. 3 they get discarded. This refitting procedure using the in-built quality controls is applied to all fit solutions obtained in the decomposition steps of GaussPy+ (Sects. 3.2.3–3.3.2). The corrected Akaike information criterion and normality tests for the normalised residual are used to decide between different fit solutions of a spectrum and to assess whether a spectrum needs to be refitted, respectively. See also Appendix C.4 for a discussion about the performance of the in-built quality controls on the decomposition results of the synthetic spectra (Sect. 4) and the GRS test field (Sect. 5).

FWHM value. If users supply limits for the lower and upper values of the FWHM ($\Theta_{\rm min}$ and $\Theta_{\rm max}$, respectively) all fitted components with FWHM values outside this defined range are

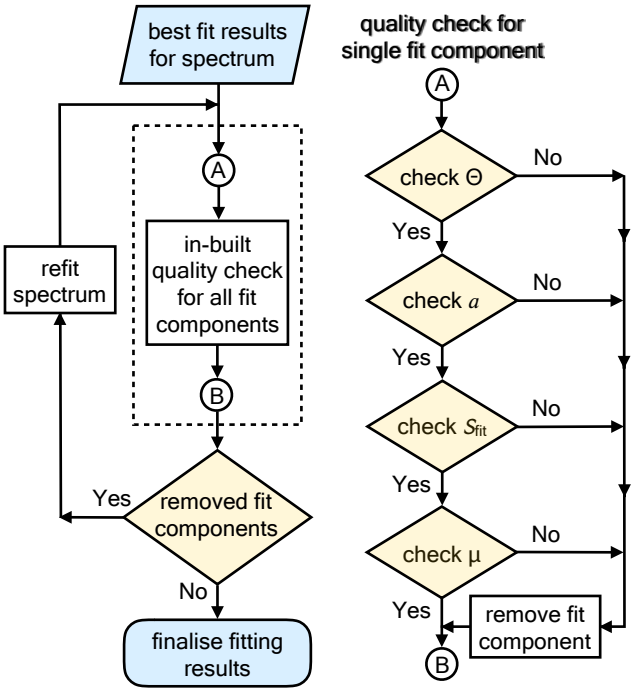

Fig. 3. Flowchart outlining how in-built quality controls from Sect. 3.2.1 are applied to fit results of a spectrum.

removed. In the GaussPy+ default settings $\Theta_{\rm min} = 1$, which means that the FWHM value of a fit component has to be at least one spectral channel. By default, GaussPy+ does not set any value for $\Theta_{\rm max}$. Users are advised to use the $\Theta_{\rm max}$ parameter with caution, as it can produce artefacts in the decomposition, such as an increase of the number of fit components whose widths are close to or exactly at this predefined upper limit.

Signal-to-noise ratio. The user-defined minimum S/N of $S/N_{\rm min}$ (default value: 3) is in the default settings used as the S/N threshold for: (i) the original spectrum and the second derivative of the smoothed spectrum in the GaussPy decomposition (i.e. $\rm SNR_1 = S/N_{min}$ and $\rm SNR_2 = S/N_{min}$); (ii) the search for new peaks in the residual (Sect. 3.2.3); (iii) the search for negative residual peaks (i.e. $S/N_{\rm min,\,neg} = S/N_{\rm min}$, Sect. 3.2.2); (iv) the decomposition of the training set (Sect. 3.1.4). These parameters can all be set to different values from each other to improve the fitting results but we advise to keep them at the same value for consistency.

The minimum required amplitude values of Gaussian fit components are determined by the $S/N_{\rm min,\,fit}$ parameter, whose default value is half the value of $S/N_{\rm min}$. All Gaussian components with $a_i < S/N_{\rm min,\,fit} \times \sigma_{\rm rms}$ will be removed from the fit. We recommend setting $S/N_{\rm min,\,fit} < S/N_{\rm min}$ to allow fit components to also converge to an amplitude value that is below $S/N_{\rm min}$, as such smaller unfit peaks might otherwise negatively influence the fitting results of higher signal peaks that are close by (cf. panel b in Fig. 5). A smaller value for $S/N_{\rm min,\,fit}$ can also be beneficial if it cannot be excluded that some of the spectra might be affected by insufficient baseline subtraction effects, in which case the spectra would show a very broad but low-amplitude feature that can stretch over all spectral channels. However, the $S/N_{\rm min,\,fit}$ can also be supplied by the user directly in case the default settings do not yield good results.

Significance. To further check the validity of fitted Gaussian components, we use the integrated area of the Gaussian as a

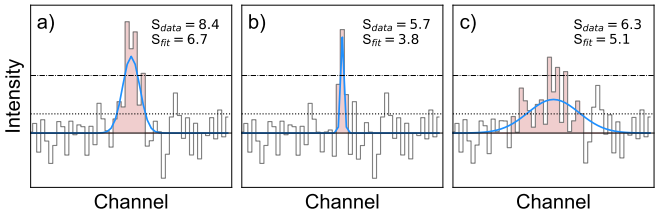

Fig. 4. Calculation of the significance for Gaussian fit components ($\mathcal{S}_{\rm fit}$; blue dashed lines) or peaks in the data ($\mathcal{S}_{\rm data}$; red-shaded areas). The dotted and dash-dotted lines indicate the $\sigma_{\rm rms}$ value and S/N thresholds of 3, respectively.

proxy for the significance of the component. Assuming that the noise properties are Gaussian (white noise), random noise fluctuations are more likely to cause narrow features with a higher amplitude than broader, extended features with a lower amplitude. With this significance criterion we basically require that the fit components, or data peaks, have either very high intensity or are extended over a wide channel range.

The integrated area W_i of a Gaussian component can be calculated from its amplitude and FWHM value Θ in terms of spectral channels:

$$W_i = a_i \cdot c \cdot \sqrt{2\pi} \tag{2}$$

with the parameter c defined as

$$c = \frac{\Theta_i}{2\sqrt{2\ln 2}}. \tag{3}$$

For the calculation of the significance value, we compare the area of the Gaussian component to the integrated $\sigma_{\rm rms}$ interval of the channels from the interval $\mu_i \pm \Theta_i$, which gives a good approximation for the total width of the emission line:

$$\mathcal{S}_{\rm fit} = \frac{W_i}{\sqrt{2\cdot\Theta_i}\cdot\sigma_{\rm rms}}. \tag{4}$$

The $\mathcal{S}_{\rm fit}$ value is then compared to a user-defined minimum $\mathcal{S}_{\rm min}$ (default value: 5) and the Gaussian component is discarded if $\mathcal{S}_{\rm fit} < \mathcal{S}_{\rm min}$. This check helps to remove noise peaks that might have been fit and were not discarded in the checks for the S/N.

We can use the significance parameter also as a threshold to decide whether peaks in the data are valid signal peaks. For this estimate of the significance ($\mathcal{S}_{\rm data}$), we first search for peaks in the data above the user-defined S/N threshold and then compare the integrated intensity of all positive consecutive channels belonging to this feature to the integrated $\sigma_{\rm rms}$ interval of the channels spanned by this feature. We discard the peak as a valid signal feature if $\mathcal{S}_{\rm data} < \mathcal{S}_{\rm min}$.

Figure 4 illustrates this significance measure for three different cases. Panel a shows a signal peak and fit component that is very likely corresponding to a true signal, with the significance measures for the data peak and the fit both above the critical default value of 5. Panel b shows a data peak with narrow linewidth that might be caused by random fluctuations of the noise. The $\mathcal{S}_{\rm data}$ value of this feature passes the threshold value $\mathcal{S}_{\rm min} = 5$, but the depicted Gaussian fit component for this data feature only has a $\mathcal{S}_{\rm fit}$ value of 3.8. This low $\mathcal{S}_{\rm fit}$ value would cause the algorithm to reject this fit component even though its peak has a high S/N of about 5. Panel c shows a broader feature, which has only low S/N values. However, since this feature is spread over more spectral channels than the feature shown in panel b, we would accept it based on its $\mathcal{S}_{\rm data}$ value. With the default settings of GaussPy+ we would also keep the depicted fit component. As already mentioned in Sect. 3.2.1, it can be beneficial to keep Gaussian components with such low S/N in the decomposition results, as to not negatively influence the fitting of nearby data peaks (cf. panel b in Fig. 5).

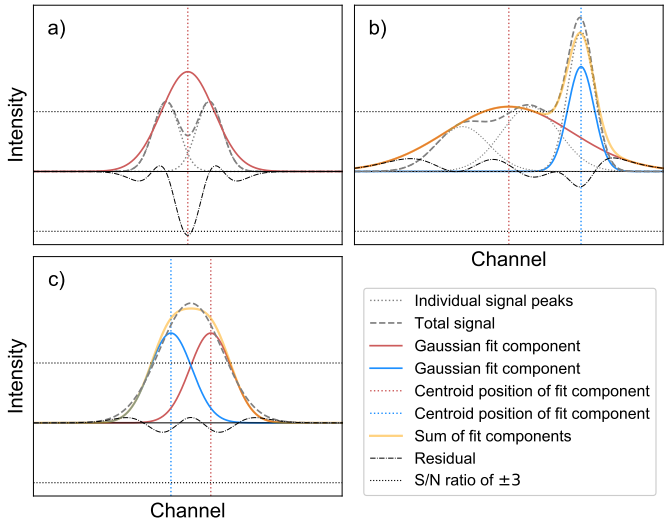

Fig. 5. Optional criteria used to flag fits in the improved fitting routine and in the spatially coherent refitting stage: (*a*) negative residual features introduced by the fit, (*b*) broad components, (*c*) blended components.

For a fitted feature or signal peak containing $N_{\rm feat}$ spectral channels, the $\mathcal{S}_{\rm min}$ parameter implies an average S/N of $\langle S/N\rangle$

$$\langle S/N\rangle = \frac{\mathcal{S}_{\rm min}}{\sqrt{N_{\rm feat}}}. \tag{5}$$

Users can apply this relation to judge which value for $\mathcal{S}_{\rm min}$ is most suitable for their data set. For the default value of $\mathcal{S}_{\rm min} = 5$, Gaussian fits or signal peaks spanning 4 or 9 spectral channels would require $\langle S/N\rangle$ values across the feature of 2.5 and ~1.7, respectively. See Appendix C.3 for a discussion about the effects a variation of the $S/N_{\rm min}$ and $\mathcal{S}$ parameters has on the decomposition results.

Mean position outside channel range or signal intervals. All Gaussian components whose mean positions μ_i are outside the channel range $[0, N_{\rm chan}]$ are automatically discarded from the fit. If the mean position of a fit component is located outside the estimated signal intervals (Sect. 3.1.2), we check the significance value of the fitted data peak $\mathcal{S}_{\rm data}$ (Sect. 3.2.1). We discard the corresponding fit component, if $\mathcal{S}_{\rm data}$ is smaller than the user-defined threshold for the significance $\mathcal{S}_{\rm min}$.

Estimation of the goodness of fit. When we fit a model to data whose errors are Gaussian distributed and homoscedastic, we can arrive at a good fit solution by minimising the chi-squared (χ^2), which is defined as the weighted sum of the squared residuals:

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - Y_i)^2}{\sigma_{\rm rms}^2}, \tag{6}$$

with y_i and Y_i denoting the data and fit value at channel position i, respectively. The reduced chi-square ($\chi^2_{\rm red}$) value is often used as an estimate for the goodness of fit, since it also takes the sample size (in our case the number of spectral channels) and number of

fit parameters into account. $\chi^2_{\rm red}$ is defined as the chi-squared per degrees of freedom:

$$\chi^2_{\rm red} = \frac{\chi^2}{N-k}, \tag{7}$$

with N being the sample size (in our case this corresponds to the number of considered spectral channels) and k denoting the degrees of freedom, which in the case of a Gaussian decomposition would be three times the number of fitted Gaussian components. It thus may seem straightforward to use the $\chi^2_{\rm red}$ value to judge whether all signal peaks in a spectrum were fitted, as one would expect $\chi^2_{\rm red} \sim 1$ in this case. However, as Andrae et al. (2010) pointed out, in case of non-linear models such as a combination of Gaussian functions, the exact value for k cannot be reliably determined and can vary between 0 and $N-1$ and need not even stay constant during the fit. The $\chi^2_{\rm red}$ estimate is thus not the best metric to decide between different fit solutions for a spectrum[6].

A more suited criterion for model selection is the Akaike information criterion (AIC; Akaike 1973), which aims for a compromise between the goodness of fit of a model and its simplicity, by penalising the use of a large number of fit components that do not contribute to a significant increase in the fit quality. The AIC is defined as

$$\mathrm{AIC} = 2k - 2\ln(\hat{L}), \tag{8}$$

with $\hat{L}$ being the maximum value of the likelihood function for the model. If the parameters of a model are estimated using the least squares statistic – as in our case – the AIC is given as[7]:

$$\mathrm{AIC} = N \cdot \ln\left(\frac{\sum_{i=1}^{N}(y_i - Y_i)^2}{N}\right) + 2k. \tag{9}$$

For small sample sizes, the AIC tends to select models that have too many parameters, meaning that it will overfit the data. Therefore a correction to the AIC was introduced for small sample sizes[8] – the corrected Akaike information criterion (AICc; Hurvich & Tsai 1989) that is defined as:

$$\mathrm{AICc} = \mathrm{AIC} + \frac{2k^2 + 2k}{N-k-1}. \tag{10}$$

We employ the AICc as our model selection criterion to decide between different fit solutions. The AICc value is meaningful only in relative terms, that is if the AICc values for two different fit solutions are compared with each other. In such a comparison, the fit solution with the lower AICc value is preferred as it incorporates a better trade-off between the used number of components and the goodness of fit of the model.

As an alternative to goodness of fit determinations based on the $\chi^2_{\rm red}$ value, Andrae et al. (2010) suggest to check whether the normalised residuals show a Gaussian distribution. We implement this additional goodness of fit criterion in GaussPy+ by subjecting the normalised residuals to two different normality tests: the Scipy.Stats.Kstest, which is a two-sided Kolmogorov-Smirnov test (Kolmogorov 1933; Smirnov 1939); and the Scipy.Stats.Normaltest, which is a based on D'Agostino (1971) and D'Agostino & Pearson (1973) and analyses the skew and kurtosis of the data points. Both of these normality tests examine the null hypothesis that the residual resembles a normal distribution, as would be expected if we are only left with Gaussian noise after we subtract the fit solution from the data. If the p-value from one of these test is less than a user-defined threshold (default: 1%), we reject the null hypothesis and will try to refit the spectrum. We found that the combined results of these two hypothesis tests allows a robust conclusion of whether the residual is consistent with Gaussian noise (see Appendix D for more details).

3.2.2. Optional quality control

The automated checks described in the previous section should already help to reject many fit components that are not satisfying our quality requirements. However, depending on the data set, the user might want to flag and refit the decomposition based on more criteria, which we outline in this section[9]. The quality criteria discussed in this section are used to flag and refit spectra in the improved fitting and spatially coherent refitting routines discussed in Sects. 3.2.3 and 3.3, respectively[10]. See Appendix C.4 for a discussion about the performance of the optional quality controls on the fitting results of the synthetic spectra (Sect. 4) and the GRS test field (Sect. 5).

Negative peaks in the residual. The first quality check examines negative peaks in the residual, since these can indicate a poor fit. Panel a in Fig. 5 presents a scenario in which a double peaked profile (shown in dashed grey lines) is fit with a single Gaussian component (red line), leading to a significant negative peak in the residual (dash-dotted black line) at the position between the two data peaks. The search for negative peaks in the residual can be controlled by the user with the $S/N_{\rm min,\,neg}$ parameter, which defines the minimum S/N that the negative peak has to have (in the default settings $S/N_{\rm min,\,neg} = S/N_{\rm min}$). To be flagged as a negative residual feature, a negative peak has to satisfy $|y_i - Y_i| \geq S/N_{\rm min,\,neg} \times \sigma_{\rm rms}$, with y_i and Y_i denoting the data and corresponding fit value at channel position i. This requirement takes into account that negative peaks could have already been present in the original spectrum and requires that a significant part of the negative peak was introduced by the fit.

Gaussian components with a broad FWHM. It can occur that a single, broad Gaussian component is fit over multiple peaks in the spectrum, which can be an undesired property. A broad feature can be caused by peaks being close to the noise limit, multiple blended components, or issues in the data reduction, for instance, insufficient baseline corrections or unsubtracted continuum emission. Panel b in Fig. 5 shows an example of a broad component that was incorrectly fit over multiple data peaks without introducing significant residual features as in panel a. This would lead to wrong estimates of the total number of components present in this spectrum, a severe overestimate of the linewidth for the two smaller peaks incorrectly fit with one component, and an underestimate of the amplitude of the rightmost component. The example presented in panel b also highlights why it can be beneficial to set the required minimum S/N threshold for fitted component $S/N_{\rm min,\,fit}$ to lower values than the S/N threshold for data peaks $S/N_{\rm min}$ (see Sect. 3.2.1). If

[6] We thus use maps of the determined $\chi^2_{\rm red}$ values only for qualitative comparisons in Sect. 5.3.

[7] For a derivation of Eq. (9) see e.g. Banks & Joyner (2017).

[8] Burnham & Anderson (1998) recommend to use the corrected AIC instead of the AIC if $N/k < 40$. If the sample size $N \to \infty$, the corrected AIC value converges to the AIC value.

[9] All quality checks or flags in this section can be selected or deselected by the user.

[10] The criterion comparing the number of fit components between neighbouring spectra is only used in the spatially coherent refitting routines.

$S/N_{\mathrm{min,\,fit}}$ were set equal to S/N_{min}, the fit component for the leftmost peak in panel b will get discarded, forcing the fit of a broad component over the two leftmost peaks to minimise the residual.

Unfortunately, it can be difficult to set a maximum allowed FWHM value for the Gaussian components, as the range of expected values in the data may not be known. Setting a strict limit for the maximum FWHM value might also lead to a large number of components which have their linewidth equal to the limiting value. To prevent such an undesired effect, we flag a component as broad if it is broader by a user-defined factor $f_{\Theta,\,\mathrm{max}}$ (default value: 2) than the second broadest fit component. This obviously does not work for spectra with only one Gaussian component fit, but this case is taken into account during the spatially coherent refitting (Sect. 3.3.1).

Another physical cause for the broadening of the lines could be opacity broadening, which is especially relevant for optically thick emission lines such as the ^{12}CO(1-0) rotational transition (Hacar et al. 2016). In case the user expects opacity broadening for a significant number of spectra in the data set, we recommend to not flag or refit broad fit components.

Blended Gaussian components. We define a Gaussian component i as blended with a neighbouring component j, if the distance between their mean positions μ_i and μ_j is less than the minimum required separation μ_{sep}. This minimum required separation is determined by multiplying the lower FWHM value of the two components with a user-defined factor f_{sep}:

$$\mu_{\mathrm{sep}} = f_{\mathrm{sep}} \times \min(\Theta_i, \Theta_j). \tag{11}$$

The default value of f_{sep} is $1/\sqrt{2\ln 2}$. This value was chosen so that the required separation between two identical Gaussian components defaults to two times their standard deviation. If two identical Gaussian fit components are separated by a distance larger than two times their standard deviation, their combined signal would have a local minimum between the two peak positions, which we define as a requirement for well resolved Gaussian fit components. Panel c in Fig. 5 shows a case in which the minimum separation between the peak positions of the two identical Gaussian fit components is not reached. The combined signal of the fit components (shown in orange) shows no local minimum between the peak positions and a single Gaussian component that corresponds to the sum of the two individual components would thus be evaluated as a better fit.

Without additional information from neighbouring spectra it can be very difficult to reliably conclude whether a two-component fit is a better choice than the fit of a single component. If this quality criterion is selected by the user we will therefore always try to replace two blended components with a single bigger component in the improved fitting routine (Sect. 3.2.3), where each spectrum is still treated independently.

Residuals not normally distributed. This flag checks whether the normalised residuals show a Gaussian distribution. We subject the normalised residual to two different tests for normality (see Sect. 3.2.1 for more details), with the null hypothesis that the residual values are normally distributed. We reject this null hypothesis if the p-value of at least one of the normality tests is less than a user-defined threshold (default: 1%), in which case the spectrum gets flagged.

Different number of components compared to neighbouring spectra. This quality criterion compares the number of fitted Gaussian components of a spectrum with its immediate neighbouring spectra. We include the fit solutions of all neighbouring spectra in this comparison, irrespective of whether they were already flagged by another optional quality criterion. There are two conditions for which a spectrum can be flagged by this check:

– The number of components N_{comp} in the spectrum is different by more than a user defined value ΔN_{max} (default value: 1) from the weighted median number of components determined from all its immediate neighbours. For a sequence of n ordered elements $x_1, x_2, \ldots, x_n$ with corresponding positive weights $w_1, w_2, \ldots, w_n$ that sum up to w_{tot}, the weighted median is defined as the element x_k for which $\sum_{i=0}^{k-1} w_i < 0.5 \times w_{\mathrm{tot}}$ and $\sum_{i=k+1}^{n} w_i < 0.5 \times w_{\mathrm{tot}}$. Panel a in Fig. 6 shows the weights we apply to the immediate neighbours, which are inversely proportional to their distance to the central spectrum.

– The spectrum shows differences in N_{comp} towards individual neighbours that exceed a user defined value ΔN_{jump} (default value: 2). We flag a spectrum if these differences occur towards more than N_{jump} (default value: 1) of its neighbouring spectra.

We illustrate this criterion in Fig. 6 for two cases and the default settings of GaussPy+. Panel b shows an instance where the fit solution of the central spectrum shows no component jumps >2 to any of its neighbours. However, we would still flag the central spectrum for its number of fitted components, since it differs by more than ΔN_{max} to the weighted median number of components as inferred from the neighbouring fit solutions (2 components). Panel c shows the opposite case, where the median number of components of 5 is still compatible with the actual number of components but the fit solution of the central spectrum would be flagged as inconsistent with its neighbours as it shows two component jumps >2 with two of its neighbours.

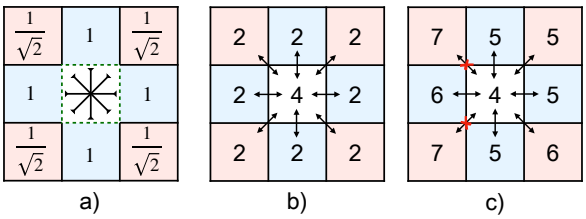

Fig. 6. Illustration of the flagging of spectra based on their number of components with the default settings of our algorithm. Each 3×3 square shows the central spectrum (in white) and the surrounding immediate neighbours coloured according to their weights. *Panel a*: Weights we apply to each neighbouring fit solution to calculate their weighted median. *Panels b and c*: Two cases where the fitted number of components of the central spectrum would be flagged as incompatible with the fitted number of components of their neighbours. See Sect. 3.2.2 for more details.

3.2.3. Improved fitting routine

The improved fitting routine in GaussPy+ aims to improve the fitting results of the original GaussPy algorithm via the use of the quality controls described in Sects. 3.2.1 and 3.2.2. The original version of GaussPy hands over its initial guesses to a least squares minimisation routine without restricting the fitting parameters, apart from a requirement of positive amplitude values. This means that the individual Gaussian components are allowed to freely vary their FWHM and mean positions. Moreover, the number of Gaussian components is set and fixed by the initial guesses, so if GaussPy determined that the fit should contain a certain number of Gaussian components, it will try to fit all those components even if one of them does not contribute to improving the fit or is making the fit worse. This unrestricted fitting can lead to unphysical results or conflicting fit solutions

between neighbouring spectra (see the quality flags discussed in Sect. 3.2.2).

The general idea of our routine is to try to improve the fit based on the residual and optional user-selected quality criteria (Sect. 3.2.2). This improved fitting phase is applied to every spectrum. The steps of this routine proceed as follows (see also Fig. 7):

1. Check the best fit result of GaussPy with the quality criteria outlined in Sect. 3.2.1 (see Fig. 3). All Gaussian components not satisfying any of these criteria are removed from the best fit solution of GaussPy and the spectrum is refit with the remaining fit components; this procedure gets repeated until all of the leftover fit components satisfy all quality criteria.

2. Try to iteratively improve the fit by adding new Gaussian components based on positive peaks in the residual of the best fit solution. Requirements for the acceptance of residual peaks as additional Gaussian component candidates are that: (i) the maximum value of the residual peak is higher than $S/N_{\rm min}$; (ii) the consecutive positive spectral channels of the residual peak satisfy the significance criterion $\mathcal{S}_{\rm data} \geq \mathcal{S}_{\rm min}$ outlined in Sect. 3.2.1. If one or multiple peaks are found in the residual that satisfy these requirements for being new Gaussian component candidates, a refit of the spectrum is performed by adding all of these new candidates. For the refit, the initial Gaussian parameter guesses for the accepted residual peaks are set to: the maximum positive value of the residual peak for the amplitude; the spectral channel containing the maximum positive value of the residual peak for the mean position; the number of consecutive positive channels of the residual peak for the FWHM parameter. After a successful pass of all quality criteria, we adopt the new fit as the new best fit if its AICc value is lower than the AICc value of the previous best fit solution. If a new best fit was chosen, a new iteration with a search for peaks in the residual of the new best fit solution continues. We proceed to the next step if no new positive peaks are found in the residual or no new best fit could be assigned.

3. Optional: Check whether a negative residual feature (Sect. 3.2.2) was introduced by the fit components. This check is only performed if it is the first pass through the main loop or a new best fit was assigned. Negative residual features can be indicative of a poor fit with multiple signal peaks fit by a single broad component. In case such a feature is present, we try to replace the broadest Gaussian component at the place of the residual feature with two narrower components. The initial guesses for the two new narrow components are estimated from the residual obtained if the broad component is removed, which proceeds in a similar way as in the previous step. If the new fit with the two narrow components passes all quality requirements and its AICc value is lower than the AICc value of the current best fit, we will assign it as the new best fit and repeat the search for negative residual peaks. In case multiple negative residual features are present in a spectrum, we deal with the features in order of increasing negative residual values, that is we will first try to replace the Gaussian component causing the residual feature that contains the most negative value. We proceed to the next step if no new negative peaks are found in the residual or no new best fit could be assigned.

4. Optional: Check for broad components (Sect. 3.2.2). If a broad Gaussian component is present we will try to replace it in this step with multiple narrower components. The number of narrow components and their initial parameter guesses are estimated from the residual we get if the broadest component is removed from the fit. If this results in a new best fit we will repeat this procedure with the resulting next broadest component. We proceed to the next step if no excessively broad component is identified anymore, or no new best fit could be assigned.

5. Optional: Check for blended components (Sect. 3.2.2). If this is the case we will try to refit the spectrum by in turn omitting one of the blended components and checking whether the AICc value of the resulting best fit is better than the AICc value of the current best fit. Blended components are omitted in order of increasing amplitude value, that is we will first try to refit the spectrum by excluding the blended component with the lowest amplitude value. If no new best fit is assigned or no blended components are present in the spectrum we exit the improved fitting procedure and finalise the fitting results if the normalised residuals of the best fit solution show a normal distribution, which we verify with two different normality tests (Sect. 3.2.1). If this is not the case, we repeat the whole improved fitting

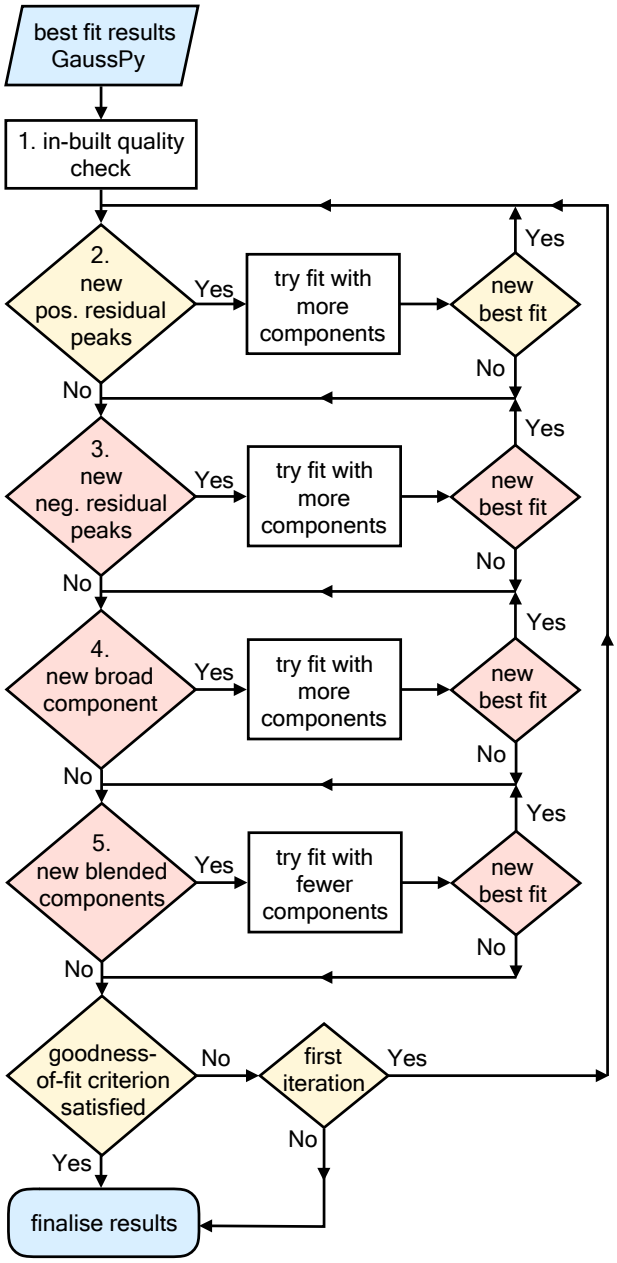

Fig. 7. Flowchart outlining basic steps of our improved fitting routine. The conditional stages in red correspond to optional stages that can be selected by the user. See Sect. 3.2.3 for more details.

procedure beginning with step 2, the search for positive peaks in the residual.

We tested the performance of our improved fitting routine on synthetic spectra and found that it yields a significant improvement in the decomposition compared to the original GaussPy algorithm. In Sect. 4 and Appendix B.6 we give a detailed discussion about the decomposition results for the synthetic spectra.

3.3. Spatially coherent refitting

So far all steps of the fitting routine treated each spectrum separately and independently from its neighbours. Here we describe a new routine that aims to also incorporate the information from neighbouring spectra and tries to refit spectra according to this information. Our routine proceeds iteratively and starts from the fitting results obtained with the method outlined in the previous section (Sect. 3.2.3). This is different to algorithms such as ScousePy, which first start with an averaged spectrum and use its decomposition result to fit the individual spectra. We proceed in a reverse manner: we first produce a sample of high quality fits for each spectrum without regarding their neighbours and then refit them, if it is deemed to be necessary, using the fit solutions of the immediate neighbouring spectra[11].

The spatial refitting proceeds in two phases. In phase 1, we try to improve the fit solutions based on a flagging system, for which the fitting results from the previous stage are checked and flagged according to user-selected criteria. We subsequently try to refit each flagged spectrum with the fit solutions from its neighbours and thereby already introduce a limited form of local spatial coherence. In phase 2, we use a weighting system to try to enforce spatial coherence more globally. We check for the entire data set if the Gaussian components of each spectrum are spatially consistent with the neighbouring spectra, by comparing the centroid positions of the Gaussian components. We then try to refit spectra whose Gaussian components show centroid velocity values that are inconsistent with the fit solutions from neighbouring spectra.

3.3.1. Phase 1: Refitting of the flagged fits

The steps of the first phase of the spatially coherent refitting method are outlined in Fig. 8. The idea here is to determine which of the spectra need to be refit based on flags set by the user. We try to refit all spectra that show features that do not satisfy the quality requirements imposed on the fits (these are also retained as flags indicating bad quality fits in case the spectrum cannot be successfully refit). Depending on the data set, the user might not always want to flag or refit spectra that show one or more of these features. Therefore, all of the following flags can be chosen as required by the user. In the current version of GaussPy+, the following features can be flagged by the user (see Sect. 3.2.2 for more explanation about the flags):

(i) $\mathcal{F}_{\text{neg. res. peak}}$: The presence of negative peaks in the residual.
(ii) $\mathcal{F}_{\Theta}$: Gaussian components with a broad FWHM value. For the spatial refitting we additionally flag a component as broad if it is broader by a user-defined factor ($f_{\Theta,\,\max}$) than the broadest component in more than half of its neighbours.
(iii) $\mathcal{F}_{\text{blended}}$: The presence of blended Gaussian components in the fit.

[11] In the current implementation of GaussPy+ we only consider directly neighbouring spectra, whereas algorithms such as ScousePy allow the user to also include information from larger spatial areas.

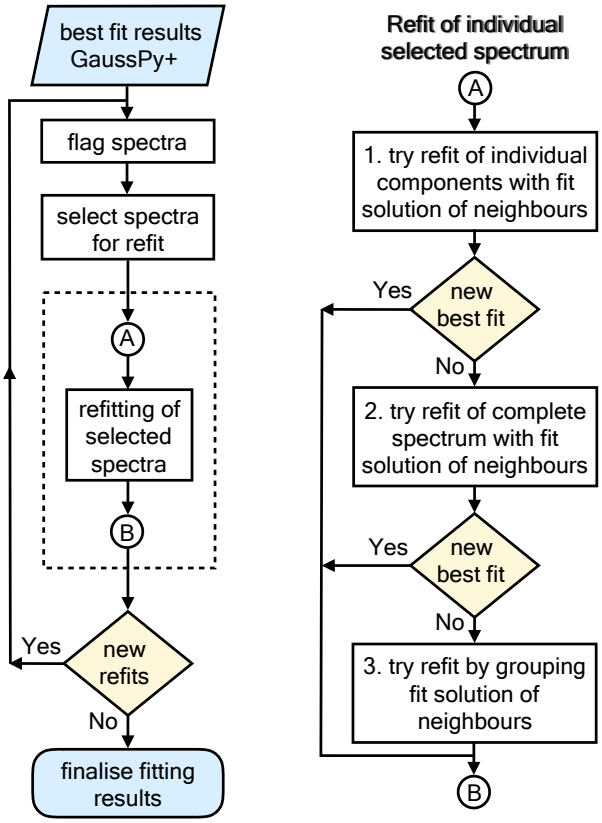

Fig. 8. Flowchart outlining the steps of the first phase of our spatially coherent refitting routine. See Sect. 3.3.1 for more details.

(iv) $\mathcal{F}_{\text{residual}}$: Fits whose normalised residual values do not pass the tests for normality.
(v) $\mathcal{F}_{N_{\text{comp}}}$: The number of components N_{comp} differs significantly from its neighbours.

Flags (i)–(v) are recomputed in each new iteration. We then try to refit each flagged spectrum with the help of one or all of the best fit solutions of its neighbouring unflagged spectra. In the default settings of the algorithm we try to refit all flagged spectra by using fit solutions from unflagged neighbouring spectra. At maximum, this provides eight new different fit solutions for the flagged spectrum (if all of its eight neighbouring spectra are unflagged). If there are multiple unflagged neighbours, they get ranked according to their χ^2_{red} values, and the neighbouring fit solution with the lowest χ^2_{red} value is used first.

It is also possible to only flag fit solutions without refitting them, though this has to be selected by the user. This might be useful, for instance, if users want to exclude neighbouring fit solutions whose normalised residuals did not satisfy the normality tests as templates for the refit but do not want to refit these spectra themselves.

The refitting of an individual flagged spectrum proceeds in the following way (see right part of Fig. 8):

1. Use the fit solutions of unflagged neighbouring spectra to refit individual components of the flagged spectrum. Spectra that are flagged as having negative residual features, broad, or blended components might show a good fit solution apart from the flagged features. Therefore we first try to replace the Gaussian components of such flagged features by using the Gaussian components of neighbouring unflagged fit solutions that cover the same region in the spectrum as new input guesses. The refit attempt is then performed for the entire spectrum by

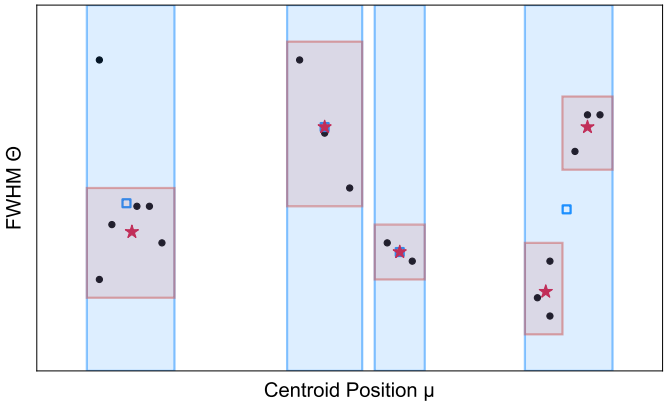

Fig. 9. Illustration of the grouping routine. Black points indicate centroid (μ) and FWHM (Θ) values of Gaussian components from the best fit solutions of unflagged neighbouring spectra. Blue shaded areas indicate the results of the first grouping, in which data points are only separated according to their μ values. Red shaded areas mark the results of the second grouping in which data points are additionally separated according to their Θ values. Blue squares and red stars indicate the initial guesses for the refitting with the first and second grouping approach, respectively.

combining these new initial guesses from a neighbouring fit solution with the remaining fit components of the old fit solution of the spectrum that were not affected by the flagged feature. If multiple regions of a spectrum are flagged with different flags we will try to refit the flagged features in the order of: negative residual feature, broad component, and blended components. As soon as a flagged feature is successfully refit we stop the refitting iteration, even if other flagged features should still be present in the spectrum. We impose no selection criteria on the neighbouring Gaussian components, that is we will in turn use all unflagged neighbouring fit solutions as new initial guesses, starting with the fit solution that has the lowest $\chi^2_{\rm red}$ value. If one of the input guesses of the unflagged neighbours leads to a new improved fit the refitting of the flagged spectrum is successfully terminated, otherwise we proceed with the next step.

2. Use the fit solutions of unflagged neighbouring spectra to refit the complete flagged spectrum. In this step all fit components of a neighbouring spectrum are used as new input guesses for refitting the entire spectrum. We again loop through all unflagged neighbouring fit solutions, starting with the one that has the lowest $\chi^2_{\rm red}$ value. The refitting of the flagged spectrum is successfully terminated as soon as one of the neighbouring fit solutions leads to a new improved fit, otherwise we continue with the next step.

3. Obtain a new set of fit parameters from the fit solutions of all unflagged neighbouring spectra, by grouping and averaging the parameters of all their Gaussian components in a parameter space spanned by the fitted velocity centroid and FWHM values. Figure 9 illustrates how the grouping proceeds. First, the grouping is only performed for the μ values (blue shaded areas). The requirement for group membership is that data points are at maximum located at a distance of $\Delta\mu_{\rm max}$ (default value: 2 channels) from any other point of this group. We require a minimum group membership of two points, which means that single points that do not belong to any group are treated as outliers. The blue points and shaded areas show the new fitting constraints used for the refitting. As initial guesses for the amplitude, FWHM value and centroid position we use the corresponding average values of all the data points belonging to a group. The fitting constraints for the centroid positions are based on the extent of the groups along the μ axis. For each amplitude value we require that it has a positive value and set its maximum limit to the maximum data point in the original spectrum that occurs in the range that encompasses all μ values of this group multiplied by a user-defined factor f_a. FWHM values are not allowed to be smaller than the user-defined parameter $\Theta_{\rm min}$ but there is no upper constraint for their values. If this first grouping approach does not lead to a successful refit, we use a second grouping approach that additionally groups the data points according to their FWHM values (red shaded areas in Fig. 9). A group membership for a data point is established if its μ and Θ values are at maximum located at a distance of $\Delta\mu_{\rm max}$ (default value: 2 channels) and $\Delta\Theta_{\rm max}$ (default value: 4 channels), respectively, from any other point of this group. The points in each group are then averaged in a similar way as for the first grouping approach and supplied as new fit parameters for the refitting.

Grouping only by the centroid values has the advantage that it will try to fit the spectrum with the least amount of components inferred from its neighbours. A disadvantage is that outliers in the FWHM regime can negatively influence the initial fit values. The second grouping approach should be able to deal better with the fidelity of the data even though some of the initial guesses for Gaussian fits could overlap heavily.

For the decision of whether to accept a refit as the new fit solution we define a total flag value $\mathcal{F}_{\rm tot}$ that increases by one for each of the user-selected flags the fit solution does not satisfy. For the proposed new fit solutions, the total flag value increases in addition by one for each flagged criterion that got worse than in the current best fit solution, that is for an increase in the number of blended components or negative residual features, broad components that got broader, smaller p-values for the null hypothesis testing for normally-distributed residuals, and a greater difference in the number of components compared to the neighbouring fit solutions.

In the stage where all spectra were treated independently (Sect. 3.2.3), the decision to accept a fit model was made via the AICc. In the spatial refitting phase this decision is mainly guided by the comparison of the total flag value of the new fit solution ($\mathcal{F}^{\rm new}_{\rm tot}$) with the old best fit solution ($\mathcal{F}^{\rm old}_{\rm tot}$). There are three possible scenarios:

- $\mathcal{F}^{\rm new}_{\rm tot} > \mathcal{F}^{\rm old}_{\rm tot}$. In this case the new fit solution is rejected.
- $\mathcal{F}^{\rm new}_{\rm tot} = \mathcal{F}^{\rm old}_{\rm tot}$. The new fit solution is accepted if its AICc value is smaller than the AICc value for the best fit solution we started out with.
- $\mathcal{F}^{\rm new}_{\rm tot} < \mathcal{F}^{\rm old}_{\rm tot}$. The new fit solution is accepted if the data points of the normalised residual pass the normality tests.

In the last case we have to test whether new fit solutions incorrectly decreased $\mathcal{F}^{\rm new}_{\rm tot}$ by removing valid fit components. For example, both $\mathcal{F}_{\rm blended}$ and $\mathcal{F}_{\Theta}$ could be reduced by one if a broad component is deleted. To prevent such incorrect fit solutions we require that the normalised residual resembles a Gaussian distribution, which we check with two different normality tests (see Sect. 3.2.1). The null hypothesis of normally distributed residual values gets rejected if the p-value is less than a user-defined threshold (default: 1%), in which case we do not accept the new fit solution.

3.3.2. Phase 2: Refitting of the spatially incoherent fits

In the second phase of the spatially coherent refitting, we check for coherence of the centroid positions of the fitted Gaussian components for all spectra. The motivation for this step is that

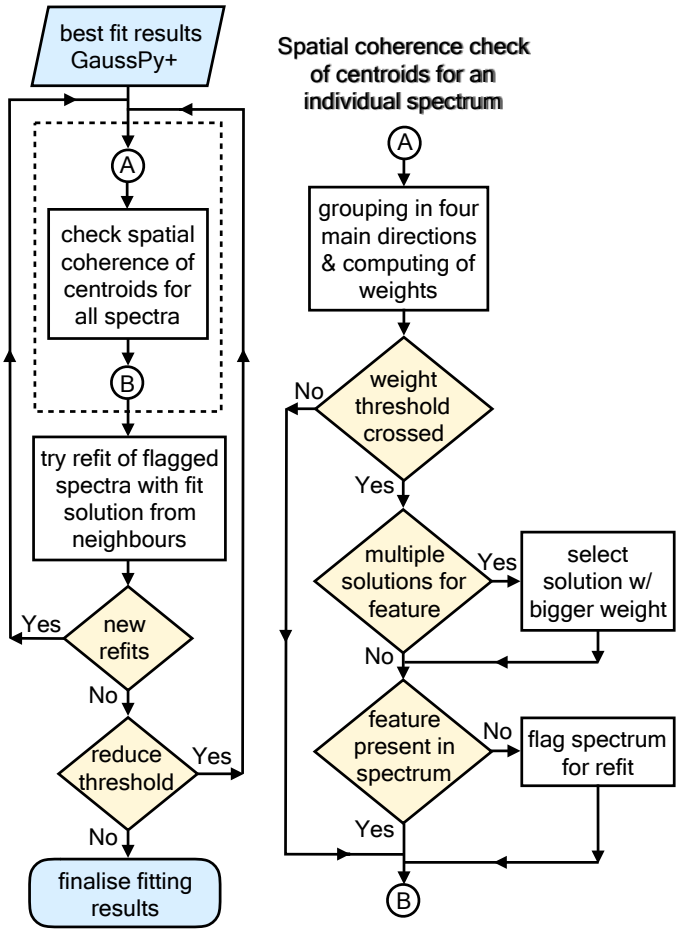

Fig. 10. Flowchart outlining the basic steps of the second phase of the spatial refitting routine. See Sect. 3.3.2 for more details.

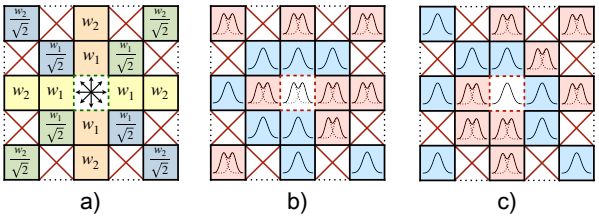

Fig. 11. Illustration of phase two of the spatial refitting routine of GaussPy+. Each 5×5 square shows a central spectrum (in white) and its surrounding neighbours. White squares that are crossed out are not considered. *Left panel*: Principal directions for which we check for consistency of the centroid positions and shows the applied weights w_1 and w_2 attached to the neighbouring spectra. *Middle and right panels*: Two different example cases with simple fits of one and two Gaussian components shaded in blue and red, respectively. Based on the fits of the neighbouring spectra we would try to refit the central spectrum in the first case (*panel b*) with one Gaussian component, whereas the central spectrum in the second case (*panel c*) is already consistent with what we would expect from our spatial consistency check of the centroid positions. See Sect. 3.3.2 for more details.

we would expect coherence in the centroid positions of the fitted Gaussian components for resolved extended objects, especially for oversampled observations where the size of a pixel is smaller than the beam size or resolution element.

The spatial consistency check, in which we determine whether a spectrum should contain Gaussian components in specific spectral ranges based on the fitting results from neighbouring spectra, proceeds in an iterative way. For that, we use 16 neighbours along the 4 main directions (see panel a in Fig. 11)[12]. For simplicity we do not consider the off-diagonal pixels.

Users can specify the ratio of the weight of the closest neighbour (w_1) to the weight of the neighbour located one pixel farther away (w_2) with the parameter $f_w = w_1/w_2$ (default value: 2). In the default settings the contribution of the neighbours is inversely proportional to their distance to the central spectrum (see left panel of Fig. 11). The weights w_1 and w_2 are normalised so that $2w_1 + 2w_2 = 1$, which means that along the horizontal and vertical direction the weights sum up to a value of 1. Setting the parameter f_w to higher values than the default value has the effect of decreasing the contribution of neighbours that are located at a distance of two pixels and thus puts even more emphasis on the closest neighbours. In case the central spectrum has Gaussian components whose centroid positions do not match with what would be expected from the fit results of its neighbouring spectra, we try to refit the spectrum with a better-matching fit solution from one of its neighbours.

In the following, we outline the spatial consistency check of the centroid positions in more detail (see also Fig. 10):

1. Check for a consistent feature in the neighbouring spectra along any of the main directions indicated in the left panel of Fig. 11. For each of the four directions, we group the centroid positions of the fitted Gaussian components as described in section Sect. 3.3.1 and shown schematically in Fig. 9 (blue hatched areas). We perform the grouping in each direction rather than globally to simplify the grouping, which might get too confused if all 16 neighbours are considered together.

2. Compute the total weight $\mathcal{W}_{\rm tot}$ for each group of centroid position data points by summing up the weights of the neighbouring spectra that contributed data points to the group and check if it exceeds a predefined weight threshold $\mathcal{W}$.

3. Check whether the central spectrum has Gaussian components compatible with the required Gaussian components inferred from its neighbours (i.e. all centroid position groups that reached the required weight threshold $\mathcal{W}$). We try to refit the central spectrum with the fit solution from individual neighbours if its Gaussian components are incompatible with the inferred required components.

In the default settings of GaussPy+, the first set of iterations use a weight threshold of $\mathcal{W} = 1 - w_2$; this threshold can only be reached in the horizontal or vertical direction if two immediate spectra and an additional spectrum further out contributed data points to the group, that is show a common feature. The threshold of $\mathcal{W} = 1 - w_2$ is used as long as it leads to new successful refits of spectra. In case no new refits were possible, $\mathcal{W}$ is reduced again by a value of w_2 so that the new threshold is $\mathcal{W} = 1 - 2 \cdot w_2$. This iterative procedure continues until $\mathcal{W}$ gets below a user defined minimum threshold $\mathcal{W}_{\rm min}$ (default value: 0.5).

We only start the refitting procedure after we looped through all spectra of the data set and determined the spatial consistency of the centroid position values for all of them. This means that the fit solutions are not dynamically updated or propagating outwards during an iteration. New fit solutions are accepted based on the flagging system introduced in the previous section. We add a new flag in this phase that increases the total flag value $\mathcal{F}_{\rm tot}$ by a value of 2 if the fit solution is inconsistent with the

[12] This number is reduced accordingly in case neighbouring spectra are masked out or the central spectrum happens to be close to or at the border of the image.

required centroid positions inferred from the spatial consistency check.

Panels b and c of Fig. 11 show example cases for the spatial consistency check of centroid values for the case of a simple emission line feature. Based on the fit solutions in the neighbouring spectra we want to establish whether a one- or two-component fit should be used for the central spectrum. For this example we use the default settings of the algorithm, that is $\mathcal{W}_{\rm min} = 0.5$ and $f_w = 2$, which sets $w_1 = 1/3$ and $w_2 = 1/6$.

For the case depicted in panel b the required weight threshold for the first set of iterations is $\mathcal{W} = 1 - w_1 = 5/6$. The $\mathcal{W}_{\rm tot}$ value for the vertical and horizontal direction would reach this threshold, giving us two conflicting fit solutions for the central spectrum. In such a case, we recompute $\mathcal{W}_{\rm tot}$ for the fit solutions by grouping the eight immediate surrounding neighbouring spectra together and choose the fit solution with the higher $\mathcal{W}_{\rm tot}$ value. For the setup depicted in panel b the fit solution with one Gaussian component would be selected, as the immediate surrounding neighbours with this fit solution have a bigger total weight of $\mathcal{W}_{\rm tot} = 2w_1 + 3w_1/\sqrt{2}$ (compared to $\mathcal{W}_{\rm tot} = 2w_1 + w_1/\sqrt{2}$ for the two-component fit solution)[13]. We would thus try to refit the central spectrum with a fit solution that uses only one Gaussian component. However, the fit solution for the central spectrum is only updated if the total flag value for the fit solution using one component is lower or equal than the total flag value for the fit solution using two components in addition to the requirements that the distribution of the residual data points resembles a normal distribution (see Sect. 3.3.1).

For the example case depicted in panel c of Fig. 11 none of the four main directions would contain fit solutions that pass a weight threshold of $\mathcal{W} = 5/6$. However, both the vertical and the diagonal direction from upper left to lower right would reach a weight threshold of $\mathcal{W} = 4/6$, which is used in the second round of iterations. The total weight for the single component fit solution in the diagonal direction ($\mathcal{W}_{\rm tot} = 2{\cdot}w_1/\sqrt{2}{+}2{\cdot}w_2/\sqrt{2} \approx 0.7$) is bigger than the total weight for the two component fit solution in the vertical direction ($\mathcal{W}_{\rm tot} = w_1 + 2 \cdot w_2 = 2/3$) and thus gets selected. Since the central spectrum already has a single component fit we would not try to refit it.

4. Performance of GaussPy+ on samples of synthetic spectra

In this section, we compare the decomposition results of the improved fitting routine of GaussPy+ (Sect. 3.2.3) with the original GaussPy algorithm. We applied both algorithms on samples of synthetic spectra containing: white noise (A); white noise and signal (B); white noise, signal, and negative noise spikes (C); white noise, weak signal, and negative noise spikes (D). We then determine how well the two algorithms were able to recover the mean position, amplitudes and FWHM values of the Gaussian components used to create the synthetic spectra. For more details about the synthetic spectra, see Appendix B.1.

To facilitate the comparison, we supplied the results from the noise calculation of GaussPy+ (Sect. 3.1.1) also to the decomposition with the original GaussPy algorithm. We also use the same S/N thresholds for the original spectrum ($\rm SNR_1 = 3$) and the second derivative of the smoothed spectrum ($\rm SNR_2 = 3$) for the decompositions with GaussPy and GaussPy+. We use the smoothing parameters α_1 and α_2 we obtained from the training sets decomposed with the method outlined in Sect. 3.1.4[14] (see Appendix B.5 for more details). We left all additional parameters of GaussPy+ at their default settings.

Table 1. Percentage of correctly and incorrectly identified mean positions of Gaussian components for decomposition runs on samples of synthetic spectra.

	GaussPy		GaussPy+	
Sample	Correct [(a)]	Incorrect [(b)]	Correct [(a)]	Incorrect [(b)]
A	–	2.8%	–	0.0%
B	78.0%	3.9%	93.7%	1.6%
C	72.6%	3.7%	93.4%	1.8%
D	29.4%	6.5%	81.7%	4.5%

Notes. [(a)]We define the mean position of a Gaussian component as correctly identified if it is within ±2 spectral channels of the true value. [(b)]We define the fraction of incorrect identifications for sample A as all spectra for which noise features were fitted. The percentage of incorrect identifications for sample B–D refers to the fraction of fitted Gaussian components whose mean position was located at a distance of more than 4 spectral channels to the true value.

Table 1 presents quality metrics of the results of the decomposition runs with GaussPy and GaussPy+ for the four samples of synthetic spectra. The percentage of correct detections refers to the number of Gaussian components that were fitted within ±2 spectral channels of the true position. For a correct identification of a peak position we do not consider whether the amplitude and FWHM values of the Gaussian component were fitted correctly. The fraction of incorrect detections refers either to all spectra for which at least one noise feature was fitted (in case of sample A) or the percentage of Gaussian fit components that were placed at a distance of more than 4 spectral channels away from the true position.

Table 1 demonstrates that GaussPy+ manages to fit significantly more Gaussian components at the correct positions in the spectrum than GaussPy, while decreasing the fraction of incorrect identifications[15]. This improvement is especially striking for weak signal peaks (sample D), where the number of correctly placed Gaussian fit components increased by more than a factor of 2.7 in the GaussPy+ decomposition. The performance of the GaussPy+ decomposition is also not affected by the presence of negative noise spikes in the spectrum (sample C), whereas this has a more significant impact on the performance of GaussPy. Moreover, GaussPy+ did not incorrectly fit any Gaussian components in sample A, whereas GaussPy mistook noise features as signal peaks for 2.8% of the spectra.

Figure 12 compares the fitted Gaussian parameters to the true values used to create the synthetic spectra. The GaussPy+ decomposition results for sample B, C, and D are shown in blue, red, and orange, respectively and the corresponding GaussPy results are indicated with the black line. The left column of

[13] In case both fit solutions have the same total weight as calculated from its immediate surrounding neighbours and this way to decide on the fit solutions thus should be inconclusive, we repeat this total weight calculation for all 16 considered neighbours (coloured squares in panel a of Fig. 11). If this is also inconclusive we choose the fit solution that uses fewer Gaussian components.

[14] For sample A we use the same smoothing parameters as for sample D.

[15] A limiting factor for the performance of GaussPy+ was that the synthetic spectra were not set up to show spatial coherence. Thus, the algorithm will have had difficulty in the decomposition of some spectra to correctly decide whether a structure might be blended and better fit by multiple peaks.

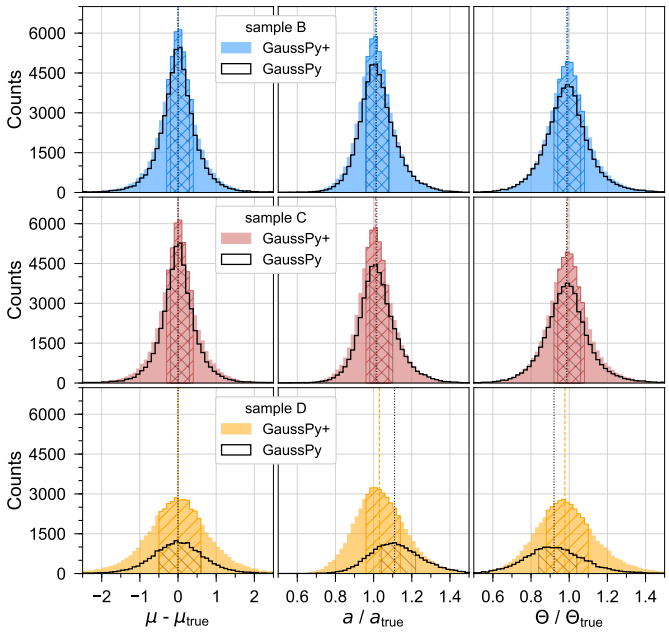

Fig. 12. Comparison of the performance results of decompositions with GaussPy+ and GaussPy for different samples of synthetic spectra. The distribution shows how the fitted parameter values (mean position μ, amplitude a, and FWHM Θ from left to right, respectively) compare to the true parameter values used to create the synthetic spectra. The unfilled and coloured histograms show the distribution of fit components obtained with GaussPy and GaussPy+, respectively. Hatched areas correspond to the interquartile ranges and the vertical lines indicate the median value of the distribution (coloured and black for the GaussPy+ and GaussPy results, respectively. The improved fitting routine of GaussPy+ leads to a significant increase of correctly fitted parameters (see also Table 1 and Sect. 4 for more details).

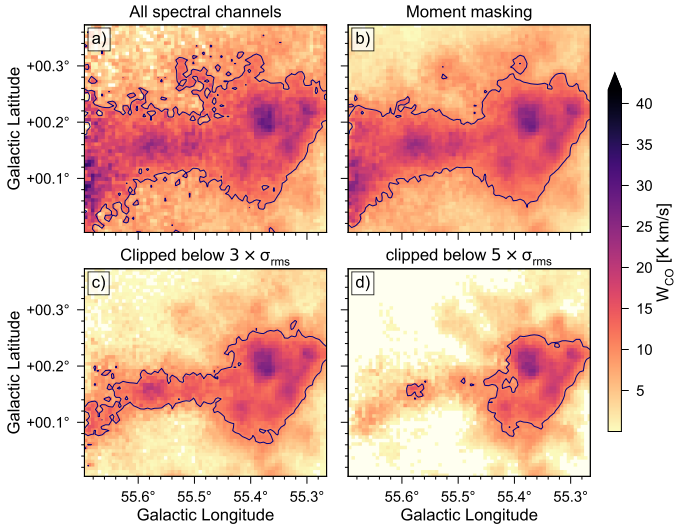

Fig. 13. Zeroth moment maps for a region in the outer parts of the GRS. *Panels a–d*: Results obtained by summing up all spectral channels, applying a moment masking technique (see Sect. 5.1 for details), and clipping all spectral channels with values below $3\times\sigma_{\rm rms}$ and $5\times\sigma_{\rm rms}$, respectively. The contour indicates a $W_{\rm CO}$ level of 5 K km s^{-1}.

panels shows the distribution of fitted mean positions from which the true mean position was subtracted. As already demonstrated in Table 1, the vast majority of components were fitted close to the true mean position. There were fewer detected peaks in sample D because the signal in these spectra was constructed to be close to or below the detection limit.

The middle and right column of panels in Fig. 12 show the distribution of amplitude and FWHM values, respectively, both normalised by the corresponding true parameter values. In these distributions we only included those fitted Gaussian components whose mean position was less than two channels away from the true mean position of the corresponding Gaussian component in the synthetic spectrum (corresponding to the percentages of correctly identified components in Table 1). For all three samples of synthetic spectra the vast majority of fitted parameters are within ±10% of the true values for both decompositions, but due to the higher amount of correctly identified peak positions, GaussPy+ manages to fit many more components correctly. Moreover, for sample D the median values of the distribution are closer to the true values for the GaussPy+ decomposition results. In contrast, GaussPy tends to fit the spectra of sample D with components that have too large amplitude values and too narrow linewidths, as demonstrated from the shape of the distributions and their median values.

We also found that the decomposition performance of GaussPy+ shows much less dependence on the number of signal peaks, their S/N, their linewidth, or their closest distance to a neighbouring signal peak than the decomposition with GaussPy. See Appendix B.6 for a discussion about these comparisons.

5. Performance of GaussPy+ on a GRS test field

In this section, we focus on a sub-region of the GRS data set and perform a detailed analysis and discussion of the decomposition results with GaussPy+ to showcase its performance. The test field we chose is a $0.43^\circ \times 0.37^\circ$ region located towards the outer part of the GRS coverage at Galactic coordinates of $l = 55.48^\circ$ and $b = 0.19^\circ$. This GRS region contains 4200 spectra with 424 spectral channels that cover $v_{\rm LSR}$ values of −5 to 85 km s^{-1}. The chosen region contains three molecular clouds (G055.64+00.14, G055.39+00.14, G055.34+00.19) and 19 clumps as identified by Rathborne et al. (2009).

In the following sections we will first describe the best way to compare flux between the original data set and the decomposition and show the improvements we gain by using the noise estimation technique built into GaussPy+. We then make a detailed comparison between the decomposition results of GaussPy and GaussPy+. Details about the execution time for the entire decomposition and the performance of the spatially coherent refitting can be found in Appendices C.2 and C.5, respectively.

5.1. Optimal flux estimate for fair comparisons between the data set and decomposition results

One measure of the quality of the decomposition results is the fraction of recovered flux from the comparison of zeroth moment maps; we aim at inspecting this fraction in Sect. 5.3. However, imperfect baseline corrections and noise spikes can lead to wrong flux estimates if all spectral channels are integrated along the spectral axis. It is therefore recommended to mask out all spectral channels that do not contain signal.

For our comparisons of the recovered flux in the decomposition (Sect. 5.3) we opted to use the moment masking technique outlined in Dame (2011). The basic idea of moment masking is to mask out spectral channels based on S/N cuts on a spatially and spectrally smoothed version of the original data set. For the smoothed data cube, Dame (2011) suggests to degrade the spatial resolution by a factor of 2 and degrade the spectral resolution

to the width of the narrowest spectral lines contained in the data set. Dame (2011) found that a threshold of $5 \times \sigma_{\mathrm{rms,\,smoothed}}$ gives the best results, where $\sigma_{\mathrm{rms,\,smoothed}}$ refers to the rms-noise of the smoothed spectra. If a spectral channel in the smoothed cube exceeds this S/N threshold, we unmask this channel and all channels that were within the spatial and spectral smoothing kernels in the original datacube. Moment masking thus allows us to also include spectral channels whose value has low S/N levels and would be masked out if we based the clipping of spectral channels on a S/N threshold of the original data set. Moreover, the high S/N requirement for spectral channels of the smoothed data set guarantees that most of the channels containing noise are masked out.

For the moment masking of the GRS test field, we created a smoothed version of the data cube by smoothing the original data set spatially with a Gaussian kernel with a FWHM value of 92″ (corresponding to twice the beam size); spectrally, we smoothed the data set with a Gaussian kernel with a FWHM value of 0.42 km s^{-1}, which corresponds to twice the spectral resolution or 2 spectral channels. We then masked out all spectral channels whose value in the smoothed data cube was below a S/N threshold of 5.

Figure 13 shows zeroth moment maps of our test region obtained by: summing up all spectral channels (panel a); using the moment masking technique described above (panel b); masking out all spectral channels with S/N values below 3 and 5 (panel c and d, respectively). The contour in the panels marks a value of $W_{\mathrm{CO}} = 5$ K km s^{-1}, with W_{CO} being the integrated CO intensity along the spectral axis. By summing up all intensity values along the spectral axis we also include a significant contribution from noise, which is clearly visible in the fraying of the contour line in panel a of Fig. 13. If we mask out all spectral channels with S/N values lower than 3 or 5 times the σ_{rms} (panels c and d), we also cut away a significant fraction of real signal, leading to a severe underestimate of the total flux contained in the region. Conversely, the zeroth moment map constructed with the moment masking technique (panel b in Fig. 13) replicates well the flux distribution of panel a, and excludes most of the noise contributions.

We quantify the recovered flux by summing up all intensity values above a value of 5 K km s^{-1} (contours in Fig. 13). The summed up value inside the contour of the map obtained with the moment masking technique is only 8% smaller than the corresponding value of the map in which we sum up all spectral channels. This small difference is likely due to contributions of spectral channels containing only noise that are also included in the zeroth moment map shown in panel a in Fig. 13. In contrast, the summed up value inside the contours of panels c and d of Fig. 13 is smaller by 33% and 56% respectively than the summed up value for the corresponding contour in panel a.

We conclude that by summing up all spectral channels or masking out spectral channels via a S/N threshold based on the original data set we would either slightly overestimate or severely underestimate the flux contained in the data set, respectively. On the other hand, the moment masking technique gives a good estimate of the total flux contained in a data set and we therefore use it in comparisons of the flux between the decomposition results and the original data set. We assume here implicitly that noise contributions in the remaining spectral channels average out when the intensity values are integrated.

5.2. Noise map

A good estimate of the noise is crucial for obtaining good fitting results if parameters of the decomposition technique are based on S/N thresholds (see Sect. 3.1.1). Figure 14 shows noise maps for the region depicted in Fig. 13 that were obtained with the noise estimation routine of GaussPy+ (panel a) and a much simpler approach that uses a fixed number of channels to calculate the σ_{rms} value (panel b). For the latter approach we used 24 spectral channels from 80−85 km s^{-1} (corresponding to ∼6% of all available spectral channels), similar as it was done in Jackson et al. (2006) for this region. For the GRS data set, the fixed channel approach can be problematic, since there is not really a channel interval that is guaranteed to be emission-free over the entire survey region. It can be clearly seen that the GaussPy+ noise estimation routine gives a much better estimate of the σ_{rms} values, as artefacts from the map-making procedures become more pronounced. There is also a clear gradient in the $\sigma(T_{\mathrm{A}}^{*})$ values in this region, which makes the GaussPy+ decomposition challenging, since it uses the same decomposition parameters throughout the whole region. We would thus expect to have more difficulty in the decomposition of spectra with high $\sigma(T_{\mathrm{A}}^{*})$ values, leading to small S/N values of the signal peaks in these spectra.

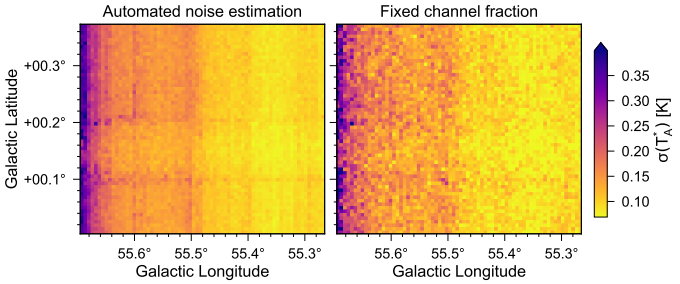

Fig. 14. Noise maps for the region in Fig. 13. *Left*: Results from the automated noise estimation technique discussed in Sect. 3.1.1. *Right*: Results from using a fixed amount of spectral channels for the noise calculation.

Panel a in Fig. 15 displays histograms of the noise maps of Fig. 14; the automated noise estimate shows a clear bimodal distribution, whereas the fixed channel fraction approach is more influenced by random fluctuations of the noise in the limited fixed number of channels used for the noise calculation. The median $\sigma(T_{\mathrm{A}}^{*})$ value of our automated noise estimation is only ∼6% higher than the median value obtained via the fixed channel approach, so globally the two methods give similar results. However, Fig. 14 shows that there are considerable differences on the individual line of sight scale, which will lead to large differences in the decomposition.

To quantify the impact of the estimated noise on the fitting results, we performed two decomposition runs with the improved fitting routine of GaussPy+ (Sect. 3.2.3) with identical settings but different noise estimates corresponding to the maps of Fig. 14. Panel b in Fig. 15 shows the difference between the number of fitted Gaussian components for the noise estimate using a fixed fraction of channels and the automated routine of GaussPy+. About 26% of the spectra in the test field get fitted with a different number of Gaussian components and the total number of fitted components increases by ∼9% for the fixed channel fraction approach. Applying the flagging procedure of GaussPy+ with its default settings (described in Sect. 3.3.1) to the two decompositions, we get that 43.8% and 51.2% of the fitted spectra would be selected for refitting if the automated noise estimate and fixed channel approach are used, respectively. Compared to the GaussPy+ decomposition with the automated noise estimate, in the fixed channel approach the number of spectra flagged as having a number of components

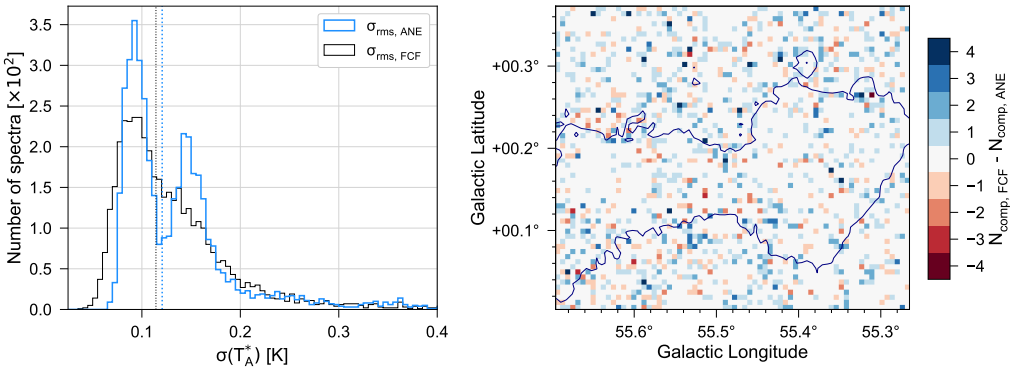

Fig. 15. *Left*: Histogram of the $\sigma_{\rm rms}$ values shown in Fig. 14 for the automated noise estimation of GaussPy+ (ANE, blue) and the fixed channel fraction approach (FCF, black). The dotted vertical lines show the corresponding median values of the two distributions. *Right*: Map showing the difference in the number of fitted components for the automated noise estimation ($N_{\rm comp,\,ANE}$) and the fixed channel fraction approach ($N_{\rm comp,\,FCF}$).

incompatible with their neighbours increase from 1.5% to 6.1% and the number of spectra having normalised residual values not matching a Gaussian distribution increases from 20.0% to 25.1%. Both of these increased numbers of flagged spectra are a good indication that the noise estimate using the fixed fraction of channels is yielding poorer decomposition results than the automated noise estimation routine incorporated in GaussPy+.

5.3. Comparison between the decomposition runs with GaussPy and GaussPy+

In this section we present decomposition runs of the GRS test field obtained with the original GaussPy algorithm and GaussPy+. The different GaussPy+ runs represent results after different stages of the algorithm (improved fitting routine, phase 1 and 2 of the spatially coherent refitting, referred to as Stage 1–3, respectively) to better illustrate the changes and improvements obtained in each individual stage.

We decomposed 250 randomly chosen spectra of the test field with the method outlined in Sect. 3.1.4 to create the training set needed to infer optimal smoothing parameters for GaussPy. Lindner et al. (2015) found that having two different smoothing parameters – one parameter with a smaller value that accentuates the narrower peaks and another parameter with a higher value that is more suitable for broader peaks – leads to huge improvements in the decomposition of HI spectra. We also found that a two-phase decomposition approach using two different smoothing parameters α_1 and α_2 yields better decomposition results for the CO emission line spectra of the GRS data set. For the same training set, the F_1 score (see Sect. 2.1) for the one-phase and two-phase decomposition approaches was 67.5% and 74.7%, respectively. We therefore used the smoothing parameters inferred from the two-phase decomposition of the training set, which yielded values of $\alpha_1 = 2.89$ and $\alpha_2 = 6.65$. For the GaussPy decomposition we set $\mathrm{SNR}_1 = \mathrm{SNR}_2 = 3$. We left all GaussPy+ parameters at their default settings, with the exception of setting $\Delta\mu_{\rm max} = 4$ for Stage 3.

Panels a–d in Fig. 16 show zeroth moment maps of the decomposition runs with the original GaussPy algorithm supplied with the improved noise estimation (panel a of Fig. 14), and GaussPy+ after the improved fitting stage (panel b; Sect. 3.2.3), and after phase 1 (panel c; Sect. 3.3.1) and phase 2 (panel d; Sect. 3.3.2) of the spatially coherent refitting. The zeroth moment maps were obtained by masking the same spectral channels as for the moment masked data in panel b of Fig. 13.

Panels e–h in Fig. 16 show the corresponding zeroth moment maps of the residual. In all three stages of GaussPy+ the performance in terms of the recovered flux is much better for the regions with lower S/N emission than the GaussPy decomposition, which was already noticeable in the case of synthetic spectra (Sect. 4, Appendices B.6 and B.7). For regions with high S/N, GaussPy and all stages of GaussPy+ perform very well[16].

The maps in panels i–l of Fig. 13 show the $\chi^2_{\rm red}$ values for the fits, with the goodness of fit calculation restricted to the channels estimated to contain signal (see Sect. 3.1.2). We can see a clear improvement towards $\chi^2_{\rm red}$ values closer to 1 for the GaussPy+ decompositions compared to the GaussPy run. In Appendix E we demonstrate the importance of restricting the calculation of the $\chi^2_{\rm red}$ values to regions in the spectrum that contain signal for the GRS test field.

The performance in recovered flux does not significantly change in the spatially coherent refitting phases of GaussPy+, since the focus in these phases is shifted to reducing flagged features and making the fit results compatible with the neighbours instead of minimising the residual. Therefore the zeroth moment, residual and $\chi^2_{\rm red}$ maps that show the quality of the fit results in terms of recovered flux do not change significantly between Stage 1 and Stage 3 of the GaussPy+ decomposition.

We can see more variation between panels m–p in Fig. 16, which show maps of the number of fitted Gaussian components per spectrum for each decomposition run. In the GaussPy+ decompositions the number of fitted components increases compared to the GaussPy run, which is due to the fitting of weaker emission lines in spectra containing increased noise levels (cf. panels e–h) and the segmentation of very broad components into individual peaks. There is also a clear progression towards more spatial coherence from panels m–p.

Figure 17 further demonstrates this transition towards spatial coherence by comparing the fitting results of GaussPy and Stage 1–3 of GaussPy+ for nine neighbouring spectra from the GRS test field. The signal peaks in these spectra show only moderate S/N and GaussPy therefore tends to fit broad Gaussian components over most of the signal peaks. Stage 1 of GaussPy+ already manages to improve upon these fitting results by separating the emission into more individual peaks; this improvement of

[16] We caution that while the recovered flux is an essential criterion for the performance of the fit it may not give a good handle on the quality of the fits themselves. For example, the spectra might not be spatially coherent and might use many blended and broad components to fit the spectrum.

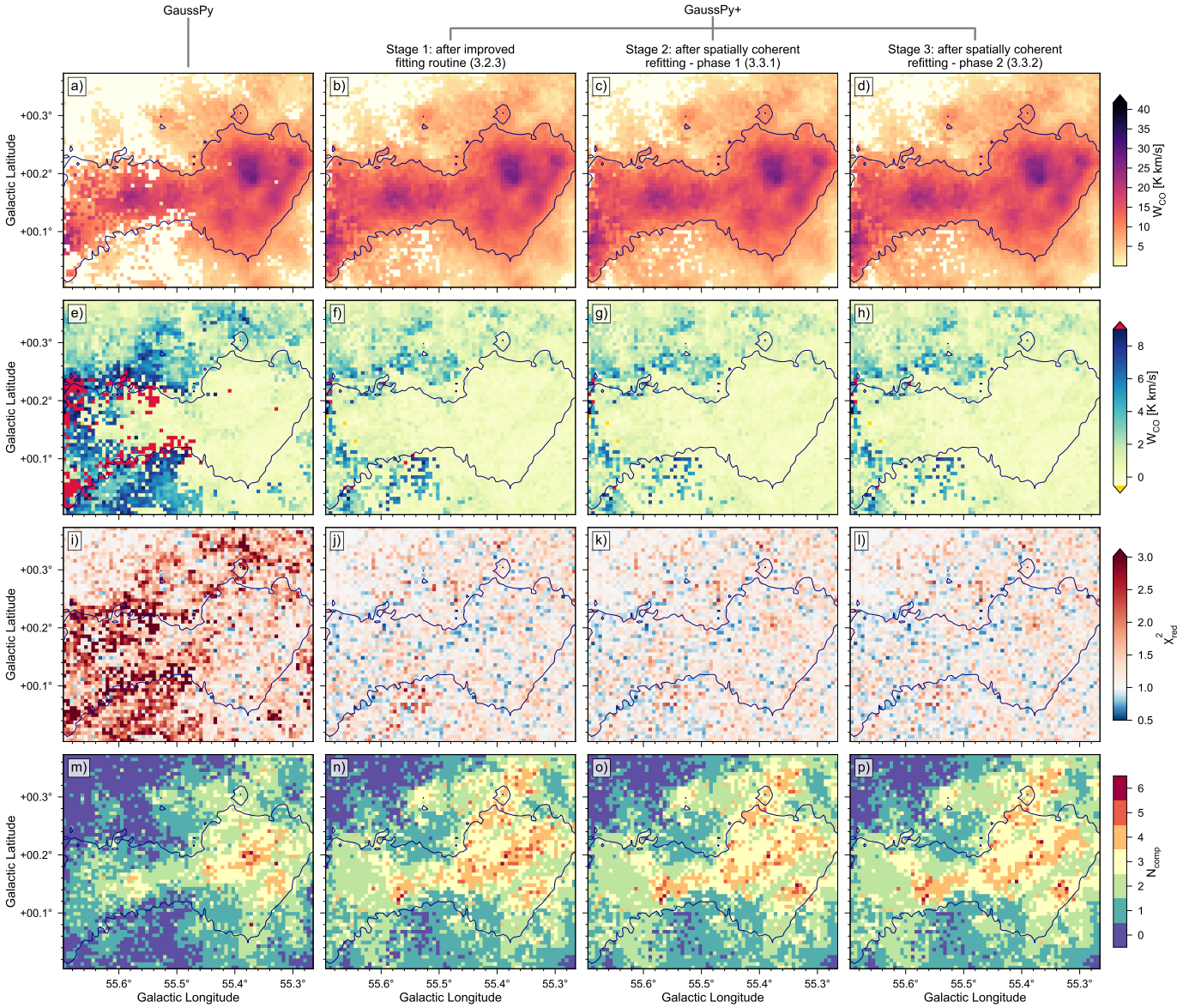

Fig. 16. *From left to right*: Decomposition results for the original GaussPy algorithm and three stages of GaussPy+ (improved fitting routine, phase 1 and 2 of the spatially coherent refitting). *From top to bottom*: Zeroth moment maps of the decomposition results; residual maps obtained by comparing the zeroth moment maps of the decomposition with *panel b* in Fig. 13; maps showing the $\chi^2_{\rm red}$ values for the fit, with the goodness of fit calculation restricted to the channels estimated to contain signal (see Sect. 3.1.2); and maps of the number of Gaussian fit components per spectrum. All panels are overplotted with the contour from *panel b* in Fig. 13. The GaussPy+ decompositions show a clear trend towards more spatial coherence and an improvement in the quality of the fits for the regions with lower emission or higher noise levels.

the decomposition results can also be seen in the decreased residuals shown in the smaller panels. Stage 2 of GaussPy+, which uses the information of already well-fit neighbouring spectra as input guesses for flagged spectra, can even further improve upon these results by creating more spatial coherence between the spectra. Finally, Stage 3 of GaussPy+, which tries to enforce spatial coherence between the centroid values of the fit components, improves the decomposition results once more, by getting rid of a fit component for the central spectrum that was inconsistent with the neighbouring fit solutions.

In Table 2, we compare parameters and the percentage of flagged spectra for the decomposition results for GaussPy and the three stages of GaussPy+ depicted in Fig. 16. The $W_{\rm CO,\,all}$ and $W_{\rm CO,\,contour}$ parameters give the fraction of recovered intensity values integrated along the spectral axis for the whole test field and inside the contour of 5 K km s^{-1}, respectively. The $W_{\rm CO,\,all}$ and $W_{\rm CO,\,contour}$ values were determined by comparing the moment maps of the decompositions (panels a–d in Fig. 16) to the moment masked zeroth moment map of panel b in Fig. 13. As already noticeable in Fig. 16, the performance of GaussPy and GaussPy+ is better for spectra containing high S/N emission peaks than for weaker emission lines. With GaussPy+ we are able to recover about 90% of the $W_{\rm CO}$ values contained in the entire test field and ~95% of the $W_{\rm CO}$ values contained inside the contour of 5 K km s^{-1}. Compared to the GaussPy+ runs, the decomposition with the original GaussPy algorithm recovers about 12% less flux inside the contour and 16% less flux in the entire field.

The total number of fitted Gaussian components $N_{\rm comp}$ increases by about half for the GaussPy+ decompositions compared to the GaussPy run. The median $\chi^2_{\rm red}$ values ($\chi^2_{\rm red,\,med}$) of the GaussPy+ fitting results are also lower by ~22% than for the GaussPy results.

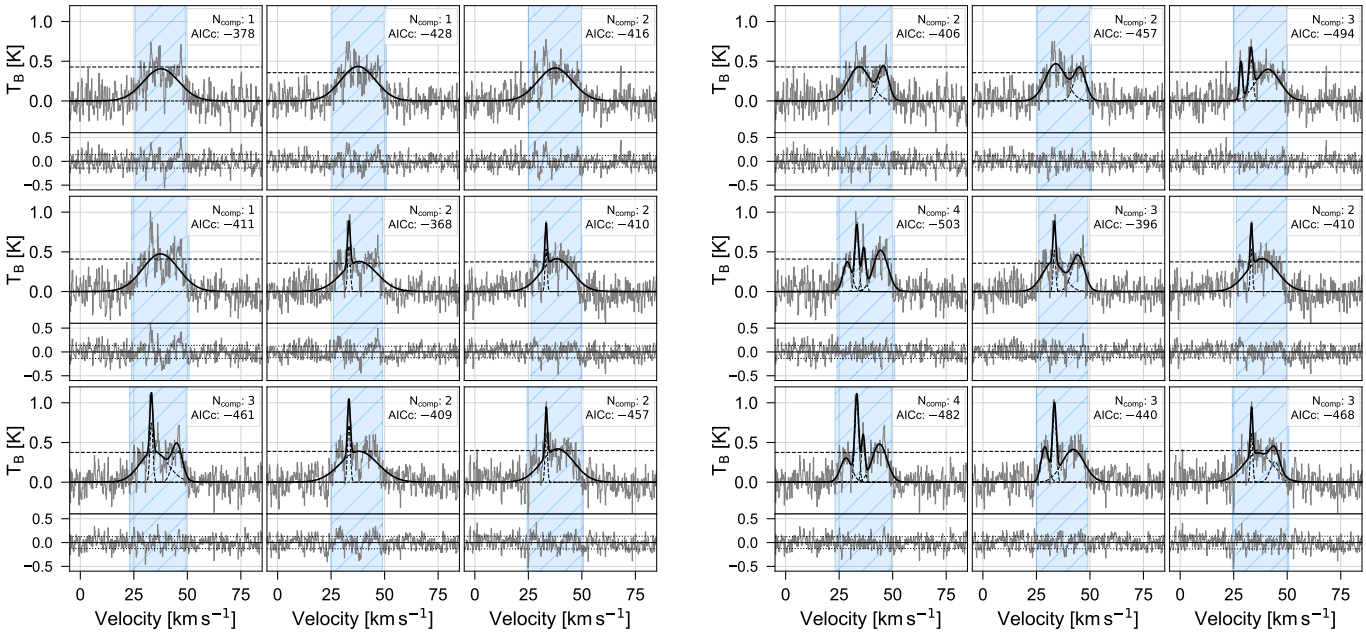

Fig. 17. Fitting results of nine neighbouring spectra in the GRS test field for the decomposition with GaussPy (*a*) and after Stage 1–3 of GaussPy+ (*b–d*, respectively). Individual fit components and their combination are shown in dashed and solid black lines, respectively. Horizontal dashed black lines mark a S/N threshold of 3 and blue shaded areas indicate the identified signal intervals. The number of used fit components $N_{\rm comp}$ and the resulting AICc values are noted in the upper right corner of the main panel. The smaller subpanels show the residual with the horizontal dotted black lines marking values of $\pm\sigma_{\rm rms}$.

Table 2 also shows the fraction of spectra of the GaussPy and GaussPy+ results that would be flagged as not satisfying the quality criteria used in the first phase of the spatially coherent refitting (Sect. 3.3.2). We use the default flagging criteria of GaussPy+, which means that spectra get flagged if they have blended components ($\mathcal{F}_{\rm blended}$), show negative residual features ($\mathcal{F}_{\rm neg.\,res.\,peak}$), have broad components ($\mathcal{F}_{\Theta}$, determined with $f_{\Theta,\,\rm max} = 2$), have residual data values whose distribution does not correspond to what is expected from Gaussian noise ($\mathcal{F}_{\rm residual}$), or were fitted with a number of components that is not consistent with the number of components used in the fit solutions of neighbouring spectra ($\mathcal{F}_{N_{\rm comp}}$). The fraction of spectra that contain broad components in relation to neighbouring components is indicated with $\mathcal{F}_{\Theta}$. To better judge how many components with very large absolute FWHM values occur in the decompositions, we also list the fraction of spectra that contain components with FWHM values above 50 spectral channels ($\mathcal{F}_{\Theta>50}$) that would imply very high velocity dispersion values

Table 2. Comparison of parameters and flagged spectra for the decomposition runs with GaussPy and GaussPy+.

	GaussPy	GaussPy+ (Stage 1)	GaussPy+ (Stage 2)	GaussPy+ (Stage 3)
$W_{\mathrm{CO,\,all}}$	73.0%	88.9%	89.6%	89.8%
$W_{\mathrm{CO,\,contour}}$	84.0%	95.5%	95.6%	95.7%
$\chi^2_{\mathrm{red,\,med}}$	1.436	1.123	1.121	1.113
$\mathcal{F}_{\mathrm{tot}}$	59.2%	43.8%	35.5%	38.0%
$\mathcal{F}_{\mathrm{blended}}$	5.8%	7.5%	2.9%	3.2%
$\mathcal{F}_{\mathrm{neg.\,res.\,peak}}$	2.6%	0.1%	0.0%	0.1%
$\mathcal{F}_{\Theta}$	24.0%	22.7%	19.1%	21.6%
$\mathcal{F}_{\Theta>50}$	10.6%	11.2%	9.6%	9.4%
$\mathcal{F}_{\mathrm{residual}}$	37.1%	20.0%	16.3%	16.5%
$\mathcal{F}_{N_{\mathrm{comp}}}$	0.3%	1.5%	1.1%	1.0%

of $\gtrsim$4.3 km s^{-1}. The total flag value $\mathcal{F}_{\mathrm{tot}}$ gives the percentage of spectra that were flagged by at least one of the individual flags. For the GaussPy decomposition about 59% of the spectra were flagged as not satisfying at least one of the flagging criteria, whereas this reduces to ~35 and 38% for Stage 2 and 3 of GaussPy+, respectively. The fit results from Stage 2 of GaussPy+ show the lowest fraction of flagged spectra, which is expected given that this stage is designed for decreasing the number of flagged spectra. Stage 3 of GaussPy+ aims to increase the spatial coherence of the fit components, which is why the percentage of flagged spectra increases slightly again compared to the Stage 2 fitting results. All three stages of GaussPy+ perform well in removing negative residual features and reducing fit results that lead to a residual whose distribution is inconsistent with Gaussian noise. The percentage of spectra flagged with $\mathcal{F}_{\mathrm{blended}}$, $\mathcal{F}_{\Theta>50}$ and $\mathcal{F}_{N_{\mathrm{comp}}}$ flags actually increases for Stage 1 of GaussPy+ compared to the results of GaussPy, which is likely just an effect of the increased number of fit components used in the GaussPy+ results. These flags are however reduced again in the spatially coherent refitting stages.

Finally, Fig. 18 shows distributions of fit parameters for the decomposition results of GaussPy and the three stages of GaussPy+. The left panel shows histograms of the number of fitted components per spectrum. As was already demonstrated by panels m–p in Fig. 16, GaussPy+ manages to fit more spectra than GaussPy, so that the total number of fitted components increases by about one third for the GaussPy+ stages.

The middle panel of Fig. 18 shows histograms of the amplitude values of all fit components. Comparing these distributions with the histogram of the estimated noise values shown in the left panel of Fig. 15 reveals that GaussPy+ manages to fit many more components whose S/N value is only ~3 or lower. The median S/N value of fit components decreases from 5.4 for the GaussPy decomposition to 4.3 for the GaussPy+ fit results.

The histograms of the velocity dispersion values for all fit components are given in the right panel of Fig. 15. The long tail towards increased $\sigma_{\mathrm{v_{los}}}$ values is mostly due to fitted components with low S/N values; about half of the fit components with $\sigma_{\mathrm{v_{los}}} >$ 4.3 km s^{-1} in the GaussPy+ decomposition results of Stages 2 and 3 have S/N values <2.

6. Discussion

In this section, we list potential applications as well as limitations of GaussPy+. We also give advice on parameter settings to obtain optimal decomposition results.

6.1. Applications and limitations of GaussPy+

The GaussPy+ algorithm should be applicable to any data set that can be well described with Gaussian components; in particular it was designed to decompose large surveys of HI and CO isotopologues. In case the line shape is better matched by a Voigt or Lorentzian profile (e.g. due to effects of pressure broadening) the decomposition with GaussPy+ will likely not give satisfactory results. The algorithm can also not fit the hyperfine structure of molecules such as NH_3 or N_2H^+ directly.

Many of the individual routines implemented in GaussPy+, such as the noise estimation (Sect. 3.1.1), signal identification (Sect. 3.1.2), and masking of noise artefacts (Sect. 3.1.3) can be used as stand-alone applications. For example, the noise estimation can be used in combination with the signal identification to detect baseline shifts, unsubtracted continuum emission, or instrumental artefacts such as increased or amplified noise fluctuations. Phase 1 of the spatially coherent refitting routine (Sect. 3.3.1) can also be used to just flag decomposition results without refitting them.

In its current version, GaussPy+ is not designed to deal with spectra that contain both emission and absorption lines. If users would like to use GaussPy+ for the decomposition of emission lines that are expected to show strong self-absorption (such as the lowest rotational transitions of the ^{12}CO molecule), we recommend to deselect the flagging of negative residual spikes, as in this case one would not want to fit a signal peak that has a dip in its centre with two components.

The GaussPy+ algorithm will only perform well on spectra whose baseline is centred on a value of zero. Incomplete continuum subtraction or baseline shifts of the spectrum will lead to wrong noise estimates, which in turn will give insufficient decomposition results, since core functionalities of GaussPy+ depend on the correctness of the estimated noise values.

The GaussPy+ algorithm can deal with large variations of the noise (see Sect. 5). However, since key steps of the algorithm are based on S/N thresholds, an inhomogeneous noise coverage or variation in the quality of the data will have an impact on the decomposition results.

In its current implementation GaussPy+ does not explicitly check for spatial coherence of the amplitude and FWHM values. In principle, these values should also become more coherent in the two phases of the spatially coherent refitting (Sect. 3.3), where neighbouring fit solutions are used to improve the fit of a spectrum. We focus on spatial coherence of the centroid positions, since it is a necessary requirement for correct amplitude and FWHM values. If Gaussian fit components are not placed correctly, their amplitude and FWHM values will by default be spatially inconsistent with neighouring fit solutions. We also have to caution against constraining the FWHM parameters of Gaussian components with too restrictive limits based on fit solutions from neighbours. In tests we performed, such a constraint could lead to Gaussian fit components with FWHM values close to the lower or upper limit of the constraint. This effect caused artefacts in the distribution of all fitted FWHM values, but in case of smaller data sets this might not be easily noticeable. We thus do not enforce limits for the width of the Gaussian fit components in any of the stages of GaussPy+, apart from the requirement that the FWHM value has to be larger than the user defined Θ_{min} parameter, whose value defaults to the channel width of the data set. This fitting without an upper limit and without a more constrained lower limit could allow fluctuations in the FWHM values between the Gaussian components of neighbouring spectra.

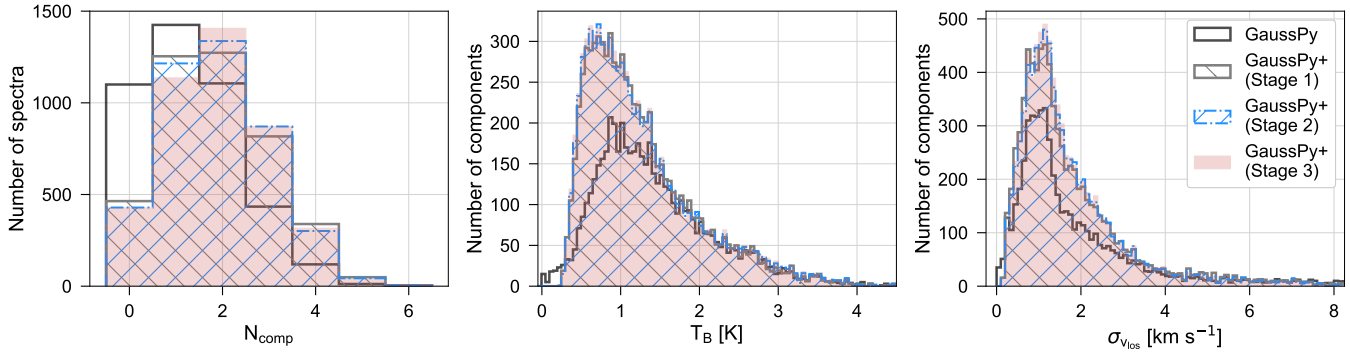

Fig. 18. Distribution of fit parameters for the decomposition results of the GRS test field with GaussPy and the three stages of GaussPy+. *Left*: Histogram of the number of fitted components per spectrum. *Middle*: Histogram of the amplitude values T_B of all Gaussian fit components. The bin size is 0.05 K. *Right*: Histogram of the velocity dispersion values $\sigma_{v_{los}}$ of all Gaussian fit components. The bin size is 0.1 km s^{-1}.

Our approach in phase 2 of the spatially coherent refitting will also favor structures with ellipsoid morphologies over possible ring-like structures (see Fig. 11). Users thus should be cautious in using the spatially coherent refitting for centroid positions if the structures probed by the observations are not expected to be continuous over multiple neighbouring pixels or the data is not Nyquist sampled.

6.2. Recommended settings for GaussPy+

We tested the default settings of GaussPy+ on synthetic spectra and line emission data from a ^{13}CO survey and obtained very good decomposition results with them. However, different data sets may require significantly different settings. For example, in HI observations we would expect two distinct populations of narrow and very broad lineshapes corresponding to contributions from the cold and warm neutral medium respectively (e.g. Heiles & Troland 2003), which is not the case for observations of CO isotopologues. For the HI observations one would thus not flag and refit broad Gaussian components, whereas this setting can lead to better decomposition results for the CO data sets. Ultimately, it is the responsibility of the user to consider if the decomposition results of GaussPy+ are scientifically meaningful for the chosen application.

In our application of GaussPy+ on the GRS data set we found it beneficial for the fitting to also retain weak components with amplitudes below a S/N threshold of 3. Since the decomposition of GaussPy+ performs a least squares minimisation of the residual, the fit of higher peaks in a spectrum can be negatively affected if weak components get discarded or neglected. We thus recommend to also accept components with $S/N_{\mathrm{min,\,fit}} < 3$ in the decomposition and only later on perform a cut based on their S/N values.

The GaussPy+ algorithm is designed to deal with spectra that contain only weak emission lines with S/N values around 3 or even lower. The quality check for the significance of a Gaussian component is specifically designed to help in such cases where GaussPy+ operates close to the noise. If the chosen settings for GaussPy+ produce too many false positives, users are advised to increase the chosen S/N limit and/or increase the value of the $\mathcal{S}_{\mathrm{min}}$ threshold. Conversely, in case the decomposition results of GaussPy+ are not including a significant fraction of signal peaks, users should try to decrease one or both of these parameter settings (see Appendix C.3 for how changing both of these parameters affects the decomposition).

We designed GaussPy+ to be customisable to different data sets, which means that most of its parameters can be changed and finetuned by the user (see Table F.2). However, the majority of parameters should yield good results for most data sets if left to their default settings. To get first decomposition results only a small number of parameters (listed as *essential parameters*) have to be specified by the user. In case the decomposition does not yield good results we recommend to first change the *essential parameters* before changing the parameters listed under *more advanced settings* in Table F.2.

7. Summary

In this work, we present the GaussPy+ algorithm, a new fully automated Gaussian fitting package for the decomposition of emission line spectra. The GaussPy+ algorithm is built upon GaussPy (Lindner et al. 2015), but significantly extends and improves upon its performance by the following added, fully automated functionality:

1. Preparatory steps that can also be used as stand-alone applications. This includes methods to accurately estimate the noise (Sect. 3.1.1), identify signal peaks (Sect. 3.1.2), and mask out noise artefacts (Sect. 3.1.3). An additional routine (Sect. 3.1.4) creates suitable training sets for the in-built machine learning process GaussPy uses to infer optimal parameter settings for the decomposition of a data set.
2. Quality controls that are highly customisable to different data sets (Sect. 3.2.1). This includes a criterion that takes into account both the S/N values and the number of spectral channels of a signal feature or fitted Gaussian component and goodness of fit criteria to aid in the selection of the best fit solution for a spectrum. Additional optional quality controls (Sect. 3.2.2) allow the user to flag and refit unwanted features in the decomposition such as blended Gaussian components, negative peaks in the residual, very broad Gaussian components, residual data points that are not normally distributed, or differences in the number of fitted components between neighbouring spectra.
3. An improved fitting routine (Sect. 3.2.3) that is guided by the user-defined optional quality controls.
4. A spatially coherent refitting routine (Sect. 3.3) that tries to refit spectra that do not pass the user-defined quality controls or spectra whose decompositions shows spatial incoherence with neighbouring fit solutions.

We thoroughly tested the performance of GaussPy+ on synthetic spectra designed to cover a wide range of spectral features expected in observations of emission lines of CO isotopologues. We found that it yields very good decomposition results

that significantly outperform the original GaussPy algorithm in all tested cases (Sect. 4). We also applied GaussPy+ to a test field from the GRS (Sect. 5) and showed that it can fit the data well resulting in considerable improvements in the decomposition compared to the original GaussPy algorithm.

We conclude that the GaussPy+ algorithm is a powerful tool to analyse large Galactic plane surveys, such as GRS or SEDIGISM (Schuller et al. 2017). We will present and discuss its application on the entire GRS data set in a forthcoming paper.

Acknowledgements. We would like to thank the anonymous referee for a very constructive, detailed and clear report that helped to significantly improve this work. This project received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 639459 (PROMISE). J.H.O acknowledges funding from the Swedish Research Council (registration number 2017-03864). C.E.M acknowledges support from a National Science Foundation Astronomy and Astrophysics Postdoctoral Fellowship under Award No. AST-1801471. This publication makes use of molecular line data from the Boston University-FCRAO Galactic Ring Survey (GRS). The GRS is a joint project of Boston University and Five College Radio Astronomy Observatory, funded by the National Science Foundation under grants AST-9800334, AST-0098562, & AST-0100793. Code bibliography: This research made use of matplotlib (Hunter 2007), a suite of open-source python modules that provides a framework for creating scientific plots; astropy, a community-developed core Python package for Astronomy (Astropy Collaboration 2013); aplpy, an open-source plotting package for Python (Robitaille & Bressert 2012); NumPy (Van der Walt et al. 2011); and SciPy (Jones et al. 2001).

References

Akaike, H. 1973, in Proceedings of the 2nd International Symposium on Information Theory, eds. B. N. Petrov, & F. Csaki, 267
Andrae, R., Schulze-Hartung, T., & Melchior, P. 2010, ArXiv e-prints [arXiv:1012.3754]
Arzoumanian, D., André, P., Peretto, N., & Könyves, V. 2013, A&A, 553, A119
Arzoumanian, D., Shimajiri, Y., Inutsuka, S.-I., Inoue, T., & Tachihara, K. 2018, PASJ, 70, 96
Astropy Collaboration (Robitaille, T. P., et al.) 2013, A&A, 558, A33
Banks, H. T., & Joyner, M. L. 2017, Appl. Math. Lett., 74, 33
Barnes, P. J., Muller, E., Indermuehle, B., et al. 2015, ApJ, 812, 6
Beuther, H., Bihr, S., Rugel, M., et al. 2016, A&A, 595, A32
Burkhart, B., Stanimirović, S., Lazarian, A., & Kowal, G. 2010, ApJ, 708, 1204
Burnham, K. P., & Anderson, D. R. 1998, Model Selection and Inference: A Practical Information-Theoretic Approach (Springer-Verlag), 80
Chen, H. H.-H., Pineda, J. E., Goodman, A. A., et al. 2019, ApJ, 877, 93
Clarke, S. D., Whitworth, A. P., Spowage, R. L., et al. 2018, MNRAS, 479, 1722
Colombo, D., Rosolowsky, E., Duarte-Cabral, A., et al. 2019, MNRAS, 483, 4291
Dame, T. M. 2011, ArXiv e-prints [arXiv:1101.1499]
Dame, T. M., Hartmann, D., & Thaddeus, P. 2001, ApJ, 547, 792
Dempsey, J. T., Thomas, H. S., & Currie, M. J. 2013, ApJS, 209, 8
Dénes, H., McClure-Griffiths, N. M., Dickey, J. M., Dawson, J. R., & Murray, C. E. 2018, MNRAS, 479, 1465
D'Agostino, R. B. 1971, Biometrika, 58, 341
D'Agostino, R. B., & Pearson, E. S. 1973, Biometrika, 60, 613
Elmegreen, B. G., & Scalo, J. 2004, ARA&A, 42, 211
Ewen, H. I., & Purcell, E. M. 1951, Nature, 168, 356
Falgarone, E., Pety, J., & Hily-Blant, P. 2009, A&A, 507, 355
Ginsburg, A., & Mirocha, J. 2011, Astrophysics Source Code Library [record ascl:1109.001]
Hacar, A., Tafalla, M., Kauffmann, J., & Kovács, A. 2013, A&A, 554, A55
Hacar, A., Alves, J., Burkert, A., & Goldsmith, P. 2016, A&A, 591, A104
Haud, U. 2000, A&A, 364, 83
Heiles, C., & Troland, T. H. 2003, ApJ, 586, 1067
Hennebelle, P., & Falgarone, E. 2012, A&ARv, 20, 55
Henshaw, J. D., Caselli, P., Fontani, F., Jiménez-Serra, I., & Tan, J. C. 2014, MNRAS, 440, 2860
Henshaw, J. D., Longmore, S. N., Kruijssen, J. M. D., et al. 2016, MNRAS, 457, 2675
Henshaw, J. D., Ginsburg, A., Haworth, T. J., et al. 2019, MNRAS, 485, 2457
Heyer, M. H., & Brunt, C. M. 2004, ApJ, 615, L45
Hily-Blant, P., & Falgarone, E. 2009, A&A, 500, L29
Hunter, J. D. 2007, Comput. Sci. Eng., 9, 90
Hurvich, C. M., & Tsai, C. L. 1989, Biometrika, 76, 297
Jackson, J. M., Rathborne, J. M., Shah, R. Y., et al. 2006, ApJS, 163, 145
Jones, E., Oliphant, T., Peterson, P., et al. 2001, SciPy: Open Source Scientific Tools for Python
Keto, E., Caselli, P., & Rawlings, J. 2015, MNRAS, 446, 3731
Kolmogorov, A. N. 1933, Giornale dell'Instituto Italiano degli Attuari, 4, 83
Larson, R. B. 1981, MNRAS, 194, 809
Lindner, R. R., Vera-Ciro, C., Murray, C. E., et al. 2015, AJ, 149, 138
Marchal, A., Miville-Deschenes, M.-A., Orieux, F., et al. 2019, A&A, 626, A101
Miville-Deschênes, M.-A., Murray, N., & Lee, E. J. 2017, ApJ, 834, 57
Murray, C. E., Stanimirović, S., Goss, W. M., et al. 2015, ApJ, 804, 89
Murray, C. E., Stanimirović, S., Goss, W. M., et al. 2018, ApJS, 238, 14
Nakanishi, H., & Sofue, Y. 2006, PASJ, 58, 847
Orkisz, J. H., Pety, J., Gerin, M., et al. 2017, A&A, 599, A99
Orkisz, J. H., Peretto, N., Pety, J., et al. 2019, A&A, 624, A113
Ossenkopf, V., & Mac Low, M.-M. 2002, A&A, 390, 307
Pineda, J. L., Goldsmith, P. F., Chapman, N., et al. 2010, ApJ, 721, 686
Rathborne, J. M., Johnson, A. M., Jackson, J. M., Shah, R. Y., & Simon, R. 2009, ApJS, 182, 131
Rigby, A. J., Moore, T. J. T., Plume, R., et al. 2016, MNRAS, 456, 2885
Robitaille, T., & Bressert, E. 2012, Astrophysics Source Code Library [record ascl:1208.017]
Roman-Duval, J., Jackson, J. M., Heyer, M., Rathborne, J., & Simon, R. 2010, ApJ, 723, 492
Schuller, F., Csengeri, T., Urquhart, J. S., et al. 2017, A&A, 601, A124
Smirnov, N. V. 1939, Bull. Moscow Univ., 2, 3 (Russian)
Stil, J. M., Taylor, A. R., Dickey, J. M., et al. 2006, AJ, 132, 1158
Su, Y., Yang, J., Zhang, S., et al. 2019, ApJS, 240, 9
Umemoto, T., Minamidani, T., Kuno, N., et al. 2017, PASJ, 69, 78
Van der Walt, S., Colbert, C., & Varoquaux, G. 2011, The NumPy Array: A Structure for Efficient Numerical Computation (IEEE), 13, 22
Wilson, R. W., Jefferts, K. B., & Penzias, A. A. 1970, ApJ, 161, L43

Appendix A: Markov chain

The basic principle or question behind this step is: given a certain peak in the spectrum, what is the probability that this peak was caused by random fluctuations of the noise? This probability depends on the size of the spectrum, as the probability that random noise fluctuations cause a feature resembling a signal peak will increase with the number of spectral channels. The following probabilistic estimation does *not* attempt to quantify the probability of a signal peak being a real feature, but tries to establish the probability of a peak being the result of random noise fluctuations.

To obtain this estimate, we first convert the spectrum into a binary sequence by setting negative channels to a value of 0 and positive channels to a value of 1 (we treat channels that have an exact value of zero as positive channels). Assuming that each channel can be treated independently from each other and is not correlated with its neighbouring channels, this binary sequence is analogous to a sequence of coin tosses, with the number of coin tosses equivalent to the number of spectral channels.

This transformation thus allows us to work out the probability of a sequence of negative or positive channels being due to random noise fluctuations. In case of pure white noise, the probability of a spectral channel having a positive or negative value is 1/2. To calculate the probability of a sequence of n negative or positive spectral channels we use a one-step Markov chain with state space of $\{1, 2, \ldots, n\}$. The $n \times n$ transition matrix $P_{i,j}$ that we use to determine the probability of a sequence of n negative or positive consecutive channels has the following structure:

$$P_{i,j} = \begin{pmatrix} p_{i=1,j=1} & p_{i=1,j=2} & \cdots & p_{i=1,j=n} \\ p_{i=2,j=1} & p_{i=2,j=2} & \cdots & p_{i=2,j=n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i=n,j=1} & p_{i=n,j=2} & \cdots & p_{i=n,j=n} \end{pmatrix}. \tag{A.1}$$

The rows i give the possible states the system can be in (pre-transition states) and the column entries give the probability of transitioning to respective new states. That means that all of the elements in a row have to sum up to a probability of 1 ($\sum_{j=1}^{n} p_{i,j} = 1$).

The individual entries $p_{i,j}$ of the transition matrix have the following values:

$$p_{i,j} = \begin{cases} 1/2, & \text{for } i = 1, 2, \ldots, n-1 \text{ and } j = i+1 \\ 1/2, & \text{for } i = 1, 2, \ldots, n-1 \text{ and } j = 1 \\ 1, & \text{for } i = n \text{ and } j = n \\ 0, & \text{otherwise.} \end{cases} \tag{A.2}$$

This allows us to determine the probability of finding a sequence of n consecutive negative or positive channels in a spectrum with N channels. We start in state 1 (the first spectral channel has either a positive or a negative value) and need to determine the probability of being in state n (corresponding to a sequence of n spectral channels with either positive or negative values) after $N - 1$ Markov chain steps. This probability is given by the $p_{1,n}$ entry of the one-step transition matrix of the form $n \times n$ raised to the power of $N - 1$. We can thus compute the probability for any sequence of n consecutive positive or negative channels in a spectrum with N_{chan} spectral channels with random values.

Let us illustrate this with the example of a Markov chain for 4 consecutive negative or positive channels. In this case the Markov chain has a state space of $\{1, 2, 3, 4\}$ and the transition matrix has the following form:

$$P_{i,j} = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.3}$$

In state 1 (which corresponds to row $i = 1$ of the transition matrix) we have a sequence of one positive or negative channel and we will always start with this state or revert to this state if the sign between neighbouring channels changes before we reached the full sequence of four consecutive channels. In state 2 (row $i = 2$) and state 3 (row $i = 3$) we have a sequence of two and three positive or negative channels, respectively. State 4 (row $i = 4$) is the absorbing final state, where we reached four consecutive positive or negative channels. The individual column entries of each row then give the probabilities of moving to a new state. In our example, the transition matrix element $p_{1,2}$ gives the probability of moving from state 1 to state 2 ($p_{1,2} = 1/2$), and the element $p_{4,3}$ gives the probability of moving from state 4 to state 3 ($p_{4,3} = 0$).

In our example we always start out with a spectral channel that has either a positive or negative value, so state 1 is just a sequence of 1 positive or negative channel. For state 1, there is a probability of 1/2 that the system stays in state 1 (if the value of the next channel changes sign) or that it moves to state 2 (row $i = 2$), in which we have two consecutive channels with the same sign. For state 2 and state 3, there is again a probability of 1/2 that the channel value changes sign and the system moves back to state 1, and a probability of 1/2 that it moves to state 3 or the absorbing state 4, respectively.

Figure A.1 shows Markov chain results for 4 consecutive positive or negative channels in a sequence of 4 or 10 channels with random values (left and right panel, respectively). These matrices were obtained by raising the one-step transition matrix given in Fig. A.3 to the power of 3 and 9, respectively. The last element in the first row of the matrices (highlighted in blue) gives the respective probabilities to get 4 consecutive positive or negative channels in random sequences of 4 or 10 channels.

Given the random fluctuations of the noise, it becomes clear that the more spectral channels there are, the higher the probability of getting a sequence of n channels with positive or negative value. For example, the probability of having a sequence of ten consecutive positive or negative channels in a spectrum of 100 channels is 0.088. If the number of spectral channels doubles to 200, the probability of getting a sequence of ten consecutive positive or negative channels increases to 0.173.

0.5	0.25	0.125	0.125
0.375	0.25	0.125	0.25
0.25	0.125	0.125	0.5
0	0	0	1

N = 4

0.291	0.158	0.086	0.465
0.244	0.133	0.072	0.551
0.158	0.086	0.047	0.709
0	0	0	1

N = 10

Fig. A.1. One-step Markov chain results for 4 consecutive negative or positive spectral channels in a sequence of 4 (*left*) or 10 (*right*) channels with random values. The value highlighted in blue gives the probability that 4 consecutive channels in the respective sequence are either positive or negative.

The noise estimation routine of GaussPy+ uses a user-defined probability threshold $P_{\rm Limit}$ (default value: 2%) to decide which features get masked out for the noise calculation in a spectrum with $N_{\rm chan}$ channels. We use an iterative approach to calculate the minimum necessary number of consecutive positive or negative spectral channels n for which $p_{1,n} < P_{\rm Limit}$. We start by constructing a transition matrix for $n = 2$ and determine the $p_{1,n}$ value of $P^{N_{\rm chan}-1}$. If $p_{1,n} > P_{\rm Limit}$ we increase n by one and repeat the calculation. We stop these iterations once $p_{1,n} < P_{\rm Limit}$ and the final value of n determines the minimum number of consecutive positive or negative channels a feature has to have to get masked out. For example, for a spectrum with 700 spectral channels, features with more than 15 consecutive positive or negative spectral channels have a probability of less than 2% to be caused by random noise fluctuations and will be thus masked out in the noise calculation routine.

Appendix B: Testing GaussPy+ on synthetic spectra

B.1. Sample of synthetic spectra

We created four different samples of 10 000 synthetic spectra each, to mimic expected properties of spectra (see Fig. B.1 for examples of each sample):

A: White noise only.
B: White noise and signal. For spectra in this sample up to 12 Gaussian components ("signal") were added to the white noise of the spectra from sample A.
C: White noise, signal, and negative noise spikes. For spectra in this sample one or two negative Gaussian components ("noise spikes") were added to the spectra from sample B to mimic instrumental artefacts.
D: White noise, weak signal, and negative noise spikes. For spectra in this sample the positive Gaussian components from sample C had their amplitudes reduced. The signal peaks can thus be hidden in the noise, which makes the decomposition very challenging.

The synthetic spectra were set up to closely mimic spectra from the GRS data set with regards to the number of spectral channels (659), and expected noise and signal properties. The $\sigma_{\rm rms}$ value used to generate the white noise was randomly sampled from a Gamma distribution of the form

$$p\,(x) = x^{k-1}\,\frac{e^{-x/\theta}}{\theta^k \Gamma\,(k)}, \tag{B.1}$$

with $k = 2$, and $\theta = 0.35$. To closely mimic the noise distribution of the GRS survey (cf. Fig. 8 from Jackson et al. 2006) we shifted the distribution by a value of 0.06 and scaled it by a factor of 0.1 (panel a in Fig. B.2). The minimum $\sigma_{\rm rms}$ value of our sample is 0.06 K and we limited the maximum $\sigma_{\rm rms}$ value to 0.4 K.

The parameters of the Gaussian components of the signal were randomly sampled from distributions set up to resemble the signal peaks observed in the GRS data set. We sampled the FWHM values from a standard normal distribution scaled by a factor of ~13 (panel b in Fig. B.2). We limited the FWHM to a maximum value of 50 spectral channels. We sampled the amplitude values from another standard normal distribution scaled by a factor of 0.4 (panel c in Fig. B.2). We limited the amplitude range to values of $[3.5 \times \sigma_{\rm rms}, 2.5]$. We sampled the mean values of the Gaussians from a uniform distribution over all 659 spectral channels. For each spectrum, we required for every Gaussian signal component i that: its significance value $\mathcal{S}$ (Sect. 3.2.1) had to be >6; its mean position μ_i had to be at a minimum distance of Θ_j to the mean position μ_j of the closest Gaussian signal component j, where Θ_j is the FWHM of the Gaussian component j; its FWHM value Θ_i had to be <20 channels if its amplitude value a_i was >1. The last condition was implemented to exclude components with both high amplitude values and broad linewidths. This exclusion of the strongest

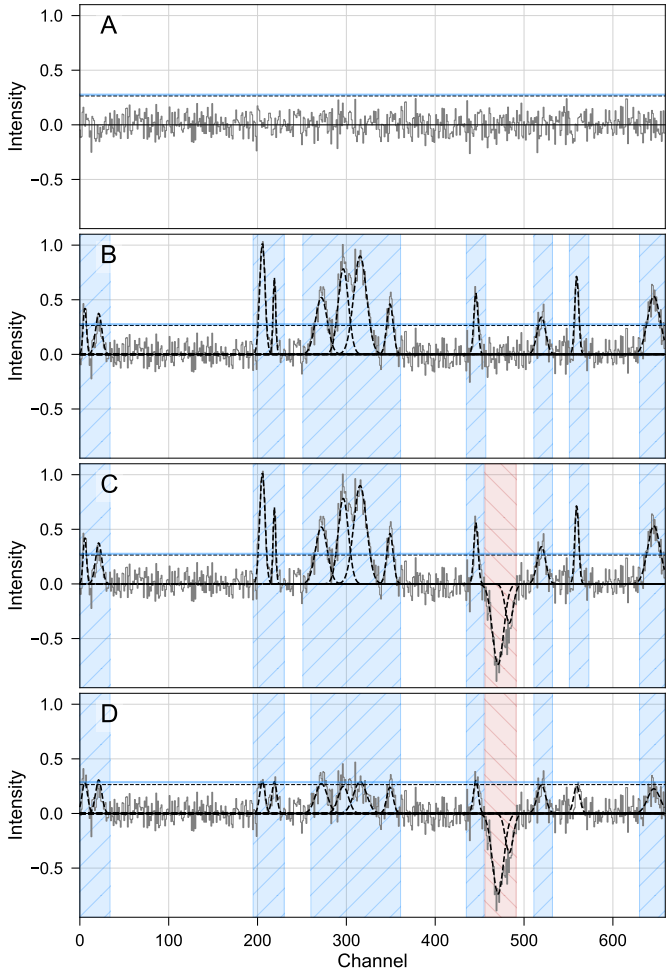

Fig. B.1. Example spectra from the four samples of synthetic spectra (A–D) used to test the performance of GaussPy+. Black dotted lines indicate individual Gaussian components of the signal and negative noise spikes. The horizontal dashed black lines show a S/N threshold of 3. Shaded areas indicate intervals that GaussPy+ classified as signal intervals (blue) and noise spikes (red). The noise is the same in all four panels.

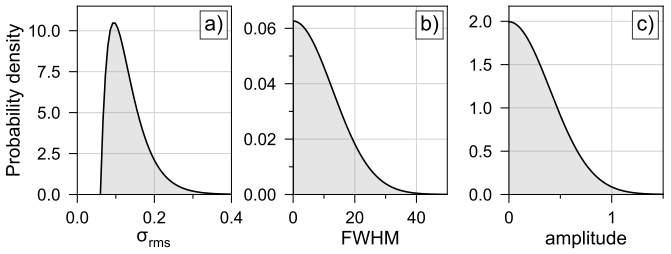

Fig. B.2. Probability distribution functions for $\sigma_{\rm rms}$ (*left*), FWHM (*middle*), and amplitude values (*right*). For the synthetic spectra, these distributions were randomly sampled to obtain the noise and Gaussian components of the signal.

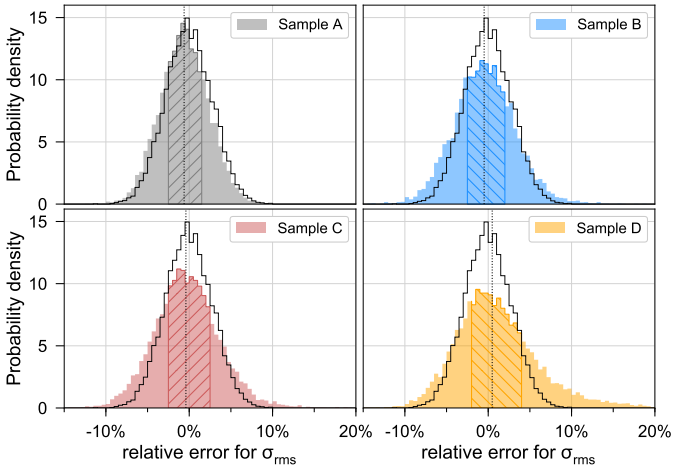

Fig. B.3. Probability density distributions showing the results of our automated noise estimation for the sample of synthetic spectra containing: only white noise (*upper left panel*); white noise and signal (*upper right panel*); white noise, signal, and negative noise spikes (*lower left panel*); white noise, weak signal, and negative noise spikes (*lower right panel*). The abscissa shows the determined root-mean-square noise value $\sigma_{\rm rms}$ normalised by the true root-mean-square noise value $\sigma_{\rm rms,\,true}$ that was used to generate the white noise. Hatched areas and vertical dotted lines show the respective interquartile ranges and median value of the respective distributions. The black solid line shows the distribution obtained by using all spectral channels from sample A for the noise calculation. See Appendix B.2 for more details.

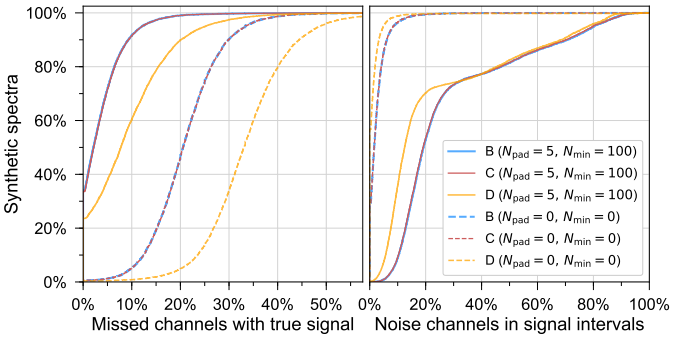

Fig. B.4. Results of the signal interval identification of GaussPy+ for our samples of synthetic spectra. *Left*: Cumulative percentage of the synthetic spectra showing the fraction of unidentified spectral channels containing true signal. *Right*: Cumulative percentage of the synthetic spectra showing the fraction of identified signal interval channels corresponding to noise. See Appendix B.3 for more details.

components was only done to create a more challenging setup for the decomposition. Data sets with low to moderate spatial resolution such as GRS are likely to contain such strong features that can be caused by the broadening of lines due to the large spatial beamsize and large distances to the emitting physical objects. However, these strong emission lines are fitted well with GaussPy and GaussPy+ in case no strong blending with other lines is present, as is the case for our samples of synthetic spectra.

The parameters for the negative Gaussian components of the noise spikes were randomly sampled in mean position, amplitude, and FWHM from uniform distributions within the limits $[0, 659]$, $[-4 \times \sigma_{\rm rms}, -1.5]$, and $[1, 20]$, respectively. We required that the noise spikes were placed at least a distance of Θ_j from the closest Gaussian signal component j. The amplitude values of the Gaussian components for sample D were sampled from a uniform distribution with the range $[2.5 \times \sigma_{\rm rms}, 3.5 \times \sigma_{\rm rms}]$.

B.2. Performance of the automated noise estimation routine

Here we report the results of the automated noise estimation of GaussPy+ (Sect. 3.1.1) on the synthetic spectra from samples A–D discussed in the last section. We used the default settings for the noise estimation routine ($P_{\rm Limit} = 2\%$, $N_{\rm pad} = 5$), which means that sequences above 15 consecutively positive or negative spectral channels get masked out for the noise estimation in addition to peaks that show high amplitude values.

Figure B.3 shows probability density distributions of the relative errors of the $\sigma_{\rm rms}$ values determined by GaussPy+. These relative errors were obtained by comparing the estimated $\sigma_{\rm rms}$ values to the true noise values ($\sigma_{\rm rms,\,true}$) used to generate the white noise for all four samples (A–D). For comparison, we also show the probability distribution obtained if all channels in the spectra of sample A are used for the calculation of the $\sigma_{\rm rms}$ value (solid black line). This distribution corresponds to the best we could do for the calculation of the $\sigma_{\rm rms}$ value and its spread around the $\sigma_{\rm rms,\,true}$ value reflects inherent random effects of the noise that would be decreased if the number of spectral channels were increased.

For the majority of the synthetic spectra the noise estimation performed very well with the median of the distribution (dotted vertical line) being very close to the $\sigma_{\rm rms,\,true}$ value and the interquartile ranges (hatched areas) within relative errors of ±3% and ±4% for samples A–C and sample D, respectively. Since the noise estimation always excludes the spectral channels with the highest negative and positive values (see Sect. 3.1.1), it tends to slightly underestimate the $\sigma_{\rm rms}$ value for spectra containing only noise (sample A). For sample B (white noise and signal), nearly all estimated $\sigma_{\rm rms}$ values are within ±10% of $\sigma_{\rm rms,\,true}$. For the spectra of sample C the performance of the noise calculation is almost as good, which demonstrates that our method is robust to the presence of negative noise spikes or similar instrumental artefacts. As expected, for sample D (white noise, weak signal, noise spikes) we tend to overestimate the $\sigma_{\rm rms}$ values. However, given that a fraction of the signal peaks in these spectra is buried within the noise, the noise calculation still performs very well, with $\sigma_{\rm rms}$ values within ±10% of $\sigma_{\rm rms,\,true}$ for about 93% of the spectra.

B.3. Performance of the identification of signal intervals

In this section we report on the results of the automated identification of signal intervals of GaussPy+ (Sect. 3.1.2) on our samples of synthetic spectra (Appendix B.1). We used the default settings of GaussPy+, with $S/N_{\rm min} = 3$, $\mathcal{S}_{\rm min} = 5$, $N_{\rm min} = 100$, and $N_{\rm pad} = 5$.

For sample A, whose spectra contain no signal, the signal identification had a false positive rate of 0.01%. That means out of a sample of 10 000 spectra with white noise there was only a single spectrum for which a signal interval was incorrectly identified.

The left panel in Fig. B.4 shows the cumulative percentage of the synthetic spectra as a function of unidentified spectral channels that contain true signal. We define the interval of channels containing true signal as all channels within $\mu_i \pm \Theta_i$ for a true Gaussian signal component i. For ~90% of the spectra in sample B and C, the fraction of unidentified spectral channels containing signal is <10%. In case of weak signal (sample D), the

percentage of unidentified spectral channels with signal is still <20% for ~90% of the spectra. This performance is very good, given that many of the signal peaks in sample D are by construction nearly indistinguishable from noise features (with their amplitude values ranging from only $2.5\times\sigma_{\rm rms}$ to $3.5\times\sigma_{\rm rms}$). The dashed lines indicate runs of the signal interval identification, for which we set the $N_{\rm pad}$ and $N_{\rm min}$ keywords to zero, meaning that there are no channels added on either side of the identified signal intervals. The left panel in Fig. B.4 demonstrates that we would miss a larger fraction of spectral channels containing true signal by setting $N_{\rm pad}$ and $N_{\rm min}$ to zero.

The right panel in Fig. B.4 shows the cumulative percentage of the synthetic spectra as a function of the fraction of noise channels included in the identified signal intervals, again for the two runs in which we vary the $N_{\rm pad}$ and $N_{\rm min}$ values as for the left panel. If $N_{\rm pad}$ and $N_{\rm min}$ are set to zero, only a very small fraction of noise channels is included in the estimated signal intervals. As expected, this fraction increases if we extend the signal intervals on both sides by $N_{\rm pad} = 5$ and require that the signal intervals contain a minimum number of channels per spectrum of $N_{\rm min} = 100$. However, this has no negative impact on the decomposition, since the signal intervals are only used in the goodness of fit calculations. It would be more problematic if we set $N_{\rm pad}$ and $N_{\rm min}$ to zero, because in that case we would miss a higher fraction of real signal, which would not be considered in the goodness of fit estimates.

We thus conclude that our method to estimate signal intervals works well. This good performance of the signal interval determination is also illustrated in Figs. 17, B.1 and B.7, where the estimated signal intervals are indicated with the blue shaded areas.

B.4. Performance of the masking of noise artefacts

In this section we report on the performance of GaussPy+ in automatically masking negative noise spikes (Sect. 3.1.3) for our samples of synthetic spectra (Appendix B.1). We used the default settings for the $S/N_{\rm spike}$ parameter that masks out all spectral features that contain negative values below $-5\times\sigma_{\rm rms}$.

Our routine managed to correctly identify 99.4% and 98.8% of all noise spikes with minimum values $<-5\times\sigma_{\rm rms,\,true}$ in the synthetic spectra of samples C and D, respectively. The small fraction of unidentified noise spikes with $S/N < -5\times\sigma_{\rm rms,\,true}$ was due to overestimates of the $\sigma_{\rm rms}$ values. The fraction of false positives – that means noise fluctuations that were incorrectly identified as noise spikes – was 0.02% for both samples. The performance of the masking of noise artefacts is also illustrated in Figs. B.1 and B.7, where the shaded red areas indicate the spectral channels identified as noise spikes.

B.5. Performance of the automated decomposition routine for the training set

As discussed in Sect. 3.1.4, GaussPy+ can supply a training set for the determination of the best smoothing parameters for a data set. Here we discuss the performance results of the automated decomposition of spectra for the training set. We quantify the performance by comparing the resulting smoothing parameters α_1 and α_2 obtained from the decomposed training set with the smoothing parameters obtained for the same training set if the true known Gaussian parameters are supplied. For the training sets, we randomly selected 250 synthetic spectra from samples B–D (Appendix B.1). We then created two training sets for each sample by: (i) decomposing the spectra via the method discussed in Sect. 3.1.4; (ii) supplying the true parameters for the Gaussian components of the synthetic spectra.

Table B.1. Comparison of obtained smoothing parameter values α_1 and α_2 and the corresponding F_1 score for different training sets.

Sample	α_1	α_2	F_1 score [%]
B	2.08	4.91	82.4
B (true)	2.03	4.91	82.7
C	2.11	4.89	79.0
C (true)	2.07	4.87	79.6
D	3.23	4.98	69.0
D (true)	3.44	5.09	71.5

Table B.1 lists the result of the gradient descent technique applied by GaussPy to determine the best smoothing parameters for the training sets. The run in which the true values of the Gaussian components were supplied in the training set is indicated with "(true)". For all runs the S/N for the spectrum and its second derivative were set to $\mathrm{SNR}_1 = \mathrm{SNR}_2 = 3$.

For sample B and C the runs for both training sets converge to essentially the same smoothing parameters α_1 and α_2. For sample D, the value for α_1 inferred from the training set decomposed with our routine is slightly smaller than the parameter we get from the true values. We tested the effect of this change by repeating the GaussPy decomposition for sample D with the smoothing parameter values $\alpha_1 = 3.44$ and $\alpha_2 = 5.09$. We then recomputed the percentage of correct identifications (30.4%) and false positives (6.9%) in the same way as for the values inferred from the decomposed training set given in Table 1 (29.4% and 6.5% for the correct identifications and false positives, respectively). This shows that the slight difference in the smoothing parameter inferred for sample D has only a limited impact on the GaussPy decomposition results.

The comparison in Table B.1 thus demonstrates that the automated method for creating training sets that is implemented in GaussPy+ works well. We thus conclude that smoothing parameters close to the optimal value can be obtained via this method.

B.6. Performance of the Gaussian decomposition

Here we compare the performance of the decomposition of the original GaussPy algorithm and the improved fitting routine of GaussPy+ (Sect. 3.2.3) on our samples of synthetic spectra (Appendix B.1). First, we explore how the performance of the decomposition results of GaussPy and GaussPy+ for sample B–D of the synthetic spectra varies with the number of components in the spectrum, the S/N, the width of the signal peaks, and the separation between signal peaks. We counted the mean position of fitted Gaussian components as correct if their values were within ±2 channels of the peak positions of the true underlying signal peak. We counted amplitude and FWHM parameters as correctly fit if their values were within ±20% of the true value in addition to the requirement that the fitted mean position is within ±2 channels of the true position of the component. Since for narrow signal peaks 20% of the FWHM can amount to only a fraction of a channel we additionally count a FWHM parameter as correctly fit if its absolute error is within ±2 spectral channels of the correct FWHM value.

The left panels in Fig. B.5 show the percentage of correctly identified Gaussian fit parameters (mean position, amplitude and FWHM value from top to bottom, respectively) as a function of the number of components in the spectrum. The GaussPy+

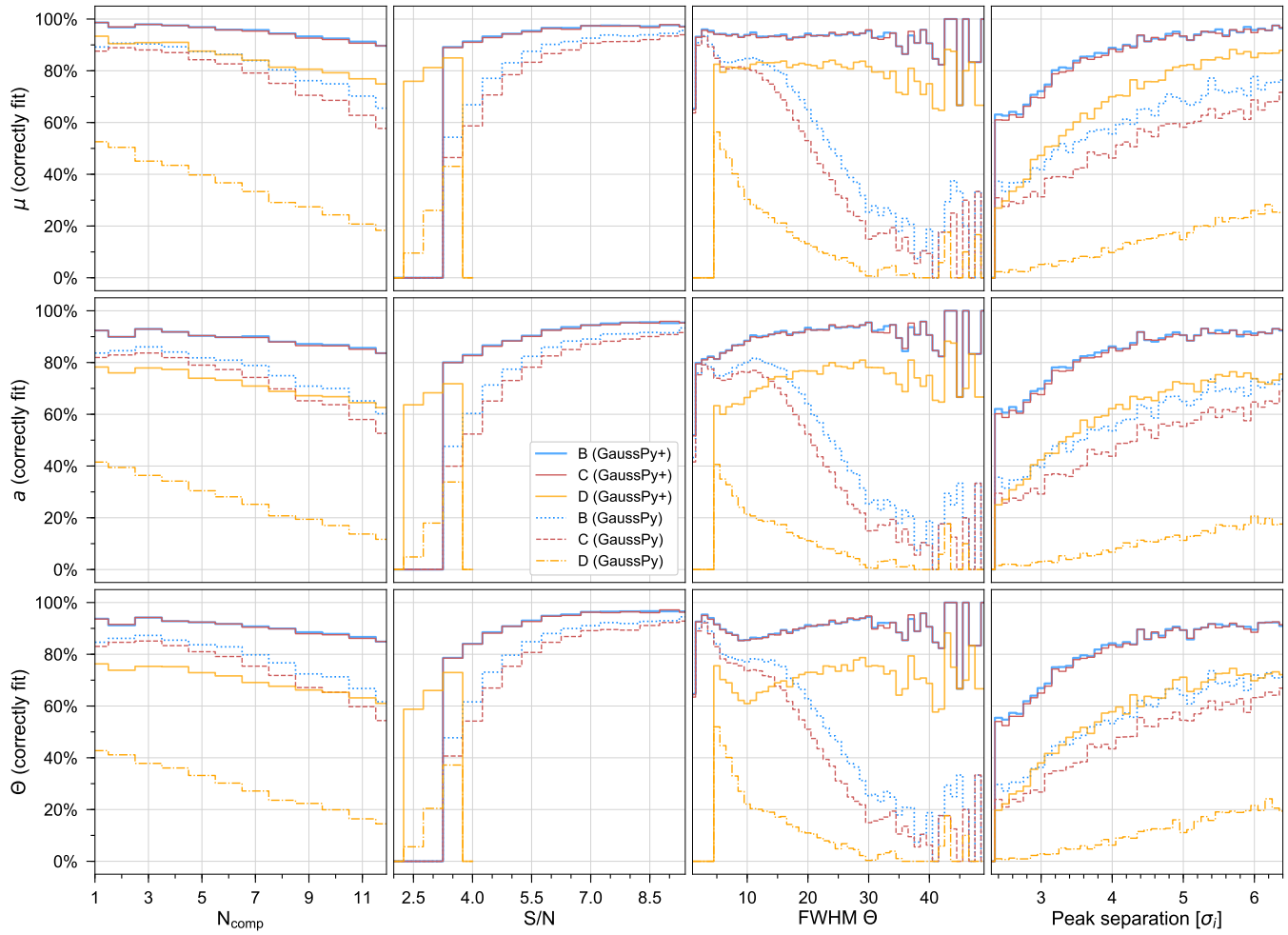

Fig. B.5. Performance of the GaussPy and GaussPy+ decomposition runs for samples of synthetic spectra. The ordinate in the *upper*, *middle*, and *lower panels* show the fraction of correctly fit Gaussian mean positions, amplitude values, and FWHM values, respectively, plotted against the number of true Gaussian components, the S/N, the true FWHM values, and the peak separation in the *left*, *centre left*, *centre right*, and *right* panels, respectively. See Appendix B.6 for more details.

decomposition shows a very stable performance that is not much affected by a higher number of components or the existence of noise spikes. Even in the case of signal peaks very close to the detection threshold (sample D) it still yields a good performance. In contrast, the ability of the original GaussPy algorithm to correctly decompose the components deteriorates by about 10−20% for the synthetic spectra of sample B and C, and up to 30% for sample D the more complex the spectra are.

The centre left panels in Fig. B.5 show the number of correctly determined Gaussian fit parameters as a function of the S/N. As expected, the performance results strongly depend on the S/N. However, compared to the results of the original GaussPy algorithm, the GaussPy+ decomposition gives a significantly better performance, especially in determining correct fit parameters for signal peaks with S/N values ≤3, which can be heavily affected by the noise.

The centre right panels in Fig. B.5 show the number of correctly determined Gaussian fit parameters as a function of the FWHM values of the true signal peaks. In contrast to the GaussPy fit results, the performance of the GaussPy+ decomposition does not deteriorate with increasing width of the signal peaks, which means that both narrow and broad components are well fitted. The decomposition with the original GaussPy algorithm shows a much stronger dependence on the line width, and has difficulties in correctly decomposing broader components.

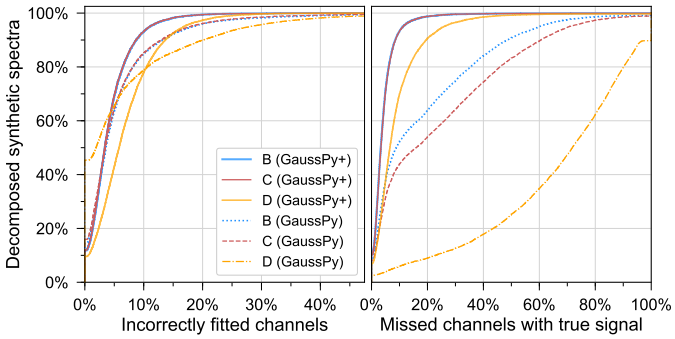

Fig. B.6. Comparison of the decomposition results obtained with GaussPy and GaussPy+ for our samples of synthetic spectra. *Left*: Cumulative percentage of decomposed spectra showing the fraction of spectral channels that were incorrectly fit. *Right*: Cumulative percentage of decomposed spectra showing the fraction of spectral channels containing true signal that were not fit. See Appendix B.6 for more details.

Finally, the right panels in Fig. B.5 show the percentage of correctly determined Gaussian fit parameters of signal component *i* as a function of peak separation to its closest neighbouring

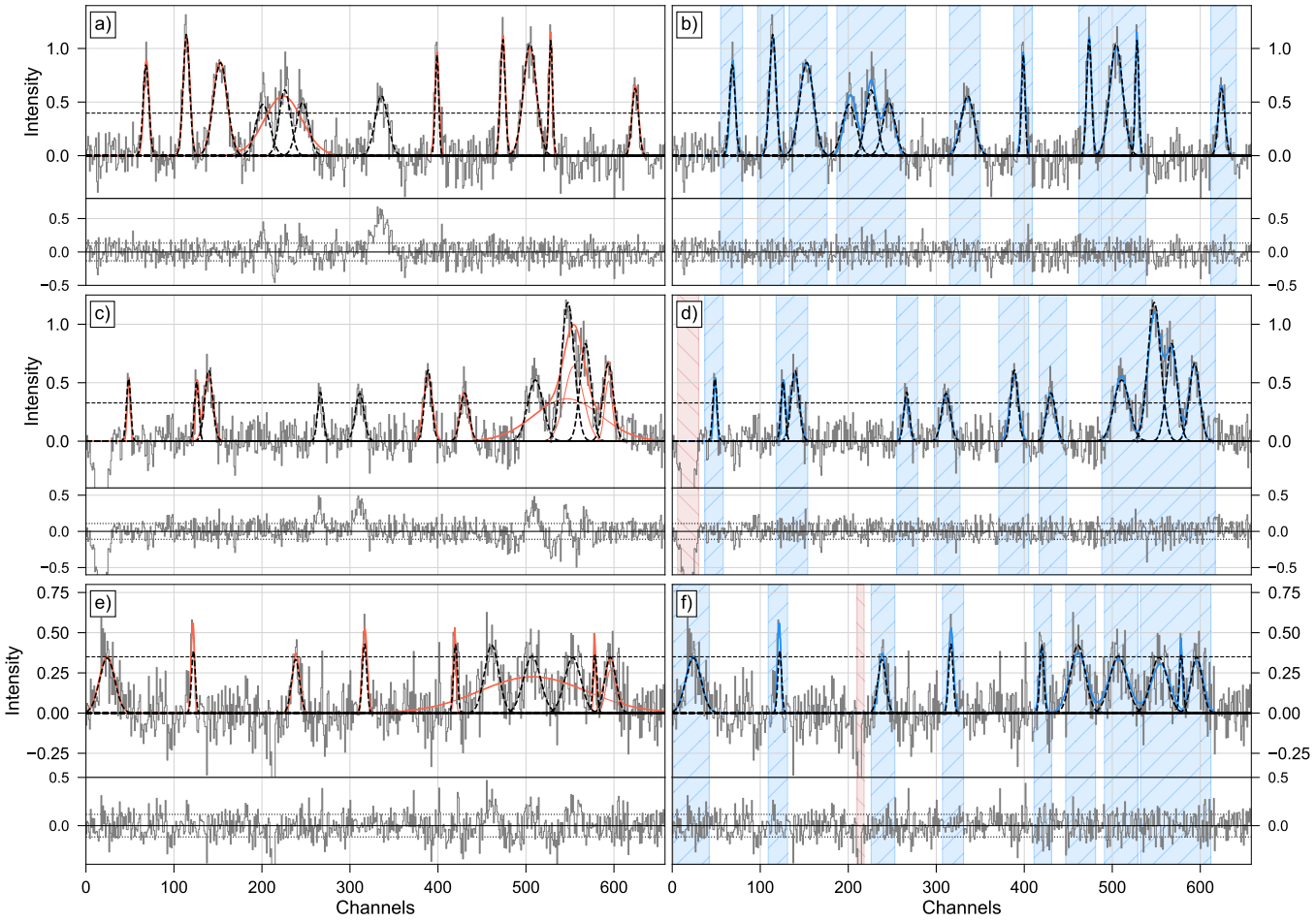

Fig. B.7. Example spectra illustrating the better performance of the improved fitting routine of GaussPy+ (Sect. 3.2.3) compared to the original GaussPy algorithm. *Upper* (*a, b*), *middle* (*c, d*), and *lower* (*e, f*) *panels*: synthetic spectra from samples B, C, and D, respectively. The panels on the *left* (*a, c, e*) show the decomposition results obtained with the original GaussPy algorithm and panels on the *right* (*b, d, f*) show the corresponding decomposition results from the improved fitting routine of GaussPy+. The correct individual Gaussian components are indicated in dashed black lines; individual Gaussian components and their combined intensity from the decomposition run with GaussPy and GaussPy+ are indicated in solid red and blue lines, respectively. The smaller panels below the spectrum show the corresponding residual with the dotted black lines indicating values of $\pm\sigma_{\rm rms}$. Dashed black lines indicate a S/N of 3. Blue and red shaded areas show the automatically identified signal and noise spike intervals, respectively.

signal components j. This peak separation is given as multiples of the standard deviation σ_i of component i. As expected, the performance of the decomposition with GaussPy+ decreases the closer two components are placed to each other as it gets exceedingly more difficult to correctly deblend them. Nonetheless, the decomposition with GaussPy+ manages to fit about ∼60% of even the most heavily blended components in sample B and C correctly, which exceeds the performance of GaussPy by more than 20%. For the challenging weak signal peaks of sample D, the fraction of correctly decomposed components that were blended the most was lower (∼20–30%). However, the percentage of correct fits increases already significantly for moderate peak separations of ∼3–4 × σ_i and reaches a stable high performance for even larger peak separations. We test the performance of GaussPy+ for blended components in more detail in Appendix B.7.

We also note that the GaussPy decomposition results deteriorate in case of the presence of negative noise spikes. However, the performance of GaussPy+ is unaffected by these negative noise spikes, as can be seen by the almost overlapping blue and red solid lines in all panels shown in Fig. B.5.

We tried to choose fair criteria for the definition of when we count components in Fig. B.5 as correctly fit. Given that many of our signal peaks show only low to moderate S/N values, noise properties might already severely affect their lineshapes, so stricter criteria would not accept decomposition results that a human would likely classify as correctly fit. Conversely, more relaxed criteria could allow too large absolute deviations from the correct parameter values. However, we repeated the analysis of Fig. B.5 for both stricter and more relaxed criteria and we do recover the same general trends: performance results that exceed the decomposition of GaussPy and are almost unaffected by the number of components in the spectrum, the FWHM value or the presence of noise spikes, and increase with higher S/N values or larger peak separations.

Next, we compare the number of fitted spectral channels with the channels containing true signal for the GaussPy and GaussPy+ decompositions. We define the interval of fitted channels or channels containing true signal as all channels within $\mu_i \pm \Theta_i$ for a fitted or true Gaussian component i. The left panel in Fig. B.6 shows the cumulative percentage of decomposed synthetic spectra as a function of the percentage of incorrectly

fitted spectral channels. Both GaussPy and GaussPy+ show a very good performance with a low fraction of false positives. The improved results of GaussPy+ are due to its ability to more correctly identify individual signal peaks where GaussPy fits a single component over multiple peaks.

The right panel in Fig. B.6 shows the cumulative percentage of decomposed synthetic spectra as a function of spectral channels containing true signal that were not fit by Gaussian components. For all three samples of synthetic spectra GaussPy+ significantly improves the decomposition results of GaussPy by fitting more components at their correct positions. The improvement is especially striking in case of the spectra from sample D that contain only weak signal.

Figure B.6 thus illustrates that GaussPy+ manages to fit significantly more channels containing true signal than GaussPy. Moreover, GaussPy+ does not fit too many noise features.

The improved performance of GaussPy+ is further illustrated in Fig. B.7, which contrasts decompositions of the original GaussPy algorithm (left panels) with decompositions obtained with our improved fitting routine (right panels) for synthetic spectra from samples B–D. Figure B.7 shows that the original GaussPy algorithm sometimes has problems in decomposing mildly blended signal peaks and signal peaks at the edge of the spectrum, whereas GaussPy+ has no problems in fitting those components correctly. The GaussPy+ algorithm also does a good job of identifying signal peaks and noise artefacts.

B.7. Recovery of identical components with different S/N and degrees of blendedness

Here we quantify how well the improved fitting algorithm of GaussPy+ (Sect. 3.2.3) is able to recover blended components. For this, we create a sample of synthetic spectra that contain two identical Gaussian signal peaks. We vary the parameters of the signal peaks between the following values: $[3, 3.5, \ldots, 7]$ for the S/N; $[5, 10, \ldots, 30]$ spectral channels for the FWHM; and $[1, 1.2, \ldots, 5] \times \sigma_i$ for the separation of the mean positions of the signal peaks. We created ten spectra of each possible parameter combination for a total of 11 340 spectra and added different noise sampled from a $\sigma_{\rm rms}$ value of 0.13 to each spectrum[17].

We constructed a training set by randomly selecting 500 spectra of different parameter combinations and inferred smoothing parameters α_1 and α_2 by supplying the true values of the signal peaks. Since our aim here is to establish the performance of our decomposition given ideal settings, we supplied the true paramater values as solutions instead of decomposing the training set with the method described in Sect. 3.1.4. From this training set we inferred smoothing parameters values of $\alpha_1 = 2.16$ and $\alpha_2 = 6.19$ that led to an F_1 score of 76.8%. We then performed decompositions with the original GaussPy algorithm and the improved fitting routine of GaussPy+, leaving all the settings at their default values.

Figure B.8 shows the performance results of the two decomposition runs. The left panel shows the percentage of fits using two Gaussian components as a function of peak separation, split into a sample with low to moderate S/N (<5, dashed lines) and high S/N (≥5, solid lines). The vertical dotted line indicates the separation threshold for two identical Gaussian components in case of no noise (see also Sect. 3.2.2). For low to moderate S/N

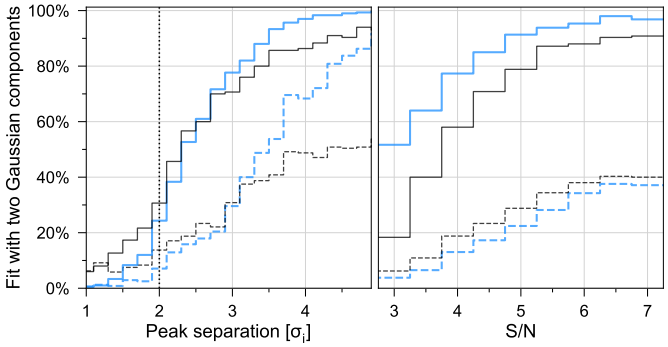

Fig. B.8. Decomposition results of a sample of synthetic spectra with two identical Gaussian components, whose S/N, peak separation and FWHM parameter were varied. Blue and black lines indicate the results obtained for the decomposition runs with GaussPy+ and GaussPy, respectively. *Left*: Percentage of fitted spectra with two Gaussian components as a function of peak separation for $S/N < 5$ (dashed lines) and ≥5 (solid lines). The dotted vertical line indicates the separation threshold for two identical Gaussian components without noise. *Right*: Percentage of fitted spectra with two Gaussian components as a function of their S/N for peak separations of $<3 \times \sigma_i$ (dashed lines) and $\geq 3 \times \sigma_i$ (solid lines).

it becomes very difficult to differentiate two similar Gaussian components if their peak positions are separated by less than about 3.5 times their standard deviation. For higher S/N identical signal peaks can be located closer together until they essentially become indistinguishable from a single component. For the decomposition with GaussPy+, the signal peaks need to have a distance to each other of more than $\sim 2.5 \times \sigma_i$ until the majority of signal peaks will be fit with two components. The GaussPy+ algorithm by design fits preferentially a single instead of two components if the peaks are only separated closely, as in such cases a fit with a single Gaussian component will already be a good match to the combined signal peaks and the simplest fit solution is preferred without additional information (e.g. from neighbouring fit solutions) to inform the fit. For larger peak separations GaussPy+ exceeds the performance of GaussPy, especially in the case of low to moderate S/N values.

The right panel of Fig. B.8 shows the percentage of decomposition results using two Gaussian fit components, split into two samples with small ($<3 \times \sigma_i$, dashed lines) and large ($\geq 3 \times \sigma_i$, solid lines) peak separations. For small peak separations, GaussPy and GaussPy+ will preferentially fit the signal peaks with a single component, even if the S/N is high. For larger peak separations the two-component fit solution is dominant and the percentage of spectra fit with two components increases significantly for high S/N.

Since the decomposition was performed without any additional knowledge about the signal peaks (as could be imposed by neighbouring spectra in spatially coherent decompositions), it can become very challenging to correctly fit signal peaks with low S/N, as random fluctuations of the noise can significantly change their shape. Moreover, the two identical signal peaks we placed in the spectra will combine to a symmetric peak that might be equally well fit by a single or two components if they are heavily blended. Spectral features of two blended components of different shape will cause an asymmetry that can make it easier to decompose them correctly.

[17] The number of spectral channels (659) and the $\sigma_{\rm rms}$ value were again chosen to closely mimic properties of the GRS data set.

Appendix C: Performance details for GaussPy+

C.1. Performance and execution time for the decomposition of the training set

Table C.1. Comparison of the execution times for sample B of the synthetic spectra.

	$t_{\rm real}$ [min]	$t_{\rm CPU}$ [min]
SLSQPLSQFitter	43.06	1868.87
GaussPy+	2.59	110.01

Table C.2. Comparison of the execution times for the GaussPy+ decomposition of the GRS test field.

Method	$t_{\rm real}$ [min]	$f_{\rm real}$ [%]	$t_{\rm CPU}$ [min]	$f_{\rm CPU}$ [%]
Training set creation	0.39	4.5	16.33	21.3
Training	5.61	64.2	20.86	27.2
Preparation	0.08	0.9	0.28	0.4
Decomposition stage 1	0.32	3.6	9.55	12.4
Decomposition stage 2	0.44	5.0	8.03	10.5
Decomposition stage 3	1.91	21.8	16.69	21.8

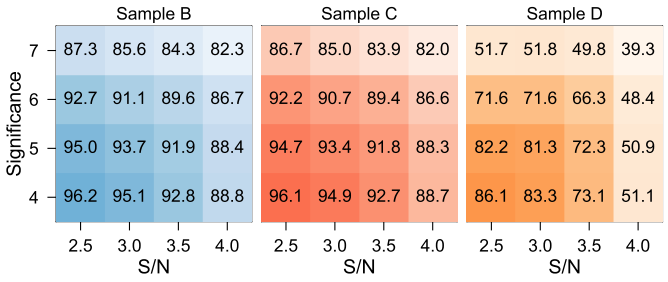

Fig. C.1. Percentage of correctly identified mean positions of Gaussian components in the decomposition of samples B–D (*left to right*) with varying values for the minimum S/N and significance parameters.

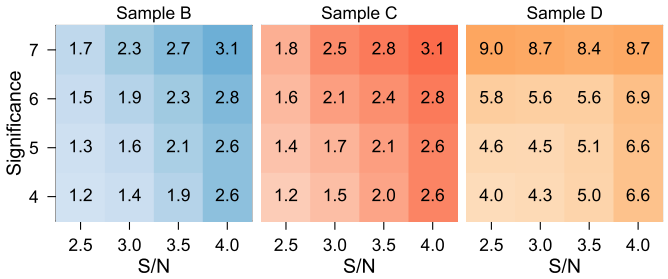

Fig. C.2. Percentage of incorrectly identified mean positions of Gaussian components in the decomposition of samples B–D (*left to right*) with varying values for the minimum S/N and significance parameters.

We compared the decomposition results and runtime of the SLSQPLSQFitter fitting routine used to create training sets for GaussPy (Sect. 3.1.4) with the runtime of the improved fitting routine of GaussPy+ (Sect. 3.2.3). We used both fitting techniques to decompose sample B of our synthetic spectra (Appendix B.1). For both algorithms we distributed the decomposition over 50 CPUs.

In terms of performance the decomposition with the SLSQPLSQFitter could correctly identify 95.4% of the signal components and had a false positive fraction of 1.5%. Both of these values exceed the corresponding numbers for the results of GaussPy+ (93.7% and 1.6%, respectively, cf. Table 1), which confirms that our routine for creating training sets produces high quality decompositions.

Table C.1 lists the results of the execution times: $t_{\rm real}$ is the elapsed wall clock time from start to finish of the execution of the decomposition and $t_{\rm CPU}$ is the total amount of spent CPU time. These results show that the SLSQPLSQFitter fitting routine is about an order of magnitude slower than GaussPy+, which is why we recommend to use the former routine only for the decomposition of spectra for the training set.

C.2. Execution time for the GRS test field

In this section we discuss the execution time of the GaussPy+ algorithm for the decomposition of the GRS test field using the default settings of GaussPy+ and distributing the computation over 50 CPUs. Table C.2 shows an overview of the execution time for all stages of GaussPy+ in terms of wall clock time $t_{\rm real}$ and total CPU time $t_{\rm CPU}$ as well as their respective relative percentages $f_{\rm real}$ and $f_{\rm CPU}$. The entire GaussPy+ decomposition for the GRS test field needed $t_{\rm real} = 8.74$ min and $t_{\rm CPU} = 76.74$ min.

Since the total size of the GRS test field (4200 spectra) is relatively small, the creation of the training set and training with GaussPy amounted to a significant contribution to $f_{\rm real}$ and $f_{\rm CPU}$, which would be reduced for larger data sets, where the decomposition steps will need a larger fraction of the total time. We also report the individual times for the execution of the three decomposition stages of GaussPy+: the improved fitting routine (Sect. 3.2.3; Stage 1), phase 1 of the spatially coherent refitting (Sect. 3.3.1, Stage 2), and phase 2 of the spatially coherent refitting (Sect. 3.3.2, Stage 3). Execution times for the spatially coherent refitting stages will typically depend on how many criteria are used in the flagging of spectra in Stage 2 and the minimum weight threshold $\mathcal{W}_{\rm min}$ the user selects in Stage 3 (cf. Fig. C.3).

C.3. Effect of varying minimum S/N and significance

Here we test how changing the values of the minimum S/N $S/N_{\rm min}$ and significance parameter $\mathcal{S}_{\rm fit}$ affects the decomposition results for our samples of synthetic spectra (Appendix B.1). We use $S/N_{\rm min}$ and $\mathcal{S}_{\rm fit}$ values of $[2.5, 3, 3.5, 4]$ and $[4, 5, 6, 7]$ respectively, and perform a decomposition with GaussPy+ for every combination of those values (16 in total). For the spectra of sample A that contain only white noise we found that with significance values $\mathcal{S}_{\rm fit} \geq 5$ no noise features were fitted. For a significance value of $\mathcal{S}_{\rm fit} = 4$ and $S/N_{\rm min}$ values of 2.5, 3, 3.5, and 4, GaussPy+ incorrectly fitted 38, 31, 21, and 8 noise features, respectively.

For samples B–D we calculated the percentage of correctly and incorrectly fitted mean position values of Gaussian components for each decomposition run, which are shown in Figs. C.1 and C.2, respectively. We count the mean position of a Gaussian component as correctly detected if it is within ±2 channels of the true value. If the mean position value of a fitted component was more than 4 channels away from the true mean positions of all signal components in the spectrum we counted it as an incorrect identification. The decomposition with $\mathcal{S}_{\rm fit} = 5$ and $S/N_{\rm min} = 3$ corresponds to the GaussPy+ run at its default

settings we presented in Sect. 4. Figures C.1 and C.2 demonstrate the interdependence between the $\mathcal{S}_{\rm fit}$ and $S/N_{\rm min}$ parameters. In general, increasing one of these parameters has adverse effects on the percentage of correct and incorrect detections of Gaussian components in the synthetic spectra of samples B–D. However, this adverse effect can be offset by decreasing the value for the other parameter. The results from Fig. C.2 show that setting the $\mathcal{S}_{\rm fit}$ and $S/N_{\rm min}$ parameters to higher values can lead to a big increase in incorrectly identified fit components. This increase is due to the large fraction of signal components with low S/N in our synthetic spectra. If we set the $S/N_{\rm min}$ or $\mathcal{S}_{\rm fit}$ parameters to higher values, those components are either prevented from being fit or are incorrectly fit with one broad component instead of multiple narrower ones.

Figures C.1 and C.2 also demonstrate that we could have improved the decomposition results reported in Sect. 4 by choosing a lower minimum S/N of $S/N_{\rm min} = 2.5$. In principle we could have even further improved upon that result by also decreasing the required significance value to $\mathcal{S}_{\rm fit} = 4$, but the results from the decomposition of sample A demonstrate that this setting would already allow the fit of noise features.

Ultimately, the choice for the values of the $S/N_{\rm min}$ and $\mathcal{S}_{\rm fit}$ parameters needs to be guided by the data set. For the synthetic spectra we used perfect Gaussian noise properties, which will likely not be the case for real observational data. Thus users might want to set higher values for the $\mathcal{S}_{\rm fit}$ parameters to exclude the fitting of noise features, even though it might result in a reduction of fitted weaker signal peaks. We constructed the synthetic spectra of samples B–D to contain a large fraction of signal peaks with amplitudes close to or even below a S/N of 3 to test how well the GaussPy+ decomposition with default settings works for weak signal peaks. Decompositions of data sets for which users expect signal peaks with high S/N will thus likely benefit from an increase of the values for the $S/N_{\rm min}$ and $\mathcal{S}_{\rm fit}$ parameters.

C.4. Performance of in-built and optional quality control procedures

Here we discuss the performance of the in-built and optional quality control procedures described in Sects. 3.2.1 and 3.2.2. While we report here only on the performance of the in-built quality criteria for the improved fitting routine, these criteria are also used in all refit attempts in the spatially coherent refitting phases.

For ∼34% of the spectra of the GRS test field at least one of the in-built quality control procedures was used to remove one or more components in the decomposition with the improved fitting routine of GaussPy+. For sample A–D of the synthetic spectra the percentage of spectra for which components were removed due to failing the in-built quality controls was ∼3%, 20%, 20% and 22%, respectively. The comparatively larger fraction of spectra with rejected fit components in the decomposition of the GRS test field was mostly due to the presence of low-intensity signal peaks that did not satisfy the criterion for the amplitude value and imperfect noise properties, which led to the fitting of noise peaks that did not satisfy the requirement for the significance value.

Table C.3 gives the exact number of fit components that were removed due to the in-built quality controls using the default settings of GaussPy+. In general, the significance criterion was most often used and thus is the strictest criterion, followed by the requirement of a minimum S/N value for the fitted amplitude and a minimum value for the fitted FWHM. Since the synthetic spectra were set up to also contain emission in the outermost channels, the criterion checking whether the fitted mean position was within the channel range was also used frequently to correct fit results for these spectra. The sequence of how the in-built quality controls are used matters, as for example a component that already failed the requirements for the amplitude value will not be subjected to the significance criterion anymore (cf. Fig. 3). Thus, had we checked the significance criterion first, it would have been responsible for removing even more components.

Table C.3. Number of fit components removed by the in-built quality control procedures (Sect. 3.2.1) for the decomposition of the GRS test field and the synthetic spectra.

	Θ	a	$\mathcal{S}_{\rm fit}$	μ
GRS test field	136	705	1127	16
Sample A	17	7	263	0
Sample B	673	669	836	506
Sample C	690	632	829	492
Sample D	447	585	1975	434

Table C.4. Number of new best fit solutions obtained by utilising the optional quality control procedures (Sect. 3.2.2) for the decomposition of the GRS test field and the synthetic spectra.

	neg. res. peak	Broad	Blended
GRS test field	25	542	14
Sample A	0	0	0
Sample B	133	353	40
Sample C	137	354	49
Sample D	2	683	1

Table C.4 lists the number of successful refits based on the optional quality control procedures for refitting negative residual features, broad and blended components for the GRS test field and the four samples of synthetic spectra. The refitting of broad fit components into multiple narrower individual components was the criterion that led to most successful refits, followed by the refitting of components that caused negative residual features. This mostly reflects the generally low S/N values of signal peaks in the spectra, for which GaussPy often fits a single broad Gaussian component over multiple individual signal peaks (cf. Fig. B.7). The refitting of features labelled as blended did not yield that many successful refits. The low success rate for refits based on the criterion for blended components is expected for the signal peaks in the synthetic spectra, which were constructed in such a way as to not show heavily blended components. For the GRS test field, deviations of emission lines from a Gaussian shape could have caused the fit of multiple blended components, which resulted in low residuals or AICc values that could not be matched with the fit of a single component.

C.5. Refit iterations of the spatially coherent refitting phases

In this section we discuss the performance of the two phases of spatially coherent refitting (Sects. 3.3.1 and 3.3.2). The total number of refit iterations needed in these two phases depends on the size of the spectral cube, the number of flags set in phase 1, and the minimum required weight threshold $\mathcal{W}$ chosen in phase 2.

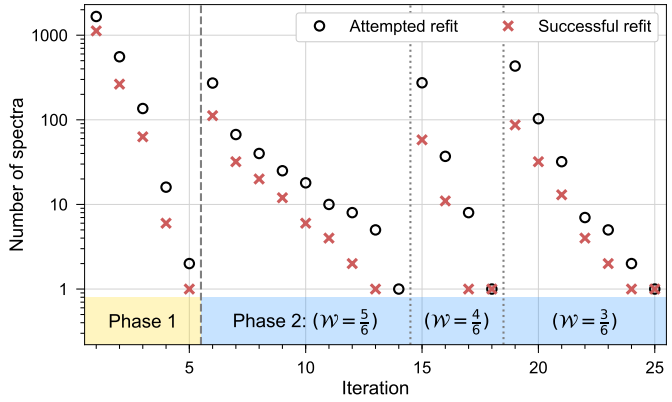

Fig. C.3. Number of refit attempts and successful refits of spectra of the GRS test field for each iteration in the two phases of the spatially coherent refitting. See Appendix C.5 for more details.

For the GRS test field, the two spatially coherent refitting phases needed 25 iterations in total to converge to a final fit solution. Figure C.3 shows the number of attempted and successful refits for all iterations. Most of the attempted and successful refits occur in phase 1, which needed 5 iterations. Since in a new iteration we will only refit spectra if they had not been flagged in the previous iteration or at least one of the fit solutions of its neighbours got updated, fewer spectra will be refit in each progressing iteration, which is demonstrated by the steep decrease of refit attempts in Fig. C.3. For example, in the first iteration of phase 1, 1839 out of the 4200 spectra were flagged and selected for refitting. The GaussPy+ algorithm tried to refit 1664 of these flagged spectra[18] with new fit solutions derived from neighbouring spectra, ~68% of which received a new best fit solution. In the second iteration, GaussPy+ only tried to refit 556 flagged spectra, of which ~47% obtained a new best fit solution.

Figure C.3 further shows the performance of phase 2 of the spatially coherent refitting, which proceeded in three stages, since in the default settings of GaussPy+ the minimum required weight threshold $\mathcal{W}$ is reset to a lower value two times. The runtime of phase 2 can therefore be decreased by setting a higher minimum weight threshold (e.g. $\mathcal{W}_{\rm min} = 4/6$), which should already lead to good spatial coherence between the neighbouring fit solutions.

In terms of total added and subtracted number of components for the decomposition of the GRS test field, phase 1 removed 226 components and added 295, whereas phase 2 subtracted 84 components and added 191 components. About 13% of the added components in phase 2 led to fit solutions being flagged as blended.

Appendix D: Normality tests

As discussed in Sect. 3.2.1, as a goodness of fit check we subject the normalised residual to two normality tests to decide whether the data points of the residual are normally distributed and thus consistent with Gaussian noise. We found that a combination of the two-sided Kolmogorov–Smirnov (K–S) test (Kolmogorov 1933; Smirnov 1939) and the normality test based on D'Agostino and Pearson (D–P; D'Agostino 1971; D'Agostino & Pearson 1973) yielded the most reliable means to detect unfitted signal peaks in the residual.

[18] For the remaining 175 flagged spectra no unflagged neighbouring fit solutions were available.

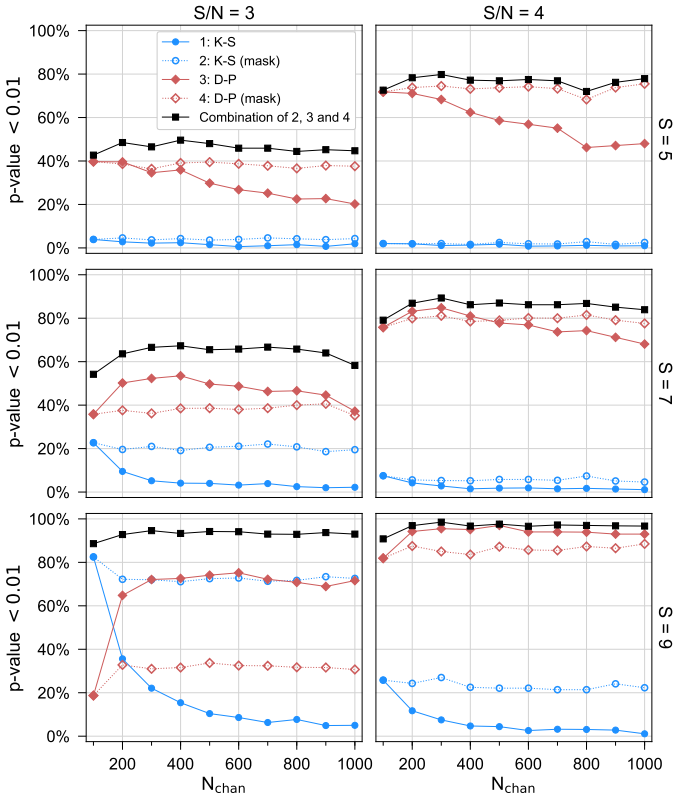

Fig. D.1. Comparison of the performance of different normality tests for mock residuals as a function of the number of spectral channels. The residuals contain a single Gaussian signal component with a signal-to-noise ratio of 3 or 4 (*left and right panels*, respectively) and significance values of 5, 7, and 9 (*upper, middle, and lower panels*, respectively). See Appendix D for more details.

We tested the performance of each normality test for mock residuals that we created by adding a single Gaussian component to white noise. We used six different combinations of the S/N and significance values for the Gaussian components. We also varied the number of spectral channels between 100 and 1000 in steps of 100. We produced 1000 spectra for each possible combination of Gaussian signal component and number of spectral channels for a total of 60 000 spectra. We then applied the normality tests to each of these mock residuals to check which test could most reliably identify the leftover signal component by rejecting the null hypothesis of normally distributed residual values.

Figure D.1 shows the performance of the normality tests for the different combinations. On the ordinate we plot the percentage of spectra for which the normality tests yielded p-values below the default threshold in GaussPy+ of 1%, which is used as an indication that the residual data points are not normally distributed. The results of the K–S and D–P test are shown in blue and red, respectively. Moreover, we applied both normality tests on the whole residual and only the residual data points within the identified signal ranges, which is indicated by the filled and unfilled symbols, respectively. The black line shows the performance of the null hypothesis testing included in GaussPy+, which combines the results of the D–P test applied to the full residual and the results of the D–P and K–S tests applied to only the residual data points within the identified signal intervals. For this combination, we use the smallest p-value resulting

Table D.1. Percentage of false positives identified by the normality tests.

Test	Combination	K–S	D–P
Median	2.2%	5.5%	7.6%
Minimum	1.4%	0.3%	1.1%
Maximum	3.3%	11.5%	13.9%

from these three normality tests. Figure D.1 demonstrates that this combination results in an increased ability to detect leftover signal peaks in the residual for both narrow and broad components (with low and high significance values, respectively). We are able to identify the majority of residual peaks in every tested case apart from the one with the weakest component ($\mathcal{S} = 5$, $S/N = 3$). The identification fraction reaches nearly 100% for the strongest tested residual peaks ($\mathcal{S} = 9$, $S/N = 4$). Moreover, this improved performance is independent of the number of spectral channels. In comparison, the individual results of the K–S and D–P tests show a decreased performance and even a complementary behaviour for broader Gaussian residual peaks with lower S/N values ($\mathcal{S} = 9$, $S/N = 3$) and low number of spectral channels (<300).

To check the fraction of false positives identified by the normality tests, we checked their performance also for Gaussian noise only, for which we removed the signal component from all residuals used in Fig. D.1. We evaluate the spectra again in groups of 1000 spectra and report the median, minimum and maximum false positive rate for all groups as the fraction of spectra for which the hypothesis tests yielded a p-value <1% and thus would not pass our criterion for normally distributed residuals.

Table D.1 lists the false positive rates. The combination of normality tests as implemented in GaussPy+ leads to the best performance over different channel ranges, as evidenced by the reduced median false positive rate compared to the individual normality tests. The K–S and D–P tests produce higher false positive rates with increasing numbers of spectral channels, whereas the combination of the two tests performed best for the highest number of spectral channels we probed.

Appendix E: $\chi^2_{\rm red}$ calculations for the GRS test field

A problem in determining the $\chi^2_{\rm red}$ value is that it depends on the number of channels in the spectrum. If the spectrum consists of many channels that contain only noise, low χ^2 values and $\chi^2_{\rm red}$ values close to 1 follow even if the performance of the fit is not satisfactory in the part of the spectrum where there is signal.

To avoid this problem we identify the regions likely to contain signal already in the noise estimation step (see Sect. 3.1.2) and use only these regions for the $\chi^2_{\rm red}$ calculations. We also mask negative noise spike features that tend to produce high $\chi^2_{\rm red}$ values even for spectra whose signal features were well fit (see Sect. 3.1.3).

To illustrate the importance of restricting the $\chi^2_{\rm red}$ calculation to intervals containing signal we recomputed the goodness of fit calculations for the decomposition results of the GRS test field (Sect. 5.3) obtained with GaussPy and after stage 3 of GaussPy+ by using all available spectral channels. Panels a and b in Fig. E.1

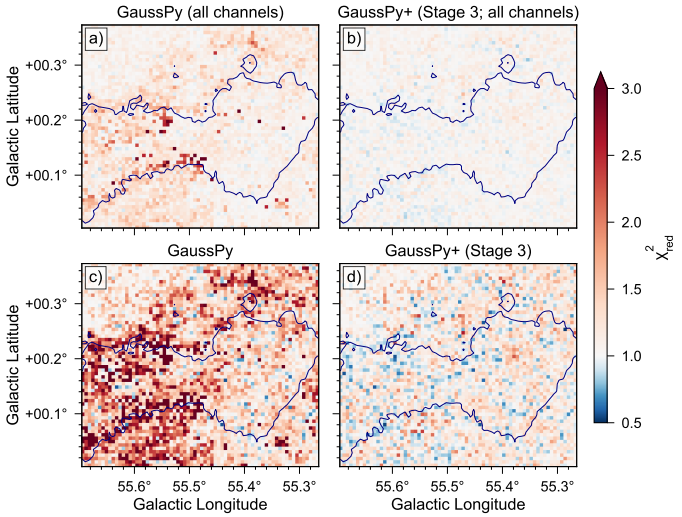

Fig. E.1. Maps showing the $\chi^2_{\rm red}$ values for the GaussPy (*left panels*) and stage 3 of the GaussPy+ (*right panels*) decomposition results, calculated by using either all available spectral channels (*upper panels*) or restricted to the spectral channels estimated to contain signal. All panels are overplotted with the contour from *panel b* in Fig. 13. *Panels c* and *d* are identical to *panels i* and *l* in Fig. 16.

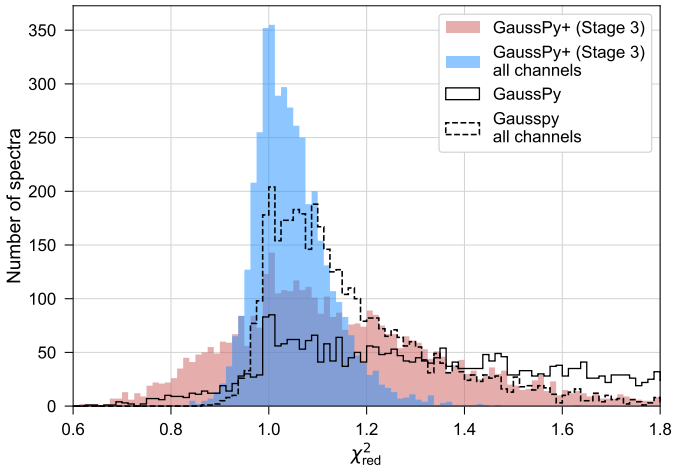

Fig. E.2. Comparison of the distribution of the $\chi^2_{\rm red}$ values for the decomposition results of GaussPy and Stage 3 of GaussPy+ restricted to spectral channels estimated to contain signal and calculated over the whole spectral range.

show the recomputed $\chi^2_{\rm red}$ values using all 424 spectral channels. For comparison, we also show the maps of $\chi^2_{\rm red}$ values again that were obtained by restricting the goodness of fit calculations to spectral channels estimated to contain signal (panels c and d, which are identical to panels m and p in Fig. 16). Figure E.2 gives the corresponding histograms. Both figures clearly illustrate how the goodness of fit values are artificially reduced if most of the spectral channels included in the calculation contain only noise. Using all available spectral channels for the goodness of fit calculations thus makes it more challenging to use the $\chi^2_{\rm red}$ values to decide which fit results were not successful.

Appendix F: Symbols, GaussPy+ keywords and default values

Table F.1 gives the description of symbols used throughout the text. Table F.2 gives an overview of the parameter settings of GaussPy+, listing their corresponding default values and symbols used throughout the text. To get first decomposition results users only need to supply values for the parameters listed under *essential parameters*. In case the decomposition does not yield good results we recommend to first use different values for the *essential parameters*. If this should not improve the results users can vary the parameters listed under *more advanced settings*.

Table F.1. Symbols used throughout the text.

Symbol	Description
μ_i	Offset or mean position of Gaussian fit component i
a_i	Amplitude of Gaussian fit component i
σ_i	Standard deviation of Gaussian fit component i
Θ_i	FWHM value of Gaussian fit component i
$\chi^2_{\rm red}$	Reduced chi-squared; chi-squared per degree of freedom
AICc	Corrected Akaike information criterion
F_1 score	measure of the accuracy for the decomposition of the training set (Sect. 2.1)
$\sigma_{\rm rms}$	Root-mean-square noise of the spectrum
$\sigma(T_{\rm A}^*)$	Root-mean-square noise of the spectrum given in antenna temperature values $T_{\rm A}^*$
$N_{\rm chan}$	Number of channels in a spectrum
$N_{\rm comp}$	Number of fitted Gaussian components in a spectrum
$\mathcal{S}_{\rm data}$	Significance estimate for signal peaks (Sect. 3.2.1)
$\mathcal{S}_{\rm fit}$	Significance estimate for fitted Gaussian components (Sect. 3.2.1)
$\mathcal{F}_{\rm tot}$	Sum of the $\mathcal{F}_{\rm blended}$, $\mathcal{F}_{\rm neg.\,res.\,peak}$, $\mathcal{F}_{\Theta}$, $\mathcal{F}_{\rm residual}$, and $\mathcal{F}_{N_{\rm comp}}$ flags
$\mathcal{W}$	Weight threshold

Table F.2. GaussPy+ keywords mentioned throughout the text.

Symbol	Description	GaussPy+ keyword	Default
	Essential parameters		
α_1	First smoothing parameter used in GaussPy decomposition (Sect. 2.1)	`alpha1`	`None`
α_2	Second GaussPy smoothing parameter; only used in two-phase decomposition (Sect. 2.1)	`alpha2`	`None`
$S/N_{\rm min}$	Minimum S/N for signal peaks in the data (Sect. 3.2.1)	`snr`	3
$\mathcal{S}_{\rm min}$	Minimum significance value for signal peaks and fitted Gaussian components (Sect. 3.2.1)	`significance`	5
	More advanced settings		
$\Delta\mu_{\rm max}$	Maximum difference in offset positions of Gaussian components for grouping (Sect. 3.3.1)	`mean_separation`	2 (*)
$\Delta\Theta_{\rm max}$	Maximum difference in FWHM values of Gaussian components for grouping (Sect. 3.3.1)	`fwhm_separation`	4 (*)
$\Theta_{\rm min}$	Minimum value for the FWHM of fitted Gaussian components	`min_fwhm`	1 (*)
$\Theta_{\rm max}$	Maximum value for the FWHM of fitted Gaussian components	`max_fwhm`	`None` (*)
f_a	Factor by which the maximum data value is multiplied to get a maximum limit for the fitted amplitude a_i	`max_amp_factor`	1.1
f_Θ	Factor by which the FWHM value of a fit component has to exceed all other (neighbouring) fit components to get flagged (Sect. 3.2.2)	`fwhm_factor`	2
$f_{\rm sep}$	Factor to determine the minimum required separation between two fit components before they are counted as blended (Sect. 3.2.2)	`separation_factor`	$1/\sqrt{2\ln 2}$
f_w	Factor that determines the weight given to neighbouring spectra located at a distance of 1 and 2 pixels (Sect. 3.3.2)	`weight_factor`	2

Notes. (*)Have to be specified in channel units.

Table F.2. continued.

Symbol	Description	GaussPy+ keyword	Default
$\mathcal{F}_{\text{neg. res. peak}}$	Flag criterion for negative residual features (Sect. 3.2.2)	`flag_neg_res_peak`	`True`
$\mathcal{F}_{\Theta}$	Flag criterion for broad fit components (Sect. 3.2.2)	`flag_broad`	`True`
$\mathcal{F}_{\text{blended}}$	Flag criterion for blended fit components (Sect. 3.2.2)	`flag_blended`	`True`
$\mathcal{F}_{\text{residual}}$	Flag criterion for fit results whose normalised residual values do not pass the tests for normality (Sect. 3.2.2)	`flag_residual`	`True`
$\mathcal{F}_{N_{\text{comp}}}$	Flag criterion for fit results whose number of components are not compatible with neighbouring fits (Sect. 3.2.2)	`flag_ncomps`	`True`
N_{pad}	Number of spectral channels added to the left and right of signal intervals (Sect. 3.1.2)	`pad_channels`	5 (*)
N_{min}	Minimum number of spectral channels the signal intervals in a spectrum must have (Sect. 3.1.2)	`min_channels`	100 (*)
ΔN_{max}	Maximum allowed difference in N_{comp} between fit solution and weighted median number of components determined from all immediate neighbours (Sect. 3.2.2)	`max_diff_comps`	1
ΔN_{jump}	Maximum allowed difference in N_{comp} between individual neighbouring spectra (Sect. 3.2.2)	`max_jump_comps`	2
N_{jump}	Maximum number of allowed ΔN_{jump} occurrences for a single spectrum (Sect. 3.2.2)	`n_max_jump_comps`	1
P_{Limit}	Probability threshold for features of consecutive positive or negative channels to be counted as more likely to be a noise feature (Sect. 3.1.1, Appendix A)	`p_limit`	0.02
p-value	p-value for the null hypothesis that the residual resembles a normal distribution (Sect. 3.3.1)	`min_pvalue`	0.01
$S/N_{\text{min, fit}}$	Minimum S/N ($=a_i/\sigma_{\text{rms}}$) for fitted Gaussian components (Sect. 3.2.1)	`snr_fit`	`None`
$S/N_{\text{min, neg}}$	Minimum S/N for negative peaks in the spectrum (Sect. 3.2.2)	`snr_negative`	`None`
S/N_{spike}	S/N threshold for noise spikes (Sect. 3.1.3)	`snr_noise_spike`	5
SNR_1	S/N threshold used by GaussPy for the original spectrum	`snr_thresh`	`None`
SNR_2	S/N threshold used by GaussPy for the second derivative of the smoothed spectrum	`snr2_thresh`	`None`
ξ	Minimum number of spectral channels a peak has to contain on either side (Sect. 3.1.4)	`order`	6 (*)
$\mathcal{W}_{\text{min}}$	Minimum weight threshold before phase 2 of the spatially coherent refitting routine is terminated (Sect. 3.3.2)	`min_weight`	0.5