# Lawrence Berkeley National Laboratory

**Title**
Automatic Kalman-filter-based wavelet shrinkage denoising of 1D stellar spectra

**Permalink**
https://escholarship.org/uc/item/1s18215b

**Journal**
Monthly Notices of the Royal Astronomical Society, 490(4)

**ISSN**
0035-8711

**Authors**
Gilda, Sankalp
Slepian, Zachary

**Publication Date**
2019-12-21

**DOI**
10.1093/mnras/stz2577

Peer reviewed

# Automatic Kalman-Filter-based Wavelet Shrinkage Denoising of 1D Stellar Spectra

Sankalp Gilda[1]⋆ & Zachary Slepian[1,2,3]

[1]*Department of Astronomy, University of Florida, 211 Bryant Space Science Center, Gainesville, FL 32611-2055, USA*
[2]*Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA*
[3]*Berkeley Center for Cosmological Physics, University of California, Berkeley, Berkeley, CA 94720, USA*

**ABSTRACT**
We propose a non-parametric method to denoise 1D stellar spectra based on wavelet shrinkage followed by adaptive Kalman thresholding. Wavelet shrinkage denoising involves applying the Discrete Wavelet Transform (DWT) to the input signal, 'shrinking' certain frequency components in the transform domain, and then applying inverse DWT to the reduced components. The performance of this procedure is influenced by the choice of base wavelet, the number of decomposition levels, and the thresholding function. Typically, these parameters are chosen by 'trial and error', which can be strongly dependent on the properties of the data being denoised. We here introduce an adaptive Kalman-filter-based thresholding method that eliminates the need for choosing the number of decomposition levels. We use the 'Haar' wavelet basis, which we found to be the best-suited for 1D stellar spectra. We introduce various levels of Poisson noise into synthetic PHOENIX spectra, and test the performance of several common denoising methods against our own. It proves superior in terms of noise suppression and peak shape preservation. We expect it may also be of use in automatically and accurately filtering low signal-to-noise galaxy and quasar spectra obtained from surveys such as SDSS, Gaia, LSST, PESSTO, VANDELS, LEGA-C, and DESI.

**Key words:** methods: data analysis—methods: statistical—techniques: spectroscopic—techniques: image processing.

## 1 INTRODUCTION

Most commonly-used methods for spectral denoising are parametric and thus require laborious user intervention for selection of appropriate parameters. The most popular among them is the Fourier transform (FT). Unlike other time-based techniques for noise removal such as the moving average (MA) and the exponential moving average (EMA), the FT provides a vision of a signal in the complementary domain (i.e., frequency domain, if the original signal is a time series). Because of the periodicity of the sinusoids used as the basis functions in the FT, it is well-suited to processing stationary signals. However, it has several drawbacks. It is inefficient when dealing with a non-stationary signal with varying shape and width, has zero resolution in the original (time) domain, and requires a user-specified cut-off frequency to estimate the noise in the signal. While the time-windowed FT offers an improvement by introducing a moving window that represents a compromise between time and frequency resolution, it is still limited by use of a fixed window size (Kaiser 2010).

Wavelet transform (WT), on the other hand, is able to achieve an excellent trade-off between resolutions in time and frequency domains, by providing high (low) resolution in the frequency (time) domain for small frequency values, and high (low) resolution in the time (frequency) domain for large frequency values. This provides a clear advantage over FT and short time-windowed FT for exploring features of interest in the input signal in both time and frequency domains. However, traditional denoising approaches built using wavelet transform still require trial-and-error by the user to determine and suppress the noise in the signal, as we discuss further in §2 and §4.

In this work, we present a new denoising approach, based on wavelet decomposition (Graps 1995) and adaptive Kalman-filter-based thresholding of the resultant components, which significantly improves denoising performance. It also provides a simpler, quicker implementation relative to previous methods, including FT, by being effectively non-parametric and hence independent of user input. For characterization of an object (e.g. a star, galaxy, or quasar) based on its spectrum, the precision and accuracy of the derived parameters depend strongly on two factors—the spectral resolution, and the signal-to-noise ratio (SNR). While the former is completely dependent on hardware (spectrograph), the latter can be controlled to some extent with software (algorithms). This dependence of object characterization on spectral SNR becomes even more prominent when dealing with low SNR spec-

⋆ Contact e-mail: s.gilda@ufl.edu

tra, as is common in large galaxy surveys. Our proposed method recursively decomposes the input stellar absorption spectrum—corrupted with Poisson (shot) noise—into high-frequency and low-frequency signals via a method called the wavelet transform. These high-frequency signals are then denoised (i.e. filtered) using Kalman filters. Kalman filters use a heuristic to derive noise statistics, and by employing a predictor-corrector scheme iteratively filter the signals. The denoised high-frequency components are then re-combined with the low-frequency signal at the last level, and an inverse wavelet transform used to recover the denoised stellar spectrum. We demonstrate that this method outperforms the conventional signal filtering methods when optimal method parameters are unavailable for the latter. This is indeed likely to be the case in practice due to an absence of a realistic 'reference' signal and noise to employ for obtaining optimal parameters for the conventional, parametric methods.

This paper is organized as follows. In §2 we discuss the wavelet transform and the wavelet shrinkage method. In §3 we briefly describe the Kalman filter. §4 describes our choice of wavelets and selection of decomposition levels to denoise, and demonstrates the effectiveness of our proposed method in addressing these issues. In §5, we introduce the synthetic PHOENIX spectra (Husser et al. 2013) used for comparing various filtering methods, and provide an in–depth explanation of our proposed algorithm. In §6 we use the PHOENIX data to compare the performance of our proposed method with commonly-used filtering methods from the literature. Finally, in §7 we comment on our findings and briefly discuss future extensions to the work.

## 2   WAVELET-BASED SIGNAL DENOISING

### 2.1   Discrete Wavelet Transform

The wavelet transform is similar to the Fourier transform (and more so to the windowed Fourier transform) in that both transform the input data into a different domain. However, the main difference between them is this—the Fourier transform decomposes the signal into sines and cosines, i.e. functions localized in Fourier space; in contrast the wavelet transform uses functions (*wavelets*) that are somewhat localized in both real and Fourier space. The sine and cosine functions used in the Fourier transform are 'infinite'—in that they never go to zero in the time domain—and hence the signal is deconstructed into waves that are infinitely long. This introduces the 'resolution problem'—if we have a high resolution in the frequency domain (i.e. focusing on one frequency in the signal) it is hard to isolate it in time, as each frequency exists across all time. Being uncertain of the time when focusing on the frequency is the flip-side of being uncertain of the frequency when focusing on time.

To overcome this resolution problem a wavelet transform is used to deconstruct the signal. The time-limited quality of wavelets is useful as it provides more resolution in the time domain. Instead of modeling with a wave that has infinite support, one uses a compactly-supported wave convolved with the input signal. To handle different frequencies, the wavelet transform employs varying *scales* for the wavelet. Wavelets in the WT are akin to sinusoidal waves in the FT: just as a sinusoid gets compressed (stretched) for high (low) frequencies, so do wavelets. For the WT, the signal is deconstructed using the same wavelet with different scales, rather than the same sinusoid at different frequencies. As there are hundreds of different wavelets, hundreds of different transforms are possible, each with an associated domain. However each domain
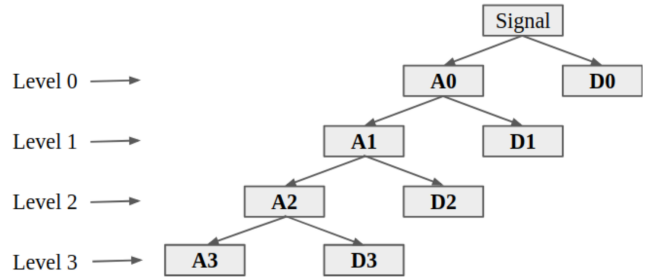


**Figure 1.** Block diagram showing the Discrete Wavelet Transform. Low pass and high pass filters are applied to the signal at each level, outputting respectively Approximation components **Ak** and Detail components **Dk**.

has 'scale' on the horizontal axis rather than 'frequency', and provides better resolution in the time domain by being finite.

In this work, we use a specific type of wavelet transform—the Discrete Wavelet Transform (DWT) (Bultheel et al. 1995 gives an in-depth review of wavelet transforms and their properties). DWT is an implementation of the wavelet transform using a discrete set of wavelet scales and translations that obey several rules. DWT is particularly suitable for signal compression and denoising. There are several implementations of the DWT algorithm. The oldest and best-known one—and the one used in this work—is the Mallat (1989) (pyramidal) algorithm. With this algorithm, the data length is restricted to a power of 2, requiring padding of the signal vector before applying the DWT. The noisy signal vector is simultaneously passed through a low-pass filter to obtain large scales (analogous to low frequencies in an FT), and a high-pass filter to obtain small scales (analogous to high frequencies in an FT). From the low-pass filter, we get a vector of approximation coefficients **A1** that represents an estimation of the original signal with half resolution. From the high-pass filter, we obtain a vector of detail coefficients **D1** that contains the details of the signal. The vector **A1** can be further decomposed to form a new vector of approximation coefficients **A2** and a new vector of detail coefficients **D2**.

With increasing decomposition level, less information will be included in the approximation coefficients. The lost information between approximation coefficients of two successive decompositions is encoded in the detail coefficients. This process can be iterated to level $\log_2 N$, where the length of the original input is $N$. We must have $N$ a power of 2 because the DWT is a recursive splitting. As a result, a vector of approximation coefficients and a series of vectors of detail coefficients are obtained that forms the DWT coefficients. The signal can be reconstructed by inverse DWT. Fig. 1 presents a block diagram demonstrating this concept. Fig. 2 demonstrates the DWT process when using a 'Haar' wavelet on a sawtooth signal.

### 2.2   Shrinkage

DWT can be used for easy and fast denoising of a noisy signal. If we take only a limited number of highest approximation coefficients of the discrete wavelet transform spectrum and perform an inverse transform, we can obtain a partially denoised signal; this process is known as *wavelet shrinkage*. A more nuanced approached involves *thresholding* the values of the approximate coefficients that lie above a certain level (the *threshold*), instead of rejecting coefficients entirely (Johnstone & Silverman 2005).

There are several different thresholding schemes in the litera-
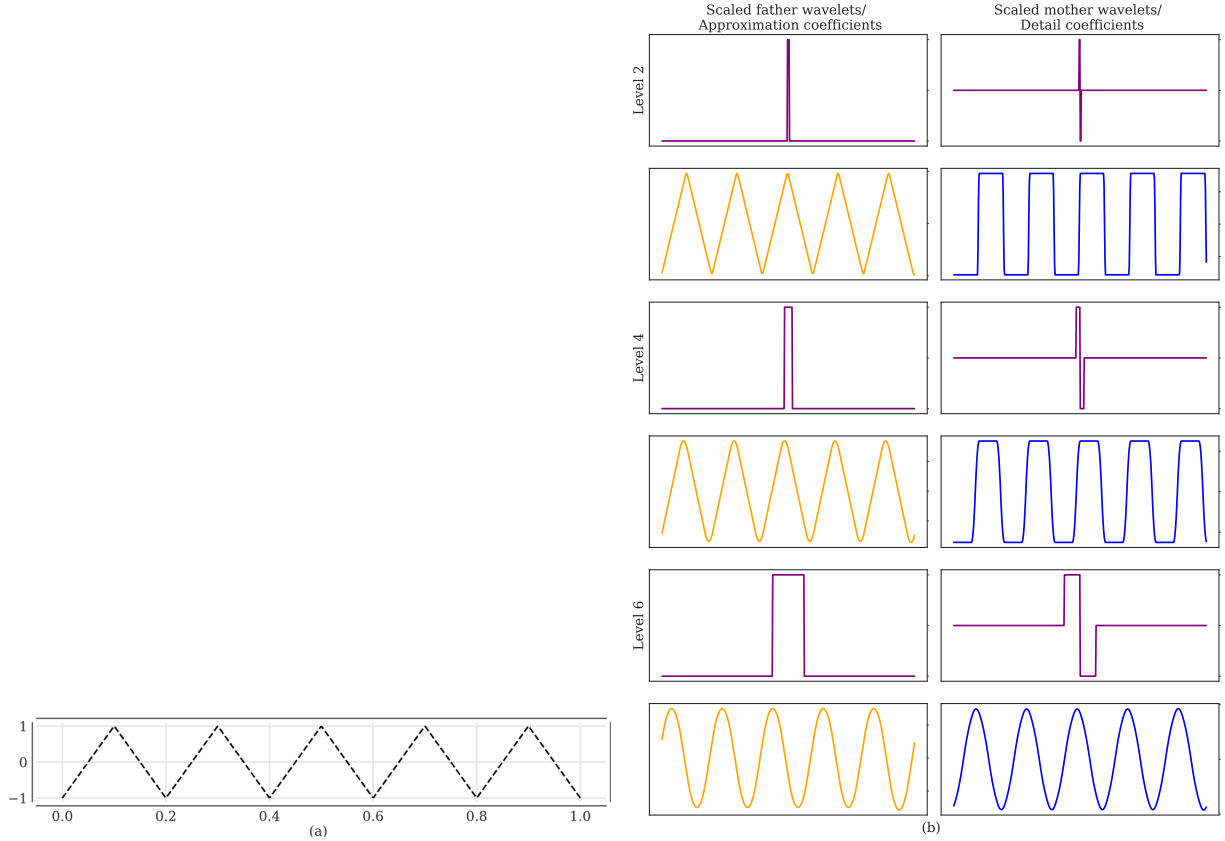
**Figure 2.** The Discrete Wavelet Transform in action, using the 'Haar' wavelet. *(a) Top Panel*—Input signal upon which DWT acts. The horizontal axis is time while the vertical axis is amplitude. There are 512 points in time; thus the maximum level of decomposition is 9 ($\log_2 512$). For illustration purposes, we have only shown the second, fourth, and sixth decomposition coefficients and wavelets. *(b)*Each six–panel column shows three DWT coefficients, along with the scaled father and mother wavelets at the relevant scale. The central tick on the vertical axis corresponds to 0. *Left Column*—Scaled father wavelets and Approximation coefficients of the input signal. *Right Column*—Scaled mother wavelets and Detail coefficients of the input signal. At each successive level (scale), the mother and father wavelets are dilated by a factor of 2, and convolved with the Approximation coefficient from the previous level. Since the father wavelet is non–negative, this is equivalent to adding successive terms of the input signal (i.e., the Approximation coefficient from the previous level) in the time domain to get the Approximation coefficient for the next level. Similarly, the action of the mother wavelet is to find differences between pairs of points for all points in the time domain; this is how we obtain the Detail coefficient for the next level. These dual actions of scaling up (equivalent to increasing the window-size of the kernel in Fourier transform) and convolving with the mother and father wavelets help DWT achieve finite resolutions in both time and frequency domains—a significant advantage over the Fourier transform.

ture, e.g. Johnstone & Silverman (2005), Li et al. (2007). The most common two are soft and hard thresholding. In soft thresholding, the $i^{th}$ component of the $j^{th}$ coefficient, $w_{j,i}$, will be set to zero if it is smaller than the threshold for that level $\lambda_j$. We note that $j$ runs from 0 up to the maximal decomposition level $k$, and we use just one threshold value at each level $j$. Otherwise, the $w_{j,i}$ are reduced by $\lambda_j$. With $\widetilde{w}_{j,i}$ denoting the thresholded coefficient, we have

$$\widetilde{w}_{j,i}^{SOFT} = \begin{cases} \text{sgn}(w_{j,i})(|w_{j,i}| - \lambda_j), & \text{for } |w_{j,i}| \geqslant \lambda_j \\ 0, & \text{for } |w_{j,i}| < \lambda_j. \end{cases} \quad (1)$$

In hard thresholding, the coefficients that are smaller than the threshold value are set to zero; those equal to it or larger are not altered. We have

$$\widetilde{w}_{j,i}^{HARD} = \begin{cases} w_{j,i}, & \text{for } |w_{j,i}| \geqslant \lambda_j \\ 0, & \text{for } |w_{j,i}| < \lambda_j. \end{cases} \quad (2)$$

The noise threshold $\lambda_j$ for a given Detail component can be obtained either using the universal level–independent threshold (Donoho & Johnstone 1994) $\lambda = \sigma^{\text{Noise}} \sqrt{2 \log N}$, with $\sigma^{\text{Noise}}$ an

estimate of the noise and $N$ the length of the input signal, or a level-dependent threshold such as in Johnstone & Silverman (1997), which sets $\lambda_j = \sigma_j^{\text{Noise}} \sqrt{2 \log_2 N_j}$ , where $N_j$ is the length of the $j^{th}$ Detail component, and $\sigma_j^{\text{Noise}}$ is an estimate of the noise level.

Hard thresholding is better suited for the case where there is a Detail wavelet coefficient is either a signal or a noise coefficient. On the other hand, soft thresholding performs better when a Detail wavelet coefficient contains both signal and noise. While there are several thresholding functions in the literature that can be used (Chang et al. 2000; Zhao & Cui 2015; Zhang & Bao 2003), in this work, we have used the universal, hard, and soft thresholding schemes to compare to our proposed algorithm. Fig. 4 displays a block diagram of the denoising process, while Fig. 3 summarizes the steps involved.

A detailed explanation and description of wavelet-based methods can be found in Graps (1995); Taswell (2000); Li et al. (2005); Tary et al. (2018).

---

**Algorithm 1** Wavelet Shrinkage Denoising

---
1: Pad the input signal with zeros until it is a power of 2.
2: Choose a basis wavelet.
3: Choose a maximal decomposition level $k$ up to which to obtain the *Detail* components, where $k \leqslant \log_2 N$, with $N$ the length of the padded discrete input signal.
4: Take the discrete wavelet transform (DWT) of $\mathbf{Z}$ up to level $k$, decomposing it into $k$ *Detail* components and the $k^{\text{th}}$ *Approximation* component.
5: Estimate the noise for the $k$ *Detail* components.
6: Calculate the noise threshold for the $k$ *Detail* components.
7: Apply noise thresholding to the $k$ *Detail* components.
8: Take the inverse discrete wavelet transform (IDWT) of the resultant $k$ *Detail* components and the $k^{\text{th}}$ *Approximation* component.
9: Un-pad the reconstructed signal.

---

**Figure 3.** Steps for the 'traditional' wavelet shrinkage denoising method. The performance is highly dependent on the number of decomposition levels of the input signal, which the user is required to choose. Too shallow a decomposition, and not all the noise is removed; too deep a decomposition, and we might end up removing parts of the signal of interest.
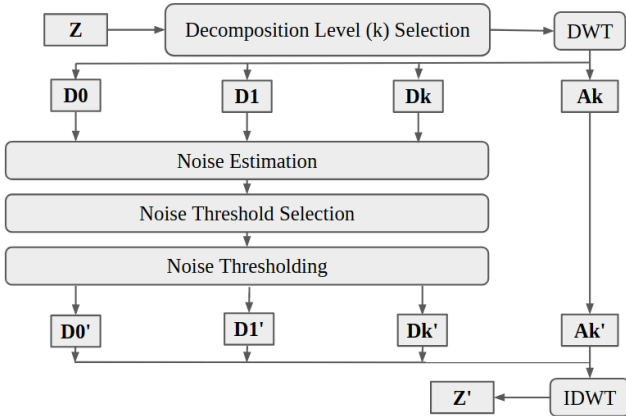


**Figure 4.** Block diagram of Wavelet Shrinkage Denoising. The input signal $\mathbf{Z}$ is decomposed into $k$ levels through the Discrete Wavelet Transform (DWT), to obtain $k$ Detail components **D0-Dk**, and one Approximation component **Ak** (second line of figure). After selecting a noise level and a thresholding scheme (third and fourth line), the latter is applied to remove the noise from the Detail components. These thresholded Detail components **D0'-Dk'**, along with the unmodified Approximation component **Ak** (all in second-to-last line of figure), are then converted back to a denoised signal **Z'** via the Inverse Discrete Wavelet Transform (IDWT).

## 3   KALMAN FILTER

The Kalman filter is a method for updating predictions in the presence of noisy measurements. Named for its inventor Rudolph E. Kalman (Kalman 1960), it uses a predictor-corrector system to recursively improve the accuracy of predictions by accounting for the noise properties of both the underlying process and the measurements. It has been extensively applied in many areas including control systems, tracking, navigation, and hydrology (Andrews & Grewal 2008; Yeh & Huang 2005). Typically, the Kalman filter (and its advanced variants, such as the Unscented Kalman filter) are

used to estimate the state of a time-varying process—such as tracking the location of an airplane using radar data and other relevant observables.

However, in this work we have used Kalman filtering to denoise stellar absorption spectra by re-interpreting the time domain as the wavelength domain. Just as with time, points in wavelength follow a fixed ordering. However, unlike with time, wavelength does not have a fixed overall direction for correlations. Whereas by causality, a given timestep of a process should correlate only with previous timesteps, the value of a spectrum's flux on a bin in wavelength is likely correlated with both lower-wavelength and higher-wavelength bins, as spectra are generally roughly predictable. Our problem is thus a more symmetric one than the usual cases where Kalman filtering is applied. To preserve this symmetry, we apply Kalman filtering in both backward and forward wavelength directions while denoising the spectra, as detailed in §5.

### 3.1   Modeling the Process

Let us denote the measured, noisy signal at wavelength bin $i$ by $\mathbf{z}_i$, and the true, unknown signal by $\mathbf{x}_i$. Let the process noise, i.e. the intrinsic noise describing the stochasticity of the system itself, independent of measurement, be $\mathbf{w}_i$. Let the measurement noise—arising due to the in-practice imperfect process of measurement—be denoted by $\mathbf{v}_i$.

The Kalman filter implementation used in this paper assumes $\mathbf{w}_i$ and $\mathbf{v}_i$ are normally-distributed random variables with independent respective covariance matrices $\mathbf{Q}_i$ and $\mathbf{R}_i$. Since the Kalman filter is a recursive filter, the true (clean) signal at wavelength $i$ depends on the signal at wavelength $i - 1$; let the evolution operator from $i - 1$ to $i$ be denoted by $\mathbf{A}_i$. Finally, let the operator relating the measured signal $\mathbf{z}_i$ to the true signal $\mathbf{x}_i$ be denoted by $\mathbf{H}_i$. Then for any wavelength point $i$ the true signal ('state', hereafter) and the measurement signal satisfy the process equation (3) and the measurement equation (4), respectively:

$$\mathbf{x}_i = \mathbf{A}_i \mathbf{x}_{i-1} + \mathbf{w}_i \tag{3}$$
$$\mathbf{z}_i = \mathbf{H}_i \mathbf{x}_i + \mathbf{v}_i. \tag{4}$$

Now that we have introduced the main elements of the Kalman filter, we proceed to briefly describe the set of recursive operations that estimate the clean signal at each wavelength $i$. This is achieved in two steps (Welch & Bishop 1995): 'prediction' and 'update', as detailed in §3.2 and §3.3. We outline the Kalman filtering process assuming we are moving in the direction of increasing wavelength; however as discussed earlier (§3), we also apply our process backwards. The appropriate equations can simply be obtained by switching $i - 1$ and $i$.

### 3.2   Predicting the Process

In the 'predict' phase, the Kalman filter uses information from the previous state to estimate the *a priori* current state. This estimate is then updated using information from a measurement to produce an *a posteriori* state estimate: this is termed the 'update' phase.

In more detail, first we initialize a mean ($\hat{\mathbf{x}}_{-1}$) and a covariance matrix ($\mathbf{P}_{-1}$) for the state of the system, which are advanced to obtain an *a priori* estimate of the state (mean $\hat{\mathbf{x}}_0$ and covariance $\mathbf{P}_0$). The covariance part is necessary to handle the uncertainty in the estimate arising due to the failure of the linear model used for the system. This is the 'prediction' part of the Kalman filtering process—where we use the estimate of the system state from the

previous run of the process to predict the *a priori* for the next run; i.e., use the flux predicted for wavelength $i$-1 to get an *a priori* estimate for the flux at wavelength $i$. This is encoded in equations (5) and (6):

$$\hat{\mathbf{x}}_i^- = \mathbf{A}_i\hat{\mathbf{x}}_{i-1} \tag{5}$$

$$\mathbf{P}_i^- = \mathbf{A}_i\mathbf{P}_{i-1}\mathbf{A}_i^T + \mathbf{Q}_i. \tag{6}$$

### 3.3 Updating the Process

Next, we use the measured signal at wavelength $i$ ($\mathbf{z}_0$, since $i = 0$ in the first iteration) to update the *a priori* state prediction from the previous step (i.e., $\hat{\mathbf{x}}_0^-$ and $\mathbf{P}_0^-$). This measurement comes from an instrument (a spectrograph in our case); its uncertainty is the measurement noise ($\mathbf{v}_i$ at wavelength $i$, with assumed mean 0 and covariance matrix $\mathbf{R}_i$). This is induced by inaccuracies in the measurement process (e.g. an imperfect CCD chip or an unstable spectrograph). In addition, we class within this measurement error any errors arising due to imperfect modeling of the system from our assumption that its evolution is linear; we highlight in §5 our rationale for this choice. This phase is known as the 'correction' or the 'update' phase; it is encoded in equations (7), (8) and (9):

$$\mathbf{K}_i = \mathbf{P}_i^-\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i^-\mathbf{H}_i^T + \mathbf{R}_i)^{-1}, \tag{7}$$

$$\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_i^- + \mathbf{K}_i(\mathbf{z}_i - \mathbf{H}_i\hat{\mathbf{x}}_i^-), \tag{8}$$

$$\mathbf{P}_i = (\mathbf{I} - \mathbf{K}_i\mathbf{H}_i)\mathbf{P}_i^-. \tag{9}$$

In equation (7), $\mathbf{K}_i$ refers to the Kalman gain, which encodes the filter's confidence in the prediction $\hat{\mathbf{x}}_i^-$ versus in the measurement $\mathbf{z}_i$. As one might expect, the Kalman gain depends on the error covariance matrices associated with both these quantities. In the limiting case where the measurement error at a wavelength point $i$ is much smaller than the prediction error (i.e., $\mathbf{R}_i \ll \mathbf{P}_i^-$), $\mathbf{K}_i \rightarrow \mathbf{H}_i^{-1}$ and the *a posteriori* state matches the measurement as acted upon by the inverse of the measurement matrix (i.e. $\hat{\mathbf{x}}_i \rightarrow \mathbf{H}_i^{-1}\mathbf{z}_i$ and $\mathbf{P}_i \rightarrow 0$). Since the present work deals with denoising spectra as measured, which are obtained by spectrographs, both the true state $\hat{\mathbf{x}}$ and the measurement $\mathbf{z}$ refer to spectra. Hence all $\mathbf{H}s$ have been assigned the identity matrix, and the *a posteriori* state matches the measurement exactly ($\hat{\mathbf{x}}_i \rightarrow \mathbf{z}_i$). This makes intuitive sense—when we are extremely confident about a measurement, we can assume that to be the real state of the system. On the other hand, when the measurement error dominates (i.e., $\mathbf{R}_i \gg \mathbf{P}_i^-$), $\mathbf{K}_i \rightarrow 0$ and the *a posteriori* state matches the *a priori* state (i.e, $\hat{\mathbf{x}}_i \rightarrow \hat{\mathbf{x}}_i^-$ and $\mathbf{P}_i \rightarrow \mathbf{P}_i^-$). When we lack confidence in a measurement, it is prudent not to update the original *a priori* state estimate. In equation (9), $\mathbf{I}$ refers to the identity matrix.

This entire process is repeated recursively until we have traversed the length of the input signal. As noted earlier, to encode the symmetry under traversing the wavelength range forwards or backwards, we run our spectral denoising algorithm for each traversal direction.

## 4 LIMITATIONS OF CURRENT METHODS

The standard soft– and hard–thresholding functions, along with other commonly used threshold-selection schemes, suffer from a few major drawbacks that preclude their use for automated signal denoising:

(i) The choice of $k$, the number of wavelet transform decomposition levels to threshold, is data-dependent and needs to be carefully chosen by the user.

(ii) The choice of $\sigma_j^{\text{Noise}}$, the noise estimate at the $j^{\text{th}}$ decomposition level, greatly influences the noise threshold $\lambda_j$, but there is no definitive way to estimate it. Different noise estimates yield different noise thresholds (Donoho & Johnstone 1994, 1995; Johnstone & Silverman 1997; Srivastava et al. 2016).

(iii) Assuming white Gaussian noise (WGN), a single noise threshold is selected and applied to the magnitudes of both negative and positive Detail coefficients. However for WGN, the assumption of no distribution bias (i.e. a distribution symmetric about zero) may often be flawed (Srivastava et al. 2016).

(iv) The methods used result in fixed thresholds that are not adjusted. Flexibility in adjusting thresholds is extremely important to optimally threshold for signals, especially when the wavelet coefficients of weak signals are close to the maximum magnitude of noise (Srivastava et al. 2016).

The method proposed in the current work is considerably more 'non-parametric' than both the traditional wavelet denoising schemes discussed above, and traditional smoothing or filtering schemes like the Fourier transform, moving mean, etc. The only 'parameter' of consequence that our proposed algorithm requires from the user is a suitable wavelet function. The criteria for choosing this basis function are: maximization of its correlation with the signal, and minimization of its correlation with the noise. Clearly, we do not want to choose an arbitrary wavelet function. The selection and development of such data-specific basis wavelets is well-studied (e.g. Rafiee et al. 2009). In the current work, we use the simplest and computationally cheapest wavelet ('Haar').

## 5 AUTOMATIC STELLAR DENOISING

In §5.1, we describe the nature and origin of the stellar spectra used to compare the performance of the proposed method with the major denoising methods from the literature. In §5.2, we explain in detail the implementation of two versions of our proposed algorithm. We then introduce error metrics used to compare the various filtering algorithms, and end by demonstrating the importance of the non-parametric nature of our proposed algorithm.

### 5.1 Test data

We use synthetic 1D PHOENIX spectra (Husser et al. 2013) for three stars of different spectral types. *Star 1* has $T_{\text{eff}} = 4,000\text{K}$, [Fe/H] = 1.0 dex, $\log g = 2.0$ dex, and spectral type = KIII. *Star 2* has $T_{\text{eff}} = 6,000\text{K}$, [Fe/H] = 1.0 dex, $\log g = 3.5$ dex, and spectral type = GV. *Star 3* has $T_{\text{eff}} = 7,000\text{K}$, [Fe/H] = 1.0 dex, $\log g = 2.5$ dex, and spectral type = KVI. The spectra are convolved from the native PHOENIX resolution of $R = \lambda/\Delta\lambda = 500,000$ down to the three lower resolutions of $2,000$, $5,000$ and $10,000$, which are then re-sampled onto the optical wavelength interval of $5,000 - 5,495.9$Å, with a sampling of 0.1Å. The wavelength range and the sampling chosen are for illustration purposes and have no bearing on the generalizability of our proposed algorithm. Our sampling yields vectors of length $4,096 = 2^{12}$. Each of the nine spectra were disturbed by twenty-five different levels of Poisson noise, to obtain final PSNRs (peak signal–to–noise ratios) going in steps of unity from $1.0 - 25.0$. For each spectrum and PSNR, we produced 100 realizations; this point is further discussed in §6. Both the *reference*

spectra and the corresponding *noisy* spectra were normalized by the peak flux in the *reference* spectra to make them all lie on a comparable flux scale. In total we deal with $9 \times 25 \times 100 = 22,500$ unique vectors to denoise, each consisting of $4,096$ data points.

## 5.2  Adaptive Kalman-Filter-based Thresholding (AKFThresh)

As described in Section 4, denoising performance with conventional wavelet shrinkage methods is a strong function of the number of decomposition levels and the thresholding scheme. Our proposed algorithm deals with this dependence by treating the individual decomposed Detail components of the noisy input signal as separate signals to denoise, and utilizes distinct Kalman filters (see equations 5-9) for each.

In brief, our proposed thresholding scheme works as follows. With every Detail level $j$, from $0 \leqslant j \leqslant k$, we associate a Kalman filter. At this level, the noisy signal $\mathbf{Z}_{Dj}$ is the Detail component, the denoised signal is $\mathbf{X}_{Dj}$, and $\mathbf{H}_{Dj}$ is set to $\mathbf{1}$ (the identity matrix). We are thus tasked with obtaining $\mathbf{Q}_{Dj}$, $\mathbf{R}_{Dj}$, and $\mathbf{H}_{Dj}$ (see equations 5-9).

Since the $0^{\text{th}}$ level Detail component consists of the highest-frequency part of the spectral signal, we assume it to be completely noise; specifically, we set the process noise covariance vector for the $0^{\text{th}}$ Detail component $\mathbf{Q}_{D0}$ to $\mathbf{0}$, and use the algorithm suggested by Meng et al. (2016) to obtain the measurement noise covariance matrix $\mathbf{R}_{D0}$. Fig. 8) illustrates the importance of choosing $\mathbf{R}_{D0}$ in this manner. Starting from the $1^{\text{st}}$ level Detail component, all the measurement noise covariance matrices $\mathbf{R}_{Dj}$ are set equal to $\mathbf{R}_{D0}$, while the process noise covariance matrices $\mathbf{Q}_{Dj}$ are obtained by subtracting the variance of the $0^{\text{th}}$ Detail component from the co-variance of the $0^{\text{th}}$ and the $j^{\text{th}}$ Detail components. Even though we are dealing with synthetically-generated spectra in the absence of a measurement device, the measurement noise matrices $\mathbf{R}_{Dj}$ are not $\mathbf{0}$ since we assign to them all error emanating from the imperfect assumption of linearity in our Kalman filters.

At the same time, we estimate the state transition matrix $\mathbf{A}_{Dj}$ by following the Support Vector Regression (SVR) procedure outlined in Salti & Di Stefano (2013) rather than setting it to unity. It is precisely this heuristic of obtaining $\mathbf{R}_{Dj}$, $\mathbf{Q}_{Dj}$, and $\mathbf{A}_{Dj}$ for the Kalman filters at the various Detail levels that allows us to make the process effectively non-parametric, instead of finding the optimal number of decompositions as is done traditionally. The pseudo-code for the proposed method is given in Fig. 6. All the variances and covariances are obtained in a robust fashion.[1]

We use two versions of our denoising algorithm—AKFThresh v1 refers to the algorithm outlined above (and in Fig. 6), whereas AKFThresh v2 uses the denoised output vector from AKFThresh v1 as input, takes the natural logarithm of the values (after adding an offset to make them strictly positive, if required), applies the AKFThresh v1 algorithm again, and exponentiates the resulting vector (followed by removing any offset that was added). The Kalman filter assumes the process and measurement noises to be additive (see equations 5 and 6). Since Poisson noise—unlike Gaussian noise—is not strictly additive, this additional process is done to convert any residual multiplicative noise to additive form, and then remove it (Zhang et al. 2014).

We compare the denoising performance of the above two versions of the proposed algorithm with those of popular denoising algorithms in the field. These are: the moving mean,[2] the moving median,[2] exponential smoothing,[3] Fast Fourier Transform with a Gaussian kernel,[4] Wiener filter,[5] the Savitzky–Golay (SG) filter,[6] wavelet transform followed by soft thresholding, and wavelet transform followed by hard thresholding. Since these commonly used methods are all parametric, any such comparison must be preceded by the selection of an appropriate loss function to minimize, followed by a search in their respective parameter spaces to optimize this loss function. Following Köhler & Lorenz (2005) we calculate three loss functions—mean absolute error, or the $L^1$ norm, least squares error, or the $L^2$ norm, and maximum absolute error, or the $L^\infty$ norm. We also consider a fourth function, the structural similarity index measure or *SSIM*, which is a measure of the similarity between two images (Wang et al. 2004; Wang & Bovik 2009).

Each error metric probes the differences between given signals in a different way. We illustrate this with an example in Fig. 5, by highlighting the differences between the mean squared error (MSE)—a 'global' error metric—and the structural similarity index measure (SSIM)—a 'local' error metric. We assume a sinusoidal form for the 'clean' signal, and create two corrupted versions of it. The MSEs for the two versions of the 'noisy' signal (dotted blue and dashed red curves) with respect to the real signal (solid black curve) are the same, while the SSIM is significantly higher for the dotted blue curve. This is also apparent by visual inspection—the dotted blue curve, while having a larger amplitude than the original signal, is clearly more 'similar' to it, in terms of capturing the peak and the valley. In contrast, the dashed red curve, being identically zero, is devoid of any information about the original signal.

We should also optimize the parameters within each method with respect to the measure of quality we want to use. For instance, in a use case, if one wished to achieve minimum least-squares error on a given data set, clearly one would optimize one's parameters (if possible) under that error metric. We analyzed each norm to fully assess the strengths and weaknesses of the different denoising methods under each.

We define appropriately expansive parameter spaces for the different algorithms and do a thorough search to find the optimal parameters to minimize the $L^2$ norm between the denoised signal and the original reference signal. We repeat this analysis two more times, with the difference that now we look for parameters that give the tenth-lowest and the twentieth-lowest $L^2$ norms. This is done to simulate the situation where, in the absence of a reference signal, manual parameter search by trial-and-error yields sub–optimal parameters for a given denoising algorithm. We then compare the denoised signals from both versions of our proposed algorithm (AKFThresh v1 and AKFThresh v2) to the denoised signals from the *optimal* and *sub-optimal* versions of the other algorithms; full tables displaying these results are available in Appendix B (on-

---

**Figure 5.** Here we illustrate the difference between Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM) to emphasize the importance of assessing algorithm performance with respect to several different error metrics. We employed MSE in Fig. 11 and SSIM in Fig. 12. The solid black curve represents the clean signal, while the dashed red and dotted blue curves represent two noisy versions of it. The MSE values for both noisy signals with respect to the clean signal are the same (0.50), while the SSIM values clearly illustrate that the blue signal (SSIM = 0.78) is much more similar to the original signal than is the red signal (SSIM = 0.02). This is of course also evident by visual inspection.

line only[7]). For robustness, this entire procedure was repeated for 100 different realizations of the noise to create synthetic noisy signals for each of the three stars, and the results averaged. A high-level pictographic overview of our proposed method is presented in Fig. 7, where we show its denoising performance in action for the absorption spectrum for *Star 1* corrupted by Poisson noise to bring the PSNR down to 10.

Our code is written entirely in PYTHON; to implement Kalman filters and wavelet decompositions, we have used the libraries *filterpy* (Labbe 2014) and *pywavelets* (Lee et al. 2006), respectively. The complete codebase and datasets used may be found online.[8]

## 6 RESULTS

We discuss here the relative denoising performances of the methods introduced in §5.2. In the interest of saving space, we only present results for synthetic spectra of *Star 2* ($T_{\text{eff}} = 4,000$K, [Fe/H] = 1.0 dex, $\log g$ = 2.0 dex, spectral type = KIII, further details in §5.1); the results for the other two stars are similar.

Tables B1 through B9[9] present the values of the different comparative metrics, at three different noise levels, for the *optimal* and *sub-optimal* parameter cases, at a resolution of 5,000 (again, the results are qualitatively similar at the other two resolutions of 2,000 and 10,000). Each table shows the mean value and the standard deviation of the results for the 100 different runs of the respective method. The best and the worst values for each column are highlighted—the best in **boldface** and the worst in *italics*. In addition, the median parameter values used for the parametric denoising methods have been specified.

In Figs. 9, 10, 11 and 12, we present the values of the four

[7] https://github.com/astrogilda/stellar_denoising
[8] https://github.com/astrogilda/stellar_denoising
[9] https://github.com/astrogilda/stellar_denoising

**Algorithm 2** Adaptive Kalman-Thresholding-based Wavelet Shrinkage Denoising

**Input:** a vector of noisy measurements of length $N$, $\mathbf{Z} = \{z_n\}_{n=1}^N$.

**Input:** a wavelet basis.

**Output:** the denoised state-space vector of length $N$, $\hat{\mathbf{X}} = \{x_n\}_{n=1}^N$.

1: Choose $k \leqslant \log_2 N$ decomposition levels to denoise the Detail components.

2: Take discrete wavelet transform (DWT) of the discrete input signal $\mathbf{Z}$ up to level $k$.

3: Calculate the measurement noise $\mathbf{R}_{D0}$ for the 0th Detail component, as a function of wavelength $\lambda$, as follows (here the subscript $D$ denotes the Detail component as opposed to the Approximation component):

4: **for** 0th Detail component $\mathbf{Z}_{D0}$ **do**

4.0:    **Initialize:** initial state probability distribution function (PDF) $p(\mathbf{x}_{D0_1}) = 0$, initial measurement noise covariance $\mathbf{R}_{D0_1} = \mathbf{0}$, process noise covariance $\mathbf{Q}_{D0} = \mathbf{0}$ . Here the sub-subscript 1 refers to the 1st of the $N$ elements in every Detail component.

4.0:    **Output:** $\mathbf{X}_{D0}, \mathbf{R}_{D0}$.

4.1:    estimate noise covariance $\mathbf{R}_{D0}$ by following procedure outlined in Meng et al. (2016).

4.2:    flip $\mathbf{Z}_{D0}$ (i.e., reverse the order of its $N$ components), and set initial state PDF $p(\mathbf{x}_{D0_1})$ to $p(\mathbf{x}_{D0_N})$, i.e., the final state PDF obtained from the original state estimation. Re-estimate $\mathbf{R}_{D0}$ and $\mathbf{X}_{D0}$.

4.3:    finally, flip $\mathbf{Z}_{D0}$ again to bring to back to the original configuration, and set initial state PDF $p(\mathbf{x}_{D0_1})$ to $p(\mathbf{x}_{D0_N})$ from above. Re-estimate $\mathbf{R}_{D0}$ and $\mathbf{X}_{D0}$.

5: **end for**

6: Set measurement noise vectors $\mathbf{R}_{Dj}$ for the $k$ Detail components ($1 \leqslant j \leqslant k$) equal to $\mathbf{R}_{D0}$.

7: Calculate the state transition function $\mathbf{A}_{Dj}{}_{j=1}^{k-1}$, as a function of wavelength $\lambda$, for the $k-1$ Detail components, as follows:

8: **for** $j$th Detail component, $j = 1 : k$ **do**

8.0:    **Initialize:** initial state PDF $p(\mathbf{x}_{Dj_0}) = 0$, process noise covariance $\mathbf{Q}_{Dj} = \text{cov}(\mathbf{Z}_{Dj}, \mathbf{Z}_{D0}) - \text{var}(\mathbf{Z}_{D0})$, measurement noise co-variance $\mathbf{R}_{Dj} = R_{D0}$ .

8.0:    **Output:** $\mathbf{X}_{Dj}$.

8.1:    estimate state transition function $\mathbf{A}_{Dj}$ by following procedure outlined in Salti & Di Stefano (2013).

8.2:    flip $\mathbf{Z}_{Dj}$, and set initial state PDF $p(\mathbf{x}_{Dj_0})$ to $p(\mathbf{x}_{Dj_N})$, i.e., the final state PDF obtained from the original state estimation. Re-estimate $\mathbf{A}_{Dj}$ and $\mathbf{X}_{Dj}$.

8.3:    finally, flip $\mathbf{Z}_{D0}$ again to bring to back to the original configuration, and set initial state PDF $p(\mathbf{x}_{D0_0})$ to $p(\mathbf{x}_{D0_N})$ from above. Re-estimate $\mathbf{A}_{Dj}$ and $\mathbf{X}_{Dj}$.

9: **end for**

10: Take the inverse discrete wavelet transform (IDWT) of the resultant $k$ Detail components $\mathbf{X}_{Dj}{}_{j=0}^{k-1}$ and the $k$th Approximation component $\mathbf{X}_{Ak}$; this is $\hat{\mathbf{X}}$.

11: **return** $\hat{\mathbf{X}}$

**Figure 6.** Steps for the proposed adaptive Kalman-thresholding-based wavelet shrinkage denoising method. By adaptively setting the noise levels for the *Detail* components instead of using a fixed level as done in Fig. 3, the denoising performance of the process becomes only weakly dependent on the number of decompositions. This is illustrated in Figs. 16 and 17.
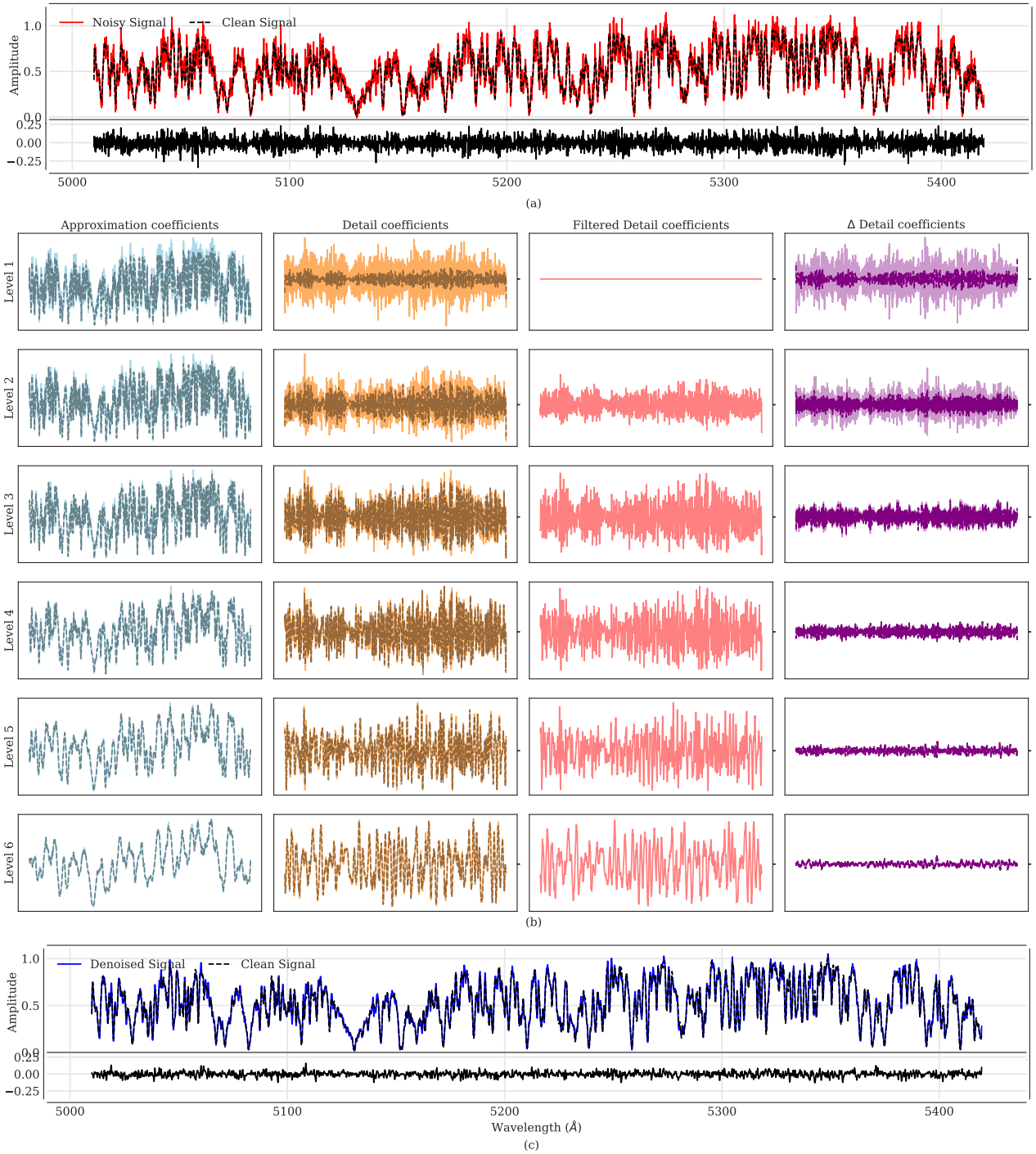
**Figure 7.** This work's proposed signal denoising algorithm in action. *(a) Top Panel, upper sub-plot*—Clean (normalized) and noisy (peak signal-to-noise ratio = 10) synthetic stellar spectra for *Star 1*. *Top Panel, lower sub-plot*—Residual between the clean and noisy signals. *(b)*—The four columns of six panels on the right show the first six Discrete Wavelet Transform coefficients and adaptive Kalman filtering of the Detail coefficients. A horizontal line through the tick–mark of each corresponds to a vertical-axis value of 0. *Leftmost Column*—Approximation coefficients for the clean (double-dashed, black) and noisy (solid, cyan) signals. *Second-from-left Column*—Detail coefficients for the clean and noisy signals; the clean Detail coefficient is in dashed brown, noisy in solid orange. The differences between the Detail coefficients for the clean and noisy signals are much larger than the differences between Approximations for clean (black, dashed) and noisy (light blue), thus justifying our decision to denoise Detail coefficients only. *Second-from-right Column*—Detail coefficients denoised by applying a Kalman filter to the Detail coefficients of the noisy signal (i.e. the orange curves in the left-neighboring column). *Rightmost Column*—Residuals obtained by subtracting the clean signal's Detail coefficients from the noisy signal's (solid lavender) and subtracting the clean signal's Detail coefficients from the denoised signal's (double-dashed purple). *(c) Top panel, upper sub-plot*—Normalized clean (dashed black) and denoised (solid blue) spectra for *Star 1*. The denoised spectrum was obtained by applying an Inverse Discrete Wavelet Transform to the 6<sup>th</sup> Approximation coefficient of the noisy signal (i.e the last row, leftmost panel in *b*) and the 6 denoised Detail coefficients (full second–from–right column in *b*).*Top panel, lower sub-plot*— Residual between the clean and denoised signals. This is plotted on the same scale as the residual in the lower panel of *(a)*, to highlight the closer match of the denoised signal to the original signal.
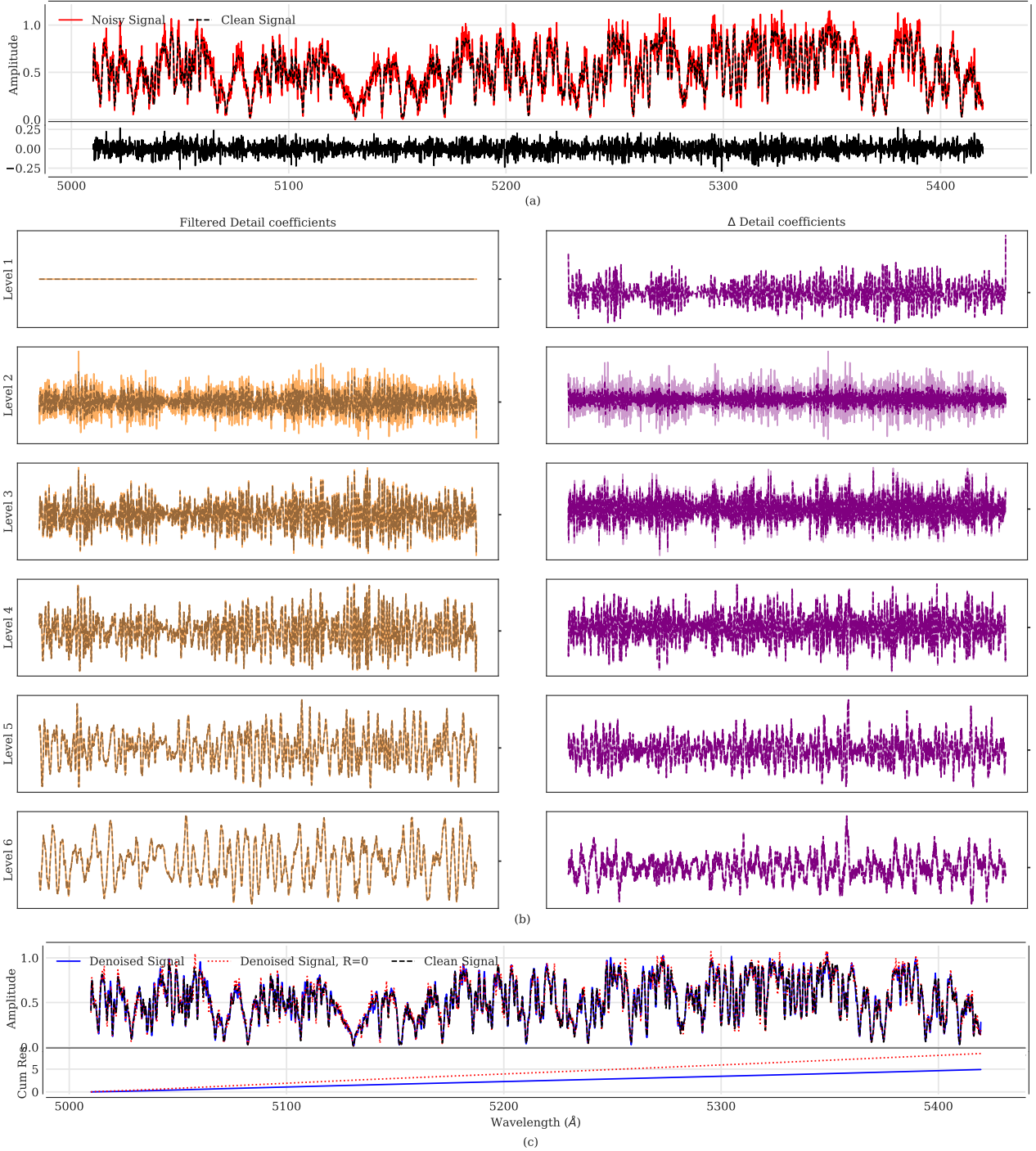
**Figure 8.** Figure illustrating the importance of using a non-zero measurement covariance matrix **R**. *(a) Top Panel*—Exactly like in *(a)* of Fig. 7. *(b)*—Each six–panel column shows the first six Discrete Wavelet Transform coefficients and adaptive Kalman filtering of the Detail coefficients. A horizontal line through the tick–mark of each corresponds to a vertical-axis value of 0. *Left Column*—Denoised Detail coefficients obtained by applying Kalman filters to the noisy counterparts. Light orange represents values with the heuristically chosen non-zero **R**, while dark orange represents decomposition with this matrix set to **0** throughout the wavelength range. *Right Column*—Residuals obtained by subtracting the Detail coefficients obtained from denoising using non-zero **R**, from the clean signal's Detail coefficients (solid lavender) and subtracting the same from the clean signal's Detail coefficients (double-dashed purple). *(c) Upper panel*—Normalized clean (dashed black), denoised with non-zero **R** (solid blue), and denoised with **R** = **0** (red) spectra for *Star 1*. *Lower panel*— Cumulative residual between the clean and the two versions of the denoised signals. It can be clearly seen that our scheme of choosing **R** by following Meng et al. (2016) instead of forcing it to **0** results in improved denoising performance.
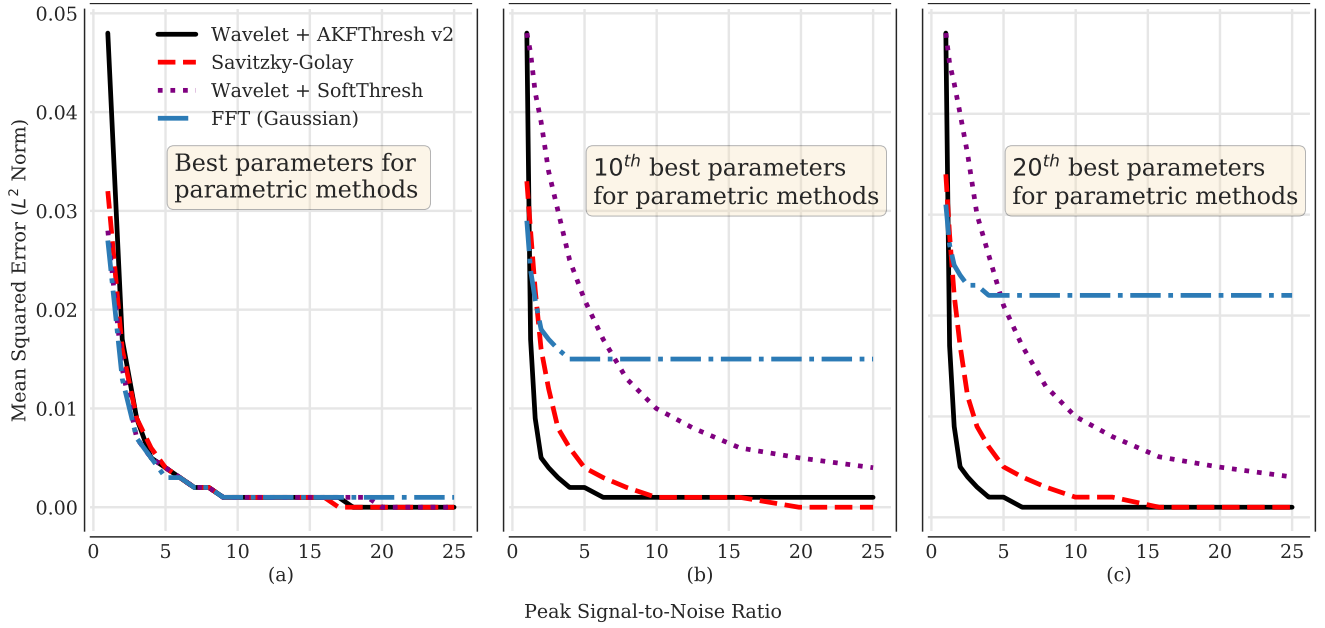
**Figure 9.** Here we show the Mean Squared Error (MSE or $L^2$ norm) as a function of the input signal Peak Signal-to-Noise Ratio (PSNR) for the four best-performing methods. For all parametric methods, the parameters have been optimized under the same metric used for this comparison, i.e. MSE. *(a)* Performance with *optimal* parameters for the parametric methods (Savitzky-Golay, Wavelet + SoftThresh, FFT with Gaussian kernel), *(b)* performance with the *tenth-best* set of parameters, and *(c)* performance with the *twentieth-best* set of parameters. In the leftmost and middle panels, our algorithm (solid black) lies essentially directly under the curve (dashed red) for Savitzky-Golay. When the parametric methods are used with their optimal parameters, they perform essentially the same as our algorithm. However, using the tenth-best parameters substantially harms their performance relative to our algorithm, with the difference from our method (solid black) getting larger with increasing PSNR for FFT (Gaussian) (solid blue), and growing and then slightly decreasing for Wavelet + SoftThresh (short-dashed purple). At tenth-best parameters, Savitzky-Golay (dashed red) remains comparable to our method, but our method performs slightly better when the twentieth-best parameters for Savitzky-Golay are used (rightmost panel).
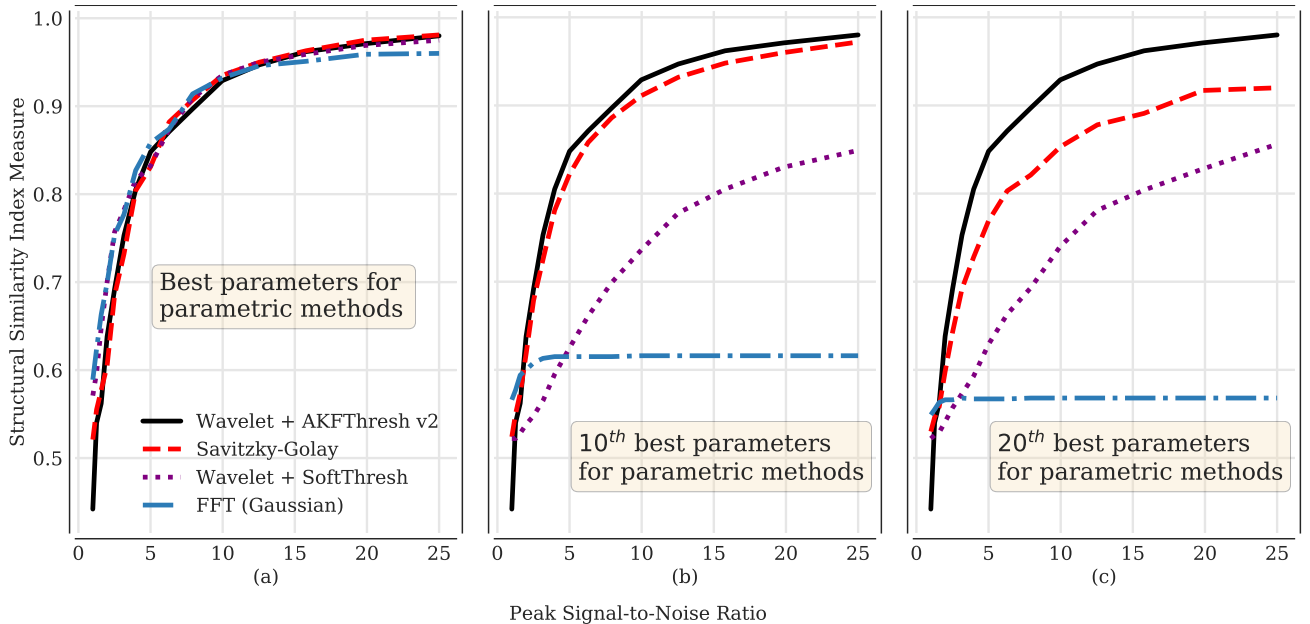


**Figure 10.** Same as Fig. 9, but with the Structural Similarity Index Measure as the error metric chosen both to optimize parameters and test performance for Savitzky-Golay, Wavelet + SoftThresh, and FFT with Gaussian kernel. Fig. 5 illustrates what SSIM vs. MSE means; higher SSIM is better. Notably, here even at tenth-best parameters our method begins to slightly outperform Savitzky-Golay, and clearly greatly outperforms the other methods. For the twentieth-best parameters, our method significantly outperforms all of the others. This observation emphasizes the importance of performing comparisons using several different error metrics.
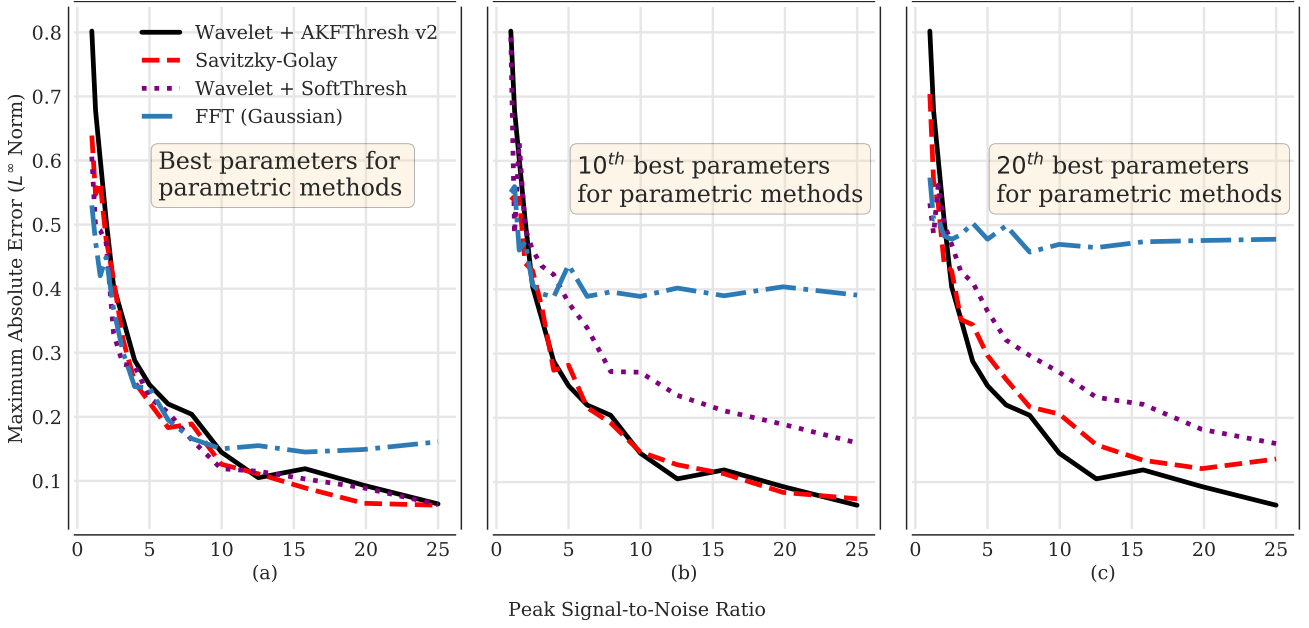
**Figure 11.** Same as Fig. 9, but with Maximum Absolute Error as the error metric. We again see that the methods are comparable (save for FFT (Gaussian)) when optimal parameters are used, but that our algorithm performs better than the others when sub-optimal parameters are used for them (rightmost panel, *c*).
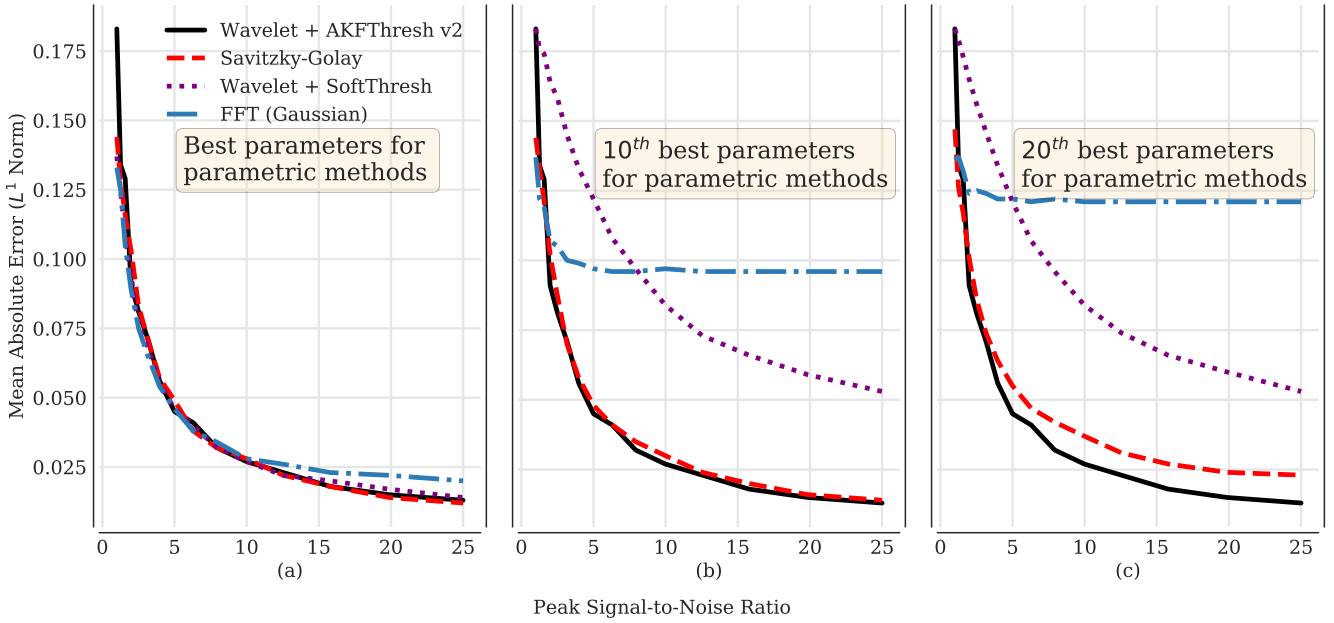


**Figure 12.** Same as Fig. 9, but with Mean Absolute Error as the error metric. This simply illustrates under another error metric the pattern shown by the previous Figures: our method outperforms the others when sub-optimal parameters are used in them.

performance metrics for the best four denoising algorithms—FFT with Gaussian kernel, Savitzky-Golay, Wavelet + Soft Thresholding, and Wavelet + AKFThresh v2—as a function of input signal PSNR and quality of parameters for the different denoising methods. The highest quality parameter is that which gives the lowest $L^2$ for a given method, the second-highest quality parameter that which gives the second lowest $L^2$, and so on. As the standard deviations are relatively small, they have been omitted for clarity.

In Figs. 16 and 17, we show the performance of both versions

of our denoising methods as a function of the decomposition level. Performance as measured by all four error metrics more or less saturates once decomposition exceeds the 6th layer. This saturation justifies our decision, and recommendation, to decompose the input noisy signal to half the maximum possible number of decomposition levels. In this work, the maximum number of levels possible is 12 because we had spectra of $4096 = 2^{12}$ wavelength bins).

Finally, Figs. 13-15 compare the denoising performance of the best version of the proposed algorithm (v2) with that of Fourier
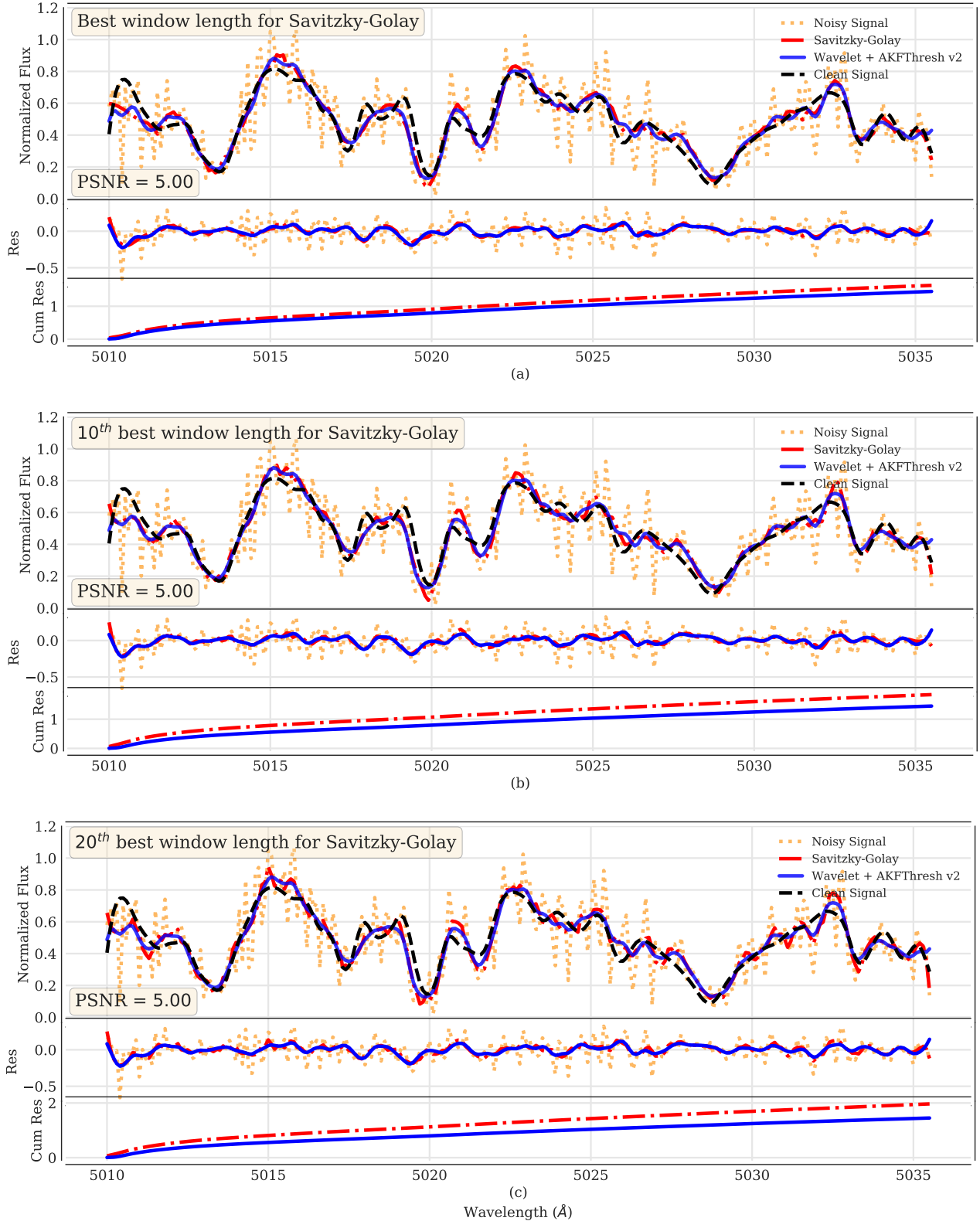
**Figure 13.** Here we show a 25Å width zoom-in, from 5010Å − 5035Å, on the original (clean), noisy, and denoised spectra for *Star 2*. PSNR = 5.0 for the noisy spectrum. *(a)* Denoised spectra using both v2 of the proposed algorithm and Savitzky-Golay (SG) with *optimal* parameters chosen by minimizing the $L^2$ norm with respect to the clean signal. *(b)* Same as *(a)* but with the *tenth-best* set of parameters for SG. *(c)* Same as *(a)* but with *twentieth-best* set of parameters for SG. When SG has the optimal window length, it just slightly underperforms our method, as can especially be seen from the cumulative residual (summed up as one moves right in wavelength—lowermost panel in *(a)*). As we move to tenth and then twentieth-best SG parameters, our method outperforms it more and more, shown by the lower cumulative residual (solid blue vs. dashed red) in the lowermost panel of each sub-figure.
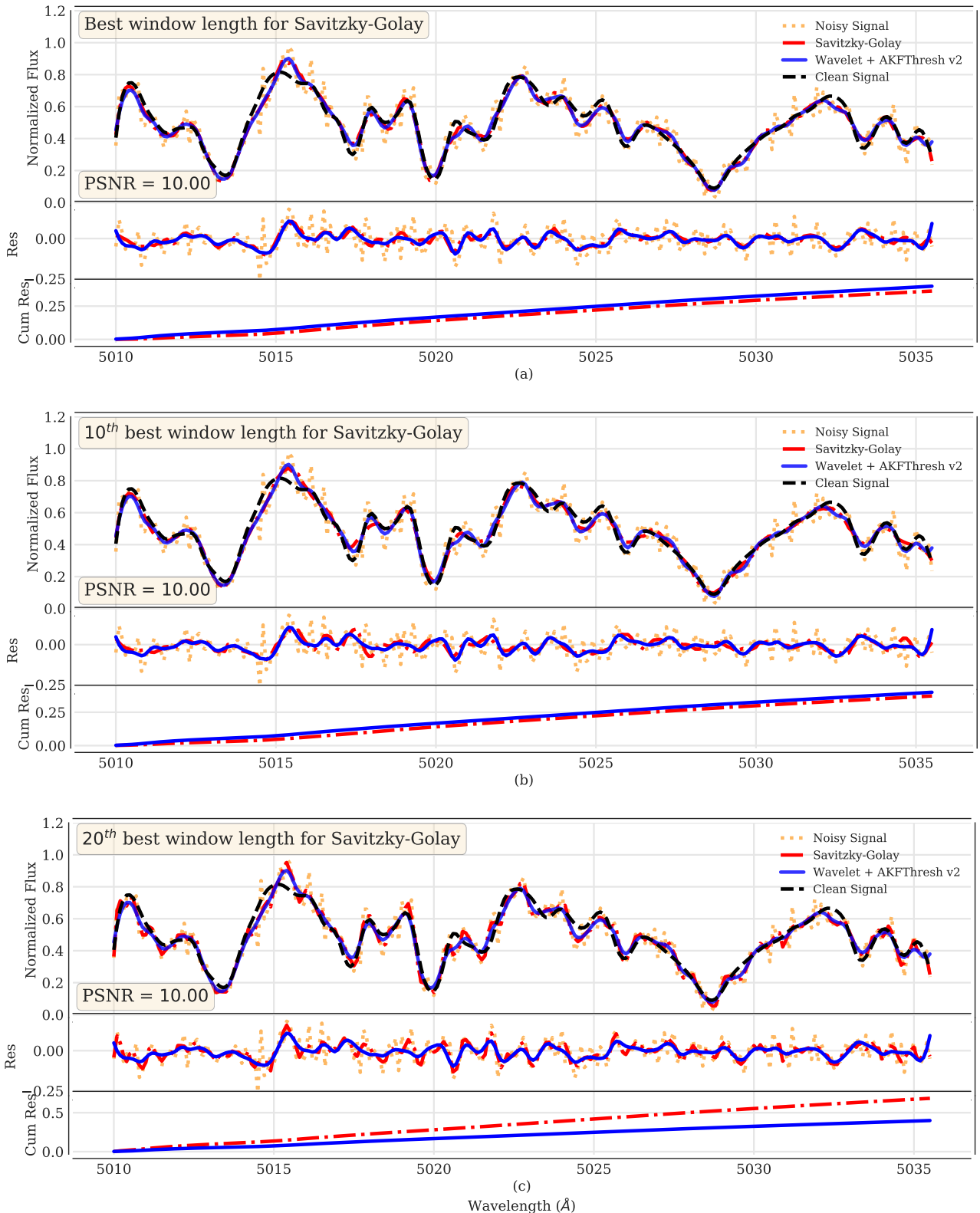
**Figure 14.** Same as Fig. 13, but with PSNR = 10.0. Here our method and SG are more comparable when SG has its best and tenth-best window lengths, but with the twentieth-best window length our method clearly outperforms SG, as shown by the lower cumulative residual (lowermost panel of sub-figure *(c)*). Interestingly, higher PSNR seems to make SG and our method more similar for tenth and twentieth-best window choices, but separates the two approaches more (to the advantage of ours) for twentieth-best window size relative to the same plots for PSNR = 5.0.
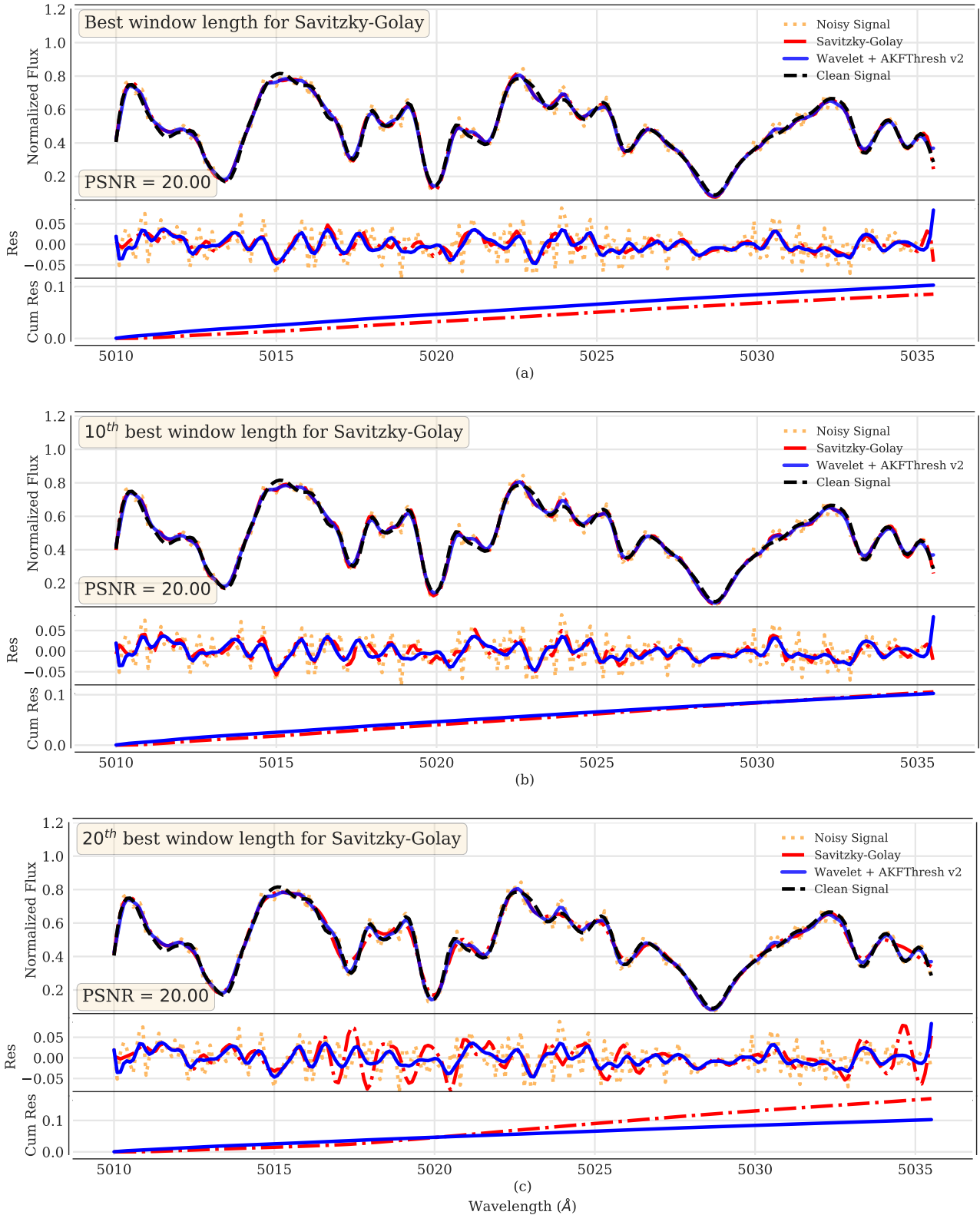
**Figure 15.** Same as Fig. 13, but with PSNR = 20.0. Here SG actually slightly outperforms our method when its best window length is used; the methods are equal at SG's tenth-best window length, and our method performs noticeably better when SG has its twentieth-best window length (lowermost panel in sub-figure *(c)*). Overall, comparing Figs. 13, 14, and this one, our method's outperformance vs. SG with twentieth-best window length is similar for PSNR of 10.0 and 20.0 and slightly smaller for PSNR of 5.0.
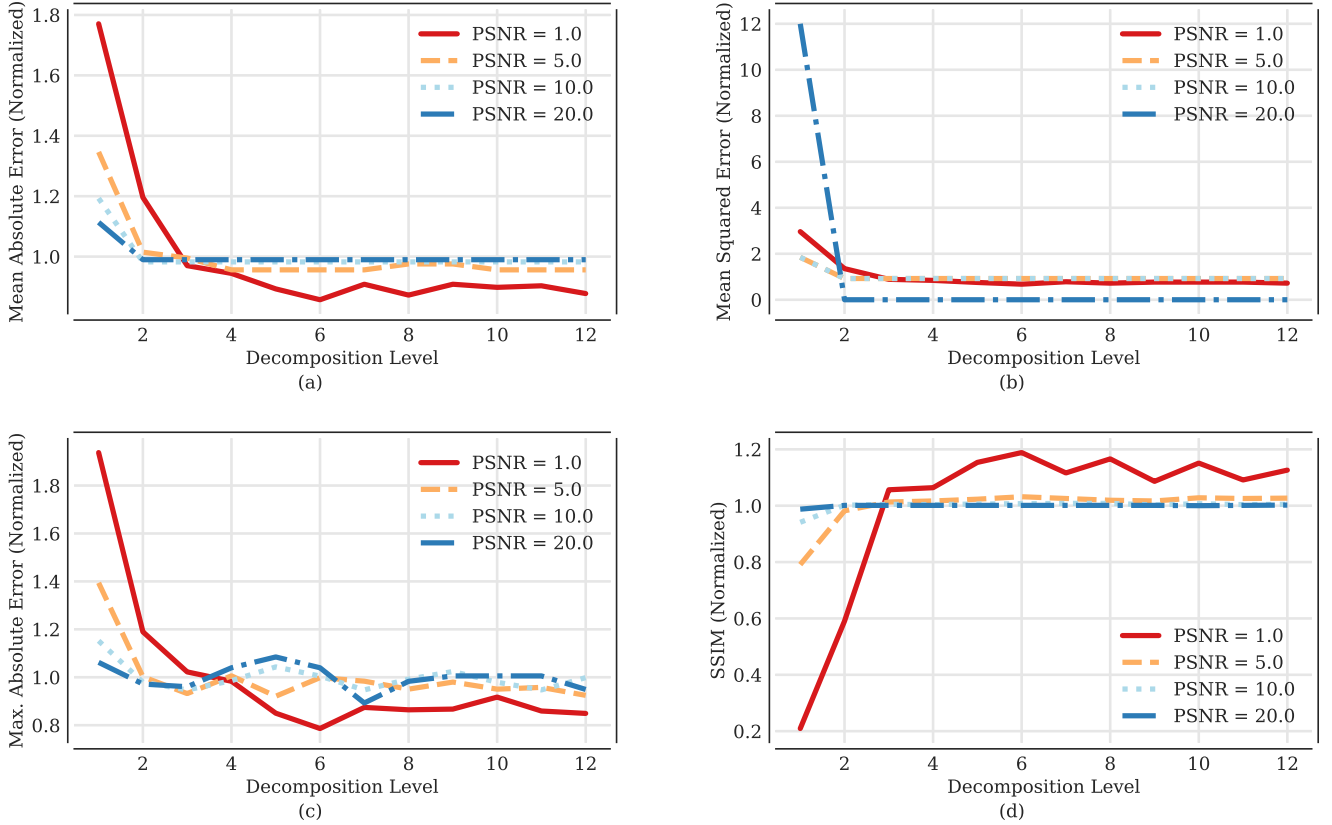
**Figure 16.** Performance of Wavelet + AKFThresh v1 as a function of the 'Haar' wavelet decomposition level under different error metrics and with different PSNRs. *(a)* Error metric = mean absolute error. *(b)* Error metric = mean squared error. *(c)* Error metric = maximum absolute error. *(d)* Error metric = structural similarity index measure. At all PSNRs and for all error metrics, the denoising performance of the proposed method saturates at a wavelet decomposition level of ≈ 6 (half the maximum possible number of decomposition levels). This justifies our choice of 6 as the level to which to decompose the noisy input signal.

transform with *optimal* and *sub-optimal* parameters. We have only shown a small section of the input spectrum to better emphasize the differences between the different spectra. At PSNRs ⩾ 5.0, our algorithm is able to extract the peaks and valleys of the original input spectrum from the noisy spectrum much better than the Fourier transform. Its superiority in filtering out the Poisson noise is especially highlighted when the width of the Gaussian kernel for the FFT is not optimal for the specific noisy spectrum. This is indeed likely to be the case when the user is responsible for picking it without access to the original spectrum.

Based on these results, we can draw several conclusions:

(i) At almost all noise levels and all choices of parameters, exponential smoothing has the worst performance.

(ii) DWT with both soft and hard thresholding gives excellent results across all four metrics for the optimal choice of parameters (i.e. maximal decomposition level, thresholding), but the performance quickly deteriorates for an incorrect choice of parameters. We emphasize that one does not know the correct parameters *a priori*, so this issue is a significant one.

(iii) At extremely low PSNRs (< 5), both versions of our algorithm provide poor denoising results, even when compared to the performance of other methods with sub-optimal parameters.

(iv) At PSNRs > 5, both versions of our algorithm start outperforming other methods, especially when sub-optimal parameters have been chosen for the other methods. This scenario is the more likely one in practice since one does not always have a refer-

ence spectrum with which to work. Indeed, calibrating a parametric denoising method on synthetic spectra before applying to real data comes with its own set of issues, primarily interpolation errors when creating the most closely-matched synthetic spectrum, as well as errors originating from instrument peculiarities' not being captured in the synthetic spectra.

(v) Savitzky–Golay is the only algorithm that compares favorably to the algorithm proposed in this paper with respect to the four error metrics we explore (Figs. 9–12). However, we note that during parameter selection for the commonly used parametric methods including Savitzky-Golay, we explored a rather wide range of parameters (discussed in the footnotes for Tables B1 through B9, e.g. for the choice of the window-length 'w'). This was computationally practical for a small number of spectral types (we considered just three different types of star in this work). However for a large survey dataset with millions to tens of millions of objects (e.g. DESI will have $\sim 10^7$ stellar spectra), with many different possible spectral types, such an exploration would likely be computationally demanding. In particular, one might, for each expected spectral type, optimize the parameters using synthetic spectra of that type. But then one would still need to build in a step where, for a given data spectrum, the correct type was automatically identified to allow the optimal parameters for that particular type for a given algorithm to be used. Having a method that simply does not require optimal parameter selection at all seems more efficient.
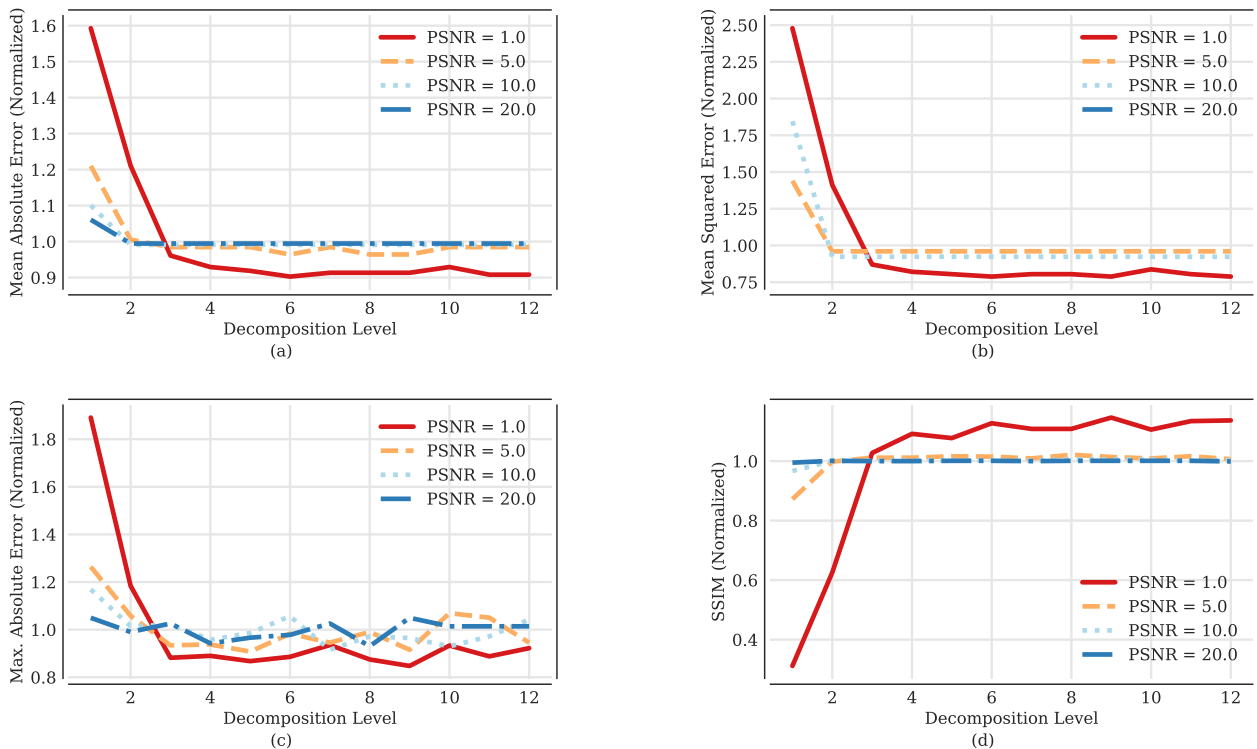
**Figure 17.** Same as Fig. 16, for Wavelet + AKFThresh v2, i.e. the second version of our algorithm that builds on the input of the first. We note that in the lower-righthand plot the light blue curve for PSNR = 10.0 lies under that for PSNR = 20.0, and in the upper righthand plot the PSNR = 20.0 curve lies directly below the PSNR = 1.0 curve. One can see that the performance of v2 is generally slightly better than that of v1 by comparing panel-by-panel to Fig. 16. Again, after 6 decomposition levels the performance roughly saturates.

## 7   CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we have proposed a new method for denoising 1D stellar spectra. Our proposed method involves two stages—a discrete wavelet transform, followed by adaptive Kalman-filter-based thresholding. By using the 'Haar' wavelet basis, the only free parameter for the user is the number of levels of decomposition for the wavelet transform. By setting this to half the maximum possible value (where the improvement of adding more levels essentially saturates), the number of free parameters can in fact be reduced to zero. Our proposed method has been validated on several types of synthetic PHOENIX spectra with Poisson noise injected. Our tests show that the method can improve denoising better than conventionally employed tools (up to 3× if the conventional methods have even slightly sub-optimal parameters; for instance as shown in Figs. 9–12). The method performs well all the way down to a PSNR of roughly 5.

Our method may also be useful denoising galaxy and quasar spectra, which often have low SNRs (< 20). There are many large current and upcoming such datasets that offer fruitful use-cases: the Sloan Digital Sky Survey (SDSS), Gaia, the Large Synoptic Survey Telescope (LSST), the Public ESO Spectroscopic Survey for Transient Objects (PESSTO), the 'deep VIMOS survey of the CANDELS UDS and CDFS fields' (VANDELS), the Large Early Galaxy Astrophysics Census survey (LEGA-C), and the Dark Energy Spectroscopic Instrument survey (DESI). DESI in particular will have low SNR, low-resolution spectra of roughly 30 million galaxies and quasars, and denoising these might aid both in redshift fitting for large-scale structure clustering analyses as well as in

ancillary science. Denoising spectra of the Lyman-$\alpha$ forest as measured using quasars as "skewers" might also enhance the science possible e.g. regarding warm dark matter constraints (for instance, Viel et al. 2013, Iršič et al. 2017) or intergalactic medium physics (e.g. Gurvich et al. 2017).

We note that although our proposed algorithm has been tested and validated on stellar absorption spectra, we believe it would perform just as well on emission spectra, which can be thought of as 'flipped' absorption spectra. In future work, we plan to evaluate the performance of AKFThresh v2 on spectra from the public data releases of the surveys listed above.

The current work is just a first step in developing better denoising algorithms based on sophisticated statistical methods, drawing on advances in the fields of digital signal processing (Oktar et al. 2016), autonomous vehicle control (Kasper & Schmidt 2008), and financial timeseries analysis (Renaud et al. 2005), among others. There is consequently significant scope for improvement and development. For instance, we are currently updating the linear Kalman filter employed here to the Unscented Kalman filter capable of modelling non-linear state-space systems (Wan & Van Der Merwe 2000; Deng et al. 2013). We are also exploring the impact of developing and using a custom mother wavelet instead of the 'Haar' wavelet. Another possible line of research would be to apply Kalman filtering to reconstructed decomposition levels, estimating the noise and process covariances via an ARMA process (Huang 2015). One might also investigate deviations from the assumption of Gaussianity for the process or measurement noise, or of the assumption of zero covariance between them (Liu et al.

2016). These ideas might enable denoising down to lower PSNRs, < 5.

In addition to wavelet shrinkage as demonstrated here, one might also explore the application of Kalman filtering in conjunction with other DWT-based methods like wavelet packet thresholding (Oktar et al. 2016; Zhao & Zhu 2017), wavelet coefficient modeling (Chen & Bui 2003; Jiang et al. 2010; Kumar & Saini 2012), and the wavelet transform modulus maxima (WTMM) method (Liu & Cheng 2007). Finally, we see no reason that this method cannot be extended to denoising astronomical time series, though this would require analysis of the data's properties to develop an appropriate basis wavelet.

## ACKNOWLEDGEMENTS

## REFERENCES

Andrews A., Grewal M., 2008, General Information. John Wiley & Sons, Ltd, doi:10.1002/9780470377819.ch1
Bultheel A., et al., 1995, Bulletin of the Belgian Mathematical Society - Simon Stevin, 2, 1
Chang S. G., Yu B., Vetterli M., 2000, IEEE Transactions on Image Processing, 9, 1532
Chen G., Bui T., 2003, IEEE Signal Processing Letters, 10, 211
Deng F., Chen J., Chen C., 2013, Journal of Systems Engineering and Electronics, 24, 655
Donoho D. L., Johnstone J. M., 1994, Biometrika, 81, 425
Donoho D. L., Johnstone I. M., 1995, Journal of the American Statistical Association, 90, 1200
Graps A., 1995, IEEE Computational Science and Engineering, 2, 50
Gurvich A., Burkhart B., Bird S., 2017, ApJ, 835, 175
Huang L., 2015, Sensors, 15, 25277
Husser T.-O., Wende-von Berg S., Dreizler S., Homeier D., Reiners A., Barman T., Hauschildt P. H., 2013, Astronomy & Astrophysics, 553, A6
Iršič V., et al., 2017, Phys. Rev. D, 96, 023522
Jiang J., Guo J., Fan W., Chen Q., 2010, in 2010 8th World Congress on Intelligent Control and Automation. pp 2894–2898, doi:10.1109/WCICA.2010.5554856
Johnstone I. M., Silverman B. W., 1997, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 59, 319
Johnstone I. M., Silverman B. W., 2005, Annals of Statistics, pp 1700–1752
Kaiser G., 2010, A Friendly Guide to Wavelets. Springer Science & Business Media
Kalman R. E., 1960, Journal of Basic Engineering, 82, 35
Kasper R., Schmidt S., 2008, in 2008 6th International Symposium on Intelligent Systems and Informatics. pp 1–5, doi:10.1109/SISY.2008.4664905
Köhler T., Lorenz D., 2005, Bremen, Germany, University of Bremen, 131
Kumar R., Saini B., 2012, International Journal of Computer Theory and Engineering, 4, 395
Labbe R., 2014, Kalman and Bayesian Filters in Python, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
Lee G., Gommers R., Wasilewski F., Wohlfahrt K., O'Leary A., Nahrstaedt H., 2006, PyWavelets - Wavelet Transforms in Python, https://github.com/PyWavelets/pywt

Li T., Ma S., Ogihara M., 2005, "Wavelet methods in data mining" in "Data Mining and Knowledge Discovery Handbook", pp 603–626
Li X., Li J., Yao X., 2007, Computers in Biology and Medicine, 37, 509
Liu Y., Cheng X., 2007, in Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007). pp 32–35, doi:10.1109/FSKD.2007.90
Liu Z., Chan S.-C., Wu H.-C., Wu J., 2016, in Signal Processing Conference (EUSIPCO), 2016 24th European. pp 443–447
Mallat S. G., 1989, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11, 674
Meng Y., Gao S., Zhong Y., Hu G., Subic A., 2016, Acta Astronautica, 120, 171
Oktar M. A., Nibouche M., Baltaci Y., 2016, in 2016 24th Signal Processing and Communication Application Conference (SIU). pp 817–820, doi:10.1109/SIU.2016.7495865
Rafiee J., Tse P. W., Harifi A., Sadeghi M. H., 2009, Expert Syst. Appl., 36, 4862
Renaud O., Starck J.-L., Murtagh F., 2005, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 35, 1241
Salti S., Di Stefano L., 2013, Image and Vision Computing, 31, 487
Srivastava M., Anderson C. L., Freed J. H., 2016, IEEE Access, 4, 3862
Tary J. B., Herrera R. H., van der Baan M., 2018, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 376, 20170254
Taswell C., 2000, Computing in Science & Engineering, 2, 12
Viel M., Becker G. D., Bolton J. S., Haehnelt M. G., 2013, Phys. Rev. D, 88, 043502
Wan E. A., Van Der Merwe R., 2000, in The IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000 (AS-SPCC). pp 153–158
Wang Z., Bovik A. C., 2009, IEEE Signal Processing Magazine, 26, 98
Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P., 2004, IEEE Transactions on Image Processing, 13, 600
Welch G., Bishop G., 1995, An Introduction to the Kalman Filter
Yeh H., Huang Y., 2005, Journal of Hydrology, 302, 28
Zhang L., Bao P., 2003, IEEE Transactions on Circuits and Systems for Video Technology, 13, 535
Zhang Y., Wang G., Xu J., Shi Z., Feng T., Dong D., Chi G., 2014, Signal Processing, 104, 401
Zhao R.-M., Cui H.-m., 2015, in 2015 7th International Conference on Modelling, Identification and Control (ICMIC). pp 1–4, doi:10.1109/ICMIC.2015.7409352
Zhao M., Zhu W.-P., 2017, in 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP). pp 71–75

## APPENDIX A: DERIVATION OF THE LINEAR KALMAN FILTER

Here we present an intuitive derivation of the linear Kalman filter used in this work. We have presented the derivation in terms of filtering a temporal signal, as this is the traditional application of a Kalman filter. However we emphasize that in this work the Kalman filter has been used to denoise spectral data in wavelength domain, leading to applying it forwards and backwards in wavelength (as discussed in §5.2). We have also used different, easier–to–understand symbols than in the main text above; the latter have been used to match the practice in signal processing literature. The transformations between the symbols used in the main text above and those used in the derivation here are specified in Table A1.

Kalman filtering answers the question of how one should optimally update predictions in the presence of noisy measurements. The prediction is smooth but likely, eventually (in a time series) to become biased; the measurements are, one hopes, unbiased (noise averages to zero), but noisy. How does one employ both?

Let us denote the true signal (also called the state vector) at

**Table A1.** Conversion between variables of the Kalman filter used in §A in the Appendix, and in §3 in the main body of this paper.

| Appendix | Main Text |
|:---:|:---:|
| $\mathbf{t}_i$ | $\mathbf{x}_i$ |
| $\mathbf{n}_i^t$ | $\mathbf{w}_i$ |
| $\hat{\mathbf{E}}_i \mathbf{t}_{i-1}$ | $\mathbf{H}_i$ |
| $\mathbf{m}_i$ | $\mathbf{z}_i$ |
| $\mathbf{n}_i^m$ | $\mathbf{v}_i$ |
| $\hat{\mathbf{M}}_i \mathbf{t}_{i-1}$ | $\mathbf{A}_i$ |
| $\mathbf{t}_i^{\text{pred}}$ | $\hat{\mathbf{x}}_i^-$ |
| $\mathbf{T}_i^{\text{pred}}$ | $\mathbf{P}_i^-$ |
| $\mathbf{T}_i^{\text{upd}}$ | $\mathbf{P}_i$ |
| $\mathbf{N}_i^t$ | $\mathbf{Q}_i$ |
| $\mathbf{t}_i^{\text{upd}}$ | $\hat{\mathbf{x}}_i$ |
| $\mathbf{W}_i$ | $\mathbf{K}_i$ |

timestep $i$ by $\mathbf{t}_i$ and the measured signal by $\mathbf{m}_i$. These are vectors, denoted by boldface. We denote the intrinsic noise, i.e. describing stochasticity of the system itself, independent of measurement, as $\mathbf{n}_i^t$, superscript $t$ for "true." We denote the measurement noise (associated with the process of measurement) $\mathbf{n}_i^m$, superscript $m$ for "measurement." We also define an evolution matrix $\hat{\mathbf{E}}_i$ that takes the true signal from time $i$ forward to time $i + 1$. The hat represents that this matrix is an operator, and operates on a state.

We also define a measurement matrix $\hat{\mathbf{M}}_i$, where the hat denotes that it is an operator (and is useful in distinguishing it later from another matrix $\mathbf{M}$ for the covariance matrix of the measurement). We also note that the hat implies the operator may not commute. However, that is not problematic—in general matrix multiplication does not commute, so we are not implying an additional property beyond what should already hold.

Finally, we make the following assumptions (the derivation and operation of the Kalman filter when these assumptions do not hold is beyond the scope of this paper; interested readers are referred to the references in §5.2):

- $\mathbf{n}_i^m$ is 0-mean white noise: $E\left(\mathbf{n}_i^m\right) = 0$, cov $\left(\mathbf{n}_i^m, \mathbf{n}_j^m\right) = \mathbf{N}_i^m \delta_{ij}^K$
- $\mathbf{n}_i^t$ is 0-mean white noise: $E\left(\mathbf{n}_i^t\right) = 0$, cov $\left(\mathbf{n}_i^t, \mathbf{n}_j^t\right) = \mathbf{N}_i^t \delta_{ij}^K$
- intrinsic noise and measurement noise are uncorrelated: cov $\left(\mathbf{n}_i^t, \mathbf{n}_i^m\right) = 0$
- true signal at timestep $i = 0$ (denoted by $\mathbf{t}_0$) is uncorrelated with the other two noise vectors. $E\left(\mathbf{t}_0\right) = \overline{t_0}$, cov $\left(\mathbf{t}_0, \mathbf{t}_0\right) = \mathbf{T}_0$.

Above, $E$ means the expectation value (also called the first moment), cov is covariance matrix, and $\delta_{ij}^K$ is the Kronecker delta ($\delta_{ij}^K = 1$ if $i = j$, 0 otherwise).

With this notation in hand we may now describe the system.

**Description of system**

$$\mathbf{t}_i = \hat{\mathbf{E}}_i \mathbf{t}_{i-1} + \mathbf{n}_i^t. \tag{A1}$$

The true signal at time (really, wavelength bin) $i$ is given by the true signal at bin $i - 1$ evolved by the evolution matrix at bin $i$ and with the intrinsic ("process") noise $\mathbf{n}_i^t$ at that time added in. Equation A1 is known as the state transition equation.

Similarly, the measurement $\mathbf{m}_i$ at bin $i$ is given by the measurement equation

$$\mathbf{m}_i = \hat{\mathbf{M}}_i \mathbf{t}_i + \mathbf{n}_i^m. \tag{A2}$$

We take the true signal at bin $i$, apply the measurement matrix $\hat{\mathbf{M}}_i$, and then add in the measurement noise at that time $\mathbf{n}_i^m$.

Note that both of the above equations are linear functions of the state (i.e., true signal), and have additive noise. This is because in the current work we have used the linear Kalman filter; for descriptions of non–linear filters such as the Extended Kalman Filter, we offer references in §5.2. We now may also describe our predictions.

**Prediction step**

Our predicted signal at bin $i$ is

$$\mathbf{t}_i^{\text{pred}} = \hat{\mathbf{E}}_i \mathbf{t}_{i-1}^{\text{pred}}. \tag{A3}$$

To predict, we simply evolve our prediction from the previous bin using the evolution matrix. We also would like the covariance matrix of the predicted true signal, which former we denote $\mathbf{T}_i^{\text{pred}}$. We have

$$\mathbf{T}_i^{\text{pred}} = \hat{\mathbf{E}}_i \mathbf{T}_{i-1}^{\text{upd}} \hat{\mathbf{E}}_i + \mathbf{N}_i^t. \tag{A4}$$

The right–hand side says that we take our updated covariance matrix from the $i - 1^{th}$ bin, evolve it (we need two evolution matrices since one can think of the true signal matrix as the outer product of two true signal vectors, each of which must be evolved by $\hat{\mathbf{E}}$), and then add the covariance matrix of the true signal noise, i.e. $\mathbf{N}^t$ is the covariance matrix of $\mathbf{n}^t$. We have now completed our prediction step. But where does the updated signal covariance matrix $\mathbf{T}_{i-1}^{\text{upd}}$ come from? How did we update this prediction from a previous one, and how will we update this prediction to move to the next one?

**Update step**

The update step is the non-trivial part of Kalman filtering: it balances the prediction with whatever measurement we have made. The updated signal is

$$\mathbf{t}_i^{\text{upd}} = \mathbf{t}_i^{\text{pred}} + \mathbf{W}_i \Delta_i^{m-p}. \tag{A5}$$

Our updated true signal ($\mathbf{t}_i^{\text{upd}}$) is the predicted one ($\mathbf{t}_i^{\text{upd}}$) plus a weight ($\mathbf{W}_i$) applied to the difference between our prediction for the measured signal and the actual signal we measured. We denote this difference as

$$\Delta_i^{m-p} \equiv \mathbf{m}_i - \hat{\mathbf{M}}_i \mathbf{t}_i^{\text{pred}}. \tag{A6}$$

The superscript $m - p$ means "measured minus predicted." The weight $\mathbf{W}_i$ is commonly referred to as the "Kalman Gain".

In other words, our actual signal measured at bin $i$ was $\mathbf{m}_i$, but our prediction for the signal we would measure was our measurement matrix operating on our prediction $\mathbf{t}_i^{\text{pred}}$ for the true signal. Equation A5 makes intuitive sense: to update, we begin with our predicted signal, and then correct it by adding in the weighted difference between our actually measured and predicted-to-be-measured signals (the weight can also take care of converting a measured signal to a true signal).

The correct form of $\mathbf{W}_i$ is the only non–obvious piece of Kalman filtering. Clearly, if we had no measurement noise and if the covariance matrix for our predicted signal were the identity matrix (i.e. all bits of the signal are independent from each other and have equal error bars), then all we would need to do would be to convert $\Delta$ from the "measured" space to the "true signal" space. In this limit, all we would need to do would be to invert our measurement matrix. We do not need anything additional to inverse–covariance–weight since we have said that our intrinsic true signal has covariance matrix as an identity matrix; inverse–covariance weighting would thus also be an identity matrix. So we have

$$\mathbf{W}_i^{\text{noiseless } m, \text{ i.e.e. sign.}} \rightarrow \hat{\mathbf{M}}_i^{-1}.$$

"i.e.e. sign." stands for "independent, equal error bars signal."

However, in practice we have measurement noise, and we also want to inverse–covariance weight the update, which is less trivial if the true signal has intrinsic covariance matrix different from the identity matrix. In this case, the predicted *measurement* covariance matrix is

$$\mathbf{M}_i^{\text{pred}} = \hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T; \tag{A7}$$

we just convert the predicted covariance matrix of the true signal into one for the measurement by applying two copies of the measurement matrix. We use two copies for the same reason as discussed earlier, regarding evolving the predicted true signal covariance matrix.

We note that the matrix $\mathbf{M}_i^{\text{pred}}$ describes the intrinsic covariance of a noiseless measurement—it occurs because our underlying true signal has some intrinsic covariance, and we are measuring that signal. However, the full covariance of the measurement will also include a contribution from the measurement noise covariance matrix, denoted $\mathbf{N}_i^m$. So our total measurement covariance matrix is

$$\mathbf{M}_i = \mathbf{M}_i^{\text{pred}} + \mathbf{N}_i^m. \tag{A8}$$

Thus our inverse covariance weight so far looks like

$$\mathbf{W}_i \sim \mathbf{M}_i^{-1} = \left(\mathbf{M}_i^{\text{pred}} + \mathbf{N}_i^m\right)^{-1} = \left(\hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T + \mathbf{N}_i^m\right)^{-1}, \tag{A9}$$

where to obtain the second equality we simply substituted equation A7 for $\mathbf{M}_i^{\text{pred}}$. However, in the limit of no measurement noise, this weight should simply reduce to the inverse of the measurement matrix. So we have

$$\mathbf{W}_i^{\text{noiseless } m} = \hat{\mathbf{M}}_i^{-1} = \mathbf{A}\left(\hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T\right)^{-1}. \tag{A10}$$

The first equality comes from demanding that the weight, in the limit of no measurement noise, should just be the inverse of the measurement matrix. We then take $\mathbf{A}$ to be an unknown matrix that multiplies our guess for the weight in the noiseless limit (with $\mathbf{N}_i^m \to 0$).

In the second equation above, $\mathbf{A}$ is an unknown matrix for which we wish to solve. We have a matrix equation of the form

$$\hat{\mathbf{M}}_i^{-1} = \mathbf{A}\mathbf{B}, \tag{A11}$$

so we apply $\mathbf{B}^{-1}$ from the right on each side. We have

$$\mathbf{B}^{-1} = \left(\hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T\right), \tag{A12}$$

meaning

$$\mathbf{A} = \hat{\mathbf{M}}_i^{-1} \left(\hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T\right). \tag{A13}$$

This simplifies to

$$\mathbf{A} = \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T. \tag{A14}$$

So our full weight is now

$$\mathbf{W}_i = \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T \left(\hat{\mathbf{M}}_i \mathbf{T}_i^{\text{pred}} \hat{\mathbf{M}}_i^T\right)^{-1}. \tag{A15}$$

We now insert the weight from equation A15 into equation A5 to update our prediction.

## APPENDIX B: TABLES

Tables are displayed on the following pages.

**Table B1.** The means and standard deviations for the error metrics for different denoising methods applied to the stellar absorption spectrum of *Star 1* corrupted with Poisson noise to get a PSNR of 5.0 (the results for the other two stars show similar trends). The methods of this paper are in the last two rows. For each denoising method, the parameters are such that they result in the lowest possible $L^1$, $L^2$, and $L^\infty$ (0 is the optimal value), and the highest-possible *SSIM* (1 is the optimal value). We generate 100 realizations of the noisy signal, denoise with the below methods, compare to the clean input signal using the four error metrics, and average the results. For a given metric, methods with the best and worst performance are highlighted—the best in **boldface** and the worst in *italics*. In case of a tie in any column (i.e. error metric), the row (denoising method) with the greatest number of best or worst solutions in the other columns is chosen and highlighted. Values are displayed to three digits after the decimal, and standard deviations are specified as fractions of their respective means. The footnotes give the values of the parameters used for the parametric denoising methods, named according to their names in the respective PYTHON implementations. 'w' refers to the length of the moving window wherever it appears (moving mean, moving median, and Savitzky-Golay), 'span' is the decay parameter for exponential smoothing, $\sigma$ is the standard deviation for the Gaussian kernel in FFT, 'mysize' is the window length for the Wiener filter, and 'k' refers to the level or scale to which the input noisy signal is decomposed in the cases of DWT with both soft– and hard–thresholding (ST and HT, respectively). See Figs. 9–12 (a) at the *x–axis* value of 5 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.113 \pm 1.5\%$ | $0.144 \pm 1.3\%$ | $0.595 \pm 10.2\%$ | $0.385 \pm 1.8\%$ |
| **Moving Mean**[i] | $0.048 \pm 2.8\%$ | $0.060 \pm 2.1\%$ | $0.246 \pm 11.9\%$ | $0.834 \pm 0.8\%$ |
| **Moving Median**[ii] | $0.055 \pm 1.8\%$ | $0.071 \pm 1.8\%$ | $0.277 \pm 5.3\%$ | $0.784 \pm 1.0\%$ |
| **Exp. Smoothing**[iii] | *$0.067 \pm 1.6\%$* | *$0.086 \pm 1.7\%$* | *$0.328 \pm 5.6\%$* | *$0.690 \pm 1.1\%$* |
| **FFT, Gaussian kernel**[iv] | **$0.047 \pm 2.1\%$** | **$0.059 \pm 2.5\%$** | **$0.219 \pm 5.1\%$** | **$0.853 \pm 0.6\%$** |
| **Wiener**[v] | $0.055 \pm 1.5\%$ | $0.073 \pm 1.4\%$ | $0.354 \pm 8.0\%$ | $0.769 \pm 0.7\%$ |
| **Savitzky-Golay**[vi] | $0.047 \pm 2.1\%$ | $0.061 \pm 2.1\%$ | $0.229 \pm 8.1\%$ | $0.841 \pm 0.5\%$ |
| **Wavelet + ST**[vii] | $0.049 \pm 2.0\%$ | $0.063 \pm 2.1\%$ | $0.246 \pm 9.4\%$ | $0.834 \pm 0.3\%$ |
| **Wavelet + HT**[viii] | $0.049 \pm 2.2\%$ | $0.064 \pm 1.7\%$ | $0.348 \pm 14.8\%$ | $0.831 \pm 0.5\%$ |
| **Wavelet + AKFThresh v1** | $0.050 \pm 1.7\%$ | $0.064 \pm 1.7\%$ | $0.256 \pm 8.6\%$ | $0.817 \pm 0.6\%$ |
| **Wavelet + AKFThresh v2** | $0.048 \pm 2.0\%$ | $0.061 \pm 2.0\%$ | $0.238 \pm 9.2\%$ | $0.838 \pm 0.8\%$ |

[i] $w = (7, 7, 8, 7)$   [ii] $w = (8, 8, 9, 8)$   [iii] span $= (4, 4, 5, 6)$   [iv] $\sigma = (2, 2, 2, 3)$   [v] mysize $= (9, 9, 16, 9)$   [vi] $w = (23, 23, 25, 23)$, polyorder $= 4$   [vii] $k = (2, 2, 3, 3)$
[viii] $k = (2, 2, 4, 3)$

**Table B2.** Same as Table B1, but using the tenth–best parameters for each denoising method, i.e. the parameters are such that they result in the tenth–to–lowest possible $L^1$, $L^2$, and $L^\infty$, and the tenth–to–highest possible *SSIM*. As can be seen, denoising as achieved by all the parametric methods gets considerably worse, while both versions of the algorithm proposed in this work (AKFThresh v1 and AKFThresh v2) remain unaffected and thus outperform the parametric methods. Specifically, **v2** of the proposed algorithm not only emerges the clear winner in terms of optimum means for the four error metrics considered, but also is the most stable based on the lowest percetage standard deviations—clearly, a very desirable property for any denoising/filtering algorithm. See Figs. 9–12 (b) at the *x–axis* value of 5 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.113 \pm 1.1\%$ | $0.145 \pm 1.1\%$ | $0.655 \pm 14.2\%$ | $0.382 \pm 1.1\%$ |
| **Moving Mean**[i] | $0.060 \pm 1.2\%$ | $0.076 \pm 1.1\%$ | $0.318 \pm 3.6\%$ | $0.726 \pm 1.1\%$ |
| **Moving Median**[ii] | $0.064 \pm 1.9\%$ | $0.082 \pm 1.5\%$ | $0.338 \pm 4.9\%$ | $0.704 \pm 1.0\%$ |
| **Exp. Smoothing**[iii] | $0.087 \pm 1.4\%$ | $0.108 \pm 1.2\%$ | $0.380 \pm 3.4\%$ | $0.654 \pm 0.5\%$ |
| **FFT, Gaussian kernel**[iv] | $0.098 \pm 0.6\%$ | $0.123 \pm 0.4\%$ | *$0.411 \pm 2.3\%$* | *$0.614 \pm 0.1\%$* |
| **Wiener**[v] | $0.062 \pm 1.1\%$ | $0.080 \pm 1.8\%$ | $0.360 \pm 5.8\%$ | $0.704 \pm 1.0\%$ |
| **Savitzky-Golay**[vi] | $0.050 \pm 2.9\%$ | $0.064 \pm 2.9\%$ | $0.249 \pm 3.8\%$ | $0.819 \pm 1.0\%$ |
| **Wavelet + ST**[vii] | *$0.121 \pm 1.5\%$* | *$0.143 \pm 1.5\%$* | $0.379 \pm 3.6\%$ | $0.623 \pm 0.8\%$ |
| **Wavelet + HT**[viii] | $0.070 \pm 2.0\%$ | $0.089 \pm 2.0\%$ | $0.383 \pm 15.6\%$ | $0.737 \pm 0.9\%$ |
| **Wavelet + AKFThresh v1** | $0.049 \pm 2.4\%$ | $0.063 \pm 2.9\%$ | $0.253 \pm 6.5\%$ | $0.815 \pm 1.2\%$ |
| **Wavelet + AKFThresh v2** | **$0.048 \pm 1.7\%$** | **$0.061 \pm 1.6\%$** | **$0.250 \pm 9.9\%$** | **$0.839 \pm 0.8\%$** |

[i] $w = (13, 8, 12, 4)$   [ii] $w = (13, 9, 14, 9)$   [iii] span $= (11, 11, 12, 13)$   [iv] $\sigma = (11, 11, 11, 11)$   [v] mysize $= (5, 15, 21, 5)$   [vi] $w = (19, 29, 25, 19)$, polyorder $= 4$
[vii] $k = (10, 10, 12, 10)$   [viii] $k = (12, 12, 12, 12)$

**Table B3.** Same as Table B2, but using the twentieth–best parameters for the error metrics, i.e., for each denoising method, the parameters are such that they result in the twentieth–to–lowest possible $L^1$, $L^2$, and $L^\infty$, and the twentieth–to–highest possible ***SSIM***. The denoising performance for the parametric methods is even worse now, and the importance of our automated algorithms becomes more evident. See Figs. 9–12 (c) at the *x–axis* value of 5 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | ***SSIM*** |
|---|---|---|---|---|
| **Noisy data** | $0.112 \pm 0.8\%$ | $0.145 \pm 0.9\%$ | $0.636 \pm 8.0\%$ | $0.384 \pm 0.9\%$ |
| **Moving Mean**[i] | $0.084 \pm 0.8\%$ | $0.107 \pm 0.6\%$ | $0.392 \pm 4.7\%$ | $0.597 \pm 0.6\%$ |
| **Moving Median**[ii] | $0.083 \pm 1.7\%$ | $0.108 \pm 1.4\%$ | $0.437 \pm 7.4\%$ | $0.594 \pm 0.6\%$ |
| **Exp. Smoothing**[iii] | $0.108 \pm 0.7\%$ | $0.135 \pm 0.5\%$ | $0.413 \pm 4.1\%$ | $0.616 \pm 0.7\%$ |
| **FFT, Gaussian kernel**[iv] | *0.122 ± 0.4%* | *0.150 ± 0.3%* | *0.478 ± 1.8%* | *0.567 ± 0.1%* |
| **Wiener**[v] | $0.077 \pm 1.2\%$ | $0.100 \pm 1.0\%$ | $0.377 \pm 5.3\%$ | $0.617 \pm 1.1\%$ |
| **Savitzky-Golay**[vi] | $0.055 \pm 2.0\%$ | $0.072 \pm 2.0\%$ | $0.298 \pm 8.6\%$ | $0.768 \pm 0.7\%$ |
| **Wavelet + ST**[vii] | $0.122 \pm 1.7\%$ | $0.145 \pm 1.5\%$ | $0.381 \pm 3.0\%$ | $0.623 \pm 0.9\%$ |
| **Wavelet + HT**[viii] | $0.070 \pm 2.7\%$ | $0.088 \pm 2.4\%$ | $0.381 \pm 11.0\%$ | $0.740 \pm 0.8\%$ |
| **Wavelet + AKFThresh v1** | $0.050 \pm 2.7\%$ | $0.064 \pm 2.4\%$ | $0.267 \pm 11.0\%$ | $0.814 \pm 0.8\%$ |
| **Wavelet + AKFThresh v2** | $\mathbf{0.048 \pm 1.5\%}$ | $\mathbf{0.061 \pm 1.6\%}$ | $\mathbf{0.251 \pm 10.0\%}$ | $\mathbf{0.837 \pm 0.9\%}$ |

[i] w = (21, 21, 22, 22)  [ii] w = (21, 21, 23, 22)  [iii] span = (21, 21, 21, 22)  [iv] $\sigma$ = (21, 21, 21, 21)  [v] mysize = (3, 23, 50, 23)  [vi] w = (15, 15, 20, 15), polyorder = 4  [vii] k = (12, 12, 12, 12)  [viii] k = (12, 12, 12, 12)

**Table B4.** Same as Table B1 (i.e. we use the optimal parameters for the parametric methods), but with PSNR of 10. See Figs. 9–12 (a) at the *x–axis* value of 10 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | ***SSIM*** |
|---|---|---|---|---|
| **Noisy data** | $0.056 \pm 1.4\%$ | $0.072 \pm 1.3\%$ | $0.324 \pm 4.3\%$ | $0.662 \pm 1.0\%$ |
| **Moving Mean**[i] | $0.028 \pm 1.6\%$ | $0.035 \pm 1.7\%$ | $0.156 \pm 7.9\%$ | $0.922 \pm 0.3\%$ |
| **Moving Median**[ii] | $0.033 \pm 2.1\%$ | $0.043 \pm 2.2\%$ | $0.185 \pm 9.8\%$ | $0.895 \pm 0.2\%$ |
| **Exp. Smoothing**[iii] | *0.042 ± 1.6%* | *0.054 ± 1.8%* | *0.225 ± 12.4%* | *0.827 ± 0.6%* |
| **FFT, Gaussian kernel**[iv] | $0.028 \pm 2.6\%$ | $0.036 \pm 2.9\%$ | $0.152 \pm 22.9\%$ | $0.932 \pm 0.3\%$ |
| **Wiener**[v] | $0.032 \pm 1.7\%$ | $0.040 \pm 1.8\%$ | $0.203 \pm 10.0\%$ | $0.901 \pm 0.4\%$ |
| **Savitzky-Golay**[vi] | $\mathbf{0.027 \pm 2.5\%}$ | $\mathbf{0.034 \pm 2.6\%}$ | $\mathbf{0.135 \pm 4.3\%}$ | $0.931 \pm 0.2\%$ |
| **Wavelet + ST**[vii] | $\mathbf{0.027 \pm 2.0\%}$ | $\mathbf{0.034 \pm 1.9\%}$ | $0.138 \pm 8.7\%$ | $\mathbf{0.933 \pm 0.3\%}$ |
| **Wavelet + HT**[viii] | $0.027 \pm 1.7\%$ | $0.036 \pm 2.0\%$ | $0.168 \pm 13.9\%$ | $0.930 \pm 0.3\%$ |
| **Wavelet + AKFThresh v1** | $0.028 \pm 2.1\%$ | $0.037 \pm 2.2\%$ | $0.159 \pm 11.1\%$ | $0.911 \pm 0.3\%$ |
| **Wavelet + AKFThresh v2** | $0.027 \pm 1.7\%$ | $0.034 \pm 1.9\%$ | $0.136 \pm 10.3\%$ | $0.926 \pm 0.3\%$ |

[i] w = (5, 5, 4, 7)  [ii] w = (6, 6, 5, 6)  [iii] span = (3, 3, 3, 4)  [iv] $\sigma$ = (2, 2, 2, 2)  [v] mysize = (5, 7, 10, 7)  [vi] w = (19, 19, 20, 19), polyorder = 4  [vii] k = (2, 2, 2, 2)  [viii] k = (2, 2, 3, 2)

**Table B5.** Same as Table B2 (i.e., tenth–best parameters for the parametric methods), but with PSNR of 10. See Figs. 9–12 (b) at the *x–axis* value of 10 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.056 \pm 1.3\%$ | $0.073 \pm 1.1\%$ | $0.330 \pm 7.6\%$ | $0.661 \pm 0.7\%$ |
| **Moving Mean**[i] | $0.044 \pm 1.0\%$ | $0.056 \pm 0.8\%$ | $0.235 \pm 7.8\%$ | $0.799 \pm 0.2\%$ |
| **Moving Median**[ii] | $0.042 \pm 1.3\%$ | $0.054 \pm 1.5\%$ | $0.254 \pm 10.3\%$ | $0.815 \pm 0.5\%$ |
| **Exp. Smoothing**[iii] | $0.081 \pm 0.5\%$ | $0.102 \pm 0.5\%$ | $0.328 \pm 3.4\%$ | $0.713 \pm 0.3\%$ |
| **FFT, Gaussian kernel**[iv] | *$0.096 \pm 0.4\%$* | *$0.121 \pm 0.2\%$* | *$0.395 \pm 1.5\%$* | *$0.616 \pm 0.1\%$* |
| **Wiener**[v] | $0.040 \pm 1.4\%$ | $0.052 \pm 0.9\%$ | $0.229 \pm 6.7\%$ | $0.820 \pm 0.4\%$ |
| **Savitzky-Golay**[vi] | $0.029 \pm 2.1\%$ | $0.037 \pm 2.3\%$ | $0.150 \pm 5.9\%$ | $0.916 \pm 0.3\%$ |
| **Wavelet + ST**[vii] | $0.084 \pm 1.1\%$ | $0.102 \pm 1.0\%$ | $0.264 \pm 3.8\%$ | $0.738 \pm 0.5\%$ |
| **Wavelet + HT**[viii] | $0.039 \pm 2.4\%$ | $0.049 \pm 2.4\%$ | $0.210 \pm 20.3\%$ | $0.862 \pm 0.4\%$ |
| **Wavelet + AKFThresh v1** | $0.028 \pm 2.0\%$ | $0.036 \pm 1.9\%$ | $0.157 \pm 9.4\%$ | $0.911 \pm 0.3\%$ |
| **Wavelet + AKFThresh v2** | $\mathbf{0.027 \pm 2.4\%}$ | $\mathbf{0.034 \pm 2.4\%}$ | $\mathbf{0.146 \pm 11.0\%}$ | $\mathbf{0.927 \pm 0.3\%}$ |

[i] w = (11, 11, 11, 12)   [ii] w = (2, 2, 11, 12)   [iii] span = (11, 11, 11, 12)   [iv] $\sigma$ = (11, 11, 11, 11)   [v] mysize = (11, 7, 13, 3)   [vi] w = (23, 23, 23, 23), polyorder = 4
[vii] k = (10, 10, 12, 10)   [viii] k = (12, 12, 12, 12)

**Table B6.** Same as Table B3 (i.e., twentieth–best parameters for the parametric methods), but with PSNR of 10. As the chosen parameters for the traditional parametric denoising methods deviate farther from their optimal values, both versions of our automatic algorithms start outperforming them by a larger and larger margin with respect to all four error metrics considered. See Figs. 9–12 (c) at the *x–axis* value of 10 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.056 \pm 0.8\%$ | $0.072 \pm 0.9\%$ | $0.313 \pm 8.7\%$ | $0.664 \pm 0.7\%$ |
| **Moving Mean**[i] | $0.081 \pm 0.4\%$ | $0.104 \pm 0.2\%$ | $0.370 \pm 5.7\%$ | $0.614 \pm 0.2\%$ |
| **Moving Median**[ii] | $0.074 \pm 0.8\%$ | $0.099 \pm 1.0\%$ | $0.430 \pm 4.2\%$ | $0.637 \pm 0.6\%$ |
| **Exp. Smoothing**[iii] | $0.107 \pm 0.5\%$ | $0.131 \pm 0.4\%$ | $0.398 \pm 2.4\%$ | $0.632 \pm 0.4\%$ |
| **FFT, Gaussian kernel**[iv] | *$0.121 \pm 0.1\%$* | *$0.149 \pm 0.1\%$* | *$0.475 \pm 1.3\%$* | *$0.568 \pm 0.0\%$* |
| **Wiener**[v] | $0.067 \pm 0.7\%$ | $0.084 \pm 0.6\%$ | $0.281 \pm 4.0\%$ | $0.686 \pm 0.5\%$ |
| **Savitzky-Golay**[vi] | $0.036 \pm 0.9\%$ | $0.047 \pm 1.2\%$ | $0.182 \pm 8.4\%$ | $0.855 \pm 0.3\%$ |
| **Wavelet + ST**[vii] | $0.084 \pm 1.6$ | $0.102 \pm 1.6$ | $0.264 \pm 3.3$ | $0.740 \pm 0.7$ |
| **Wavelet + HT**[viii] | $0.040 \pm 2.6\%$ | $0.050 \pm 2.6\%$ | $0.218 \pm 23.2\%$ | $0.861 \pm 0.6\%$ |
| **Wavelet + AKFThresh v1** | $0.028 \pm 1.5\%$ | $0.036 \pm 1.9\%$ | $0.154 \pm 7.3\%$ | $0.913 \pm 0.3\%$ |
| **Wavelet + AKFThresh v2** | $\mathbf{0.027 \pm 1.3\%}$ | $\mathbf{0.035 \pm 1.4\%}$ | $\mathbf{0.155 \pm 11.8\%}$ | $\mathbf{0.928 \pm 0.2\%}$ |

[i] w = (21, 21, 21, 21)   [ii] w = (21, 21, 21, 21)   [iii] span = (21, 21, 21, 21)   [iv] $\sigma$ = (21, 21, 21, 21)   [v] mysize = (21, 21, 21, 21)   [vi] w = (19, 29, 19, 19), polyorder = 4   [vii] k = (12, 12, 12, 12)   [viii] k = (12, 12, 12, 12)

**Table B7.** Same as Tables B1 and B4 (i.e., best parameters for the parametric methods), but with PSNR of 20. See Figs. 9–12 (a) at the *x–axis* value of 20 for a graphical illustration of this table.

|  | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.028 \pm 1.2\%$ | $0.036 \pm 1.2\%$ | $0.151 \pm 10.8\%$ | $0.869 \pm 0.3\%$ |
| **Moving Mean**[i] | $0.017 \pm 1.2\%$ | $0.021 \pm 1.2\%$ | $0.089 \pm 4.2\%$ | $0.969 \pm 0.1\%$ |
| **Moving Median**[ii] | $0.020 \pm 1.4\%$ | $0.026 \pm 0.9\%$ | $0.109 \pm 5.6\%$ | $0.947 \pm 0.2\%$ |
| **Exp. Smoothing**[iii] | *$0.024 \pm 1.0\%$* | *$0.031 \pm 0.9\%$* | $0.124 \pm 8.7\%$ | *$0.922 \pm 0.2\%$* |
| **FFT, Gaussian kernel**[iv] | $0.021 \pm 1.2\%$ | $0.027 \pm 1.2\%$ | *$0.132 \pm 14.5\%$* | $0.958 \pm 0.1\%$ |
| **Wiener**[v] | $0.017 \pm 1.5\%$ | $0.022 \pm 1.7\%$ | $0.103 \pm 9.6\%$ | $0.965 \pm 0.1\%$ |
| **Savitzky-Golay**[vi] | **$0.015 \pm 1.6\%$** | **$0.018 \pm 1.4\%$** | **$0.073 \pm 5.8\%$** | $0.974 \pm 0.1\%$ |
| **Wavelet + ST**[vii] | $0.017 \pm 1.8\%$ | $0.022 \pm 1.6\%$ | $0.091 \pm 7.3\%$ | $0.969 \pm 0.1\%$ |
| **Wavelet + HT**[viii] | $0.018 \pm 1.1\%$ | $0.023 \pm 1.3\%$ | $0.102 \pm 12.6\%$ | $0.966 \pm 0.2\%$ |
| **Wavelet + AKFThresh v1** | $0.016 \pm 2.4\%$ | $0.020 \pm 2.1\%$ | $0.083 \pm 6.9\%$ | $0.966 \pm 0.1\%$ |
| **Wavelet + AKFThresh v2** | **$0.015 \pm 1.9\%$** | **$0.019 \pm 2.1\%$** | $0.080 \pm 11.1\%$ | **$0.972 \pm 0.1\%$** |

[i] w = (3, 5, 3, 5)  [ii] w = (5, 5, 4, 5)  [iii] span = (2, 2, 2, 2)  [iv] $\sigma$ = (2, 2, 2, 2)  [v] mysize = (5, 5, 7, 5)  [vi] w = (15, 15, 17, 15), polyorder = 4  [vii] k = (1, 2, 2, 2)
[viii] k = (1, 1, 2, 2)

**Table B8.** Same as Tables B2 and B5 (i.e., tenth–best parameters for the parametric methods), but with PSNR of 20. See Figs. 9–12 (b) at the *x–axis* value of 20 for a graphical illustration of this table.

|  | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.028 \pm 1.1\%$ | $0.036 \pm 1.0\%$ | $0.152 \pm 7.9\%$ | $0.868 \pm 0.4\%$ |
| **Moving Mean**[i] | $0.041 \pm 0.6\%$ | $0.053 \pm 0.4\%$ | $0.237 \pm 5.1\%$ | $0.842 \pm 0.1\%$ |
| **Moving Median**[ii] | $0.032 \pm 1.5\%$ | $0.043 \pm 1.9\%$ | $0.270 \pm 8.7\%$ | $0.873 \pm 0.4\%$ |
| **Exp. Smoothing**[iii] | $0.080 \pm 0.5\%$ | $0.100 \pm 0.4\%$ | $0.310 \pm 1.9\%$ | $0.727 \pm 0.2\%$ |
| **FFT, Gaussian kernel**[iv] | *$0.096 \pm 0.2\%$* | *$0.121 \pm 0.1\%$* | *$0.397 \pm 1.0\%$* | *$0.616 \pm 0.0\%$* |
| **Wiener**[v] | $0.034 \pm 0.9\%$ | $0.044 \pm 1.0\%$ | $0.155 \pm 9.0\%$ | $0.866 \pm 0.2\%$ |
| **Savitzky-Golay**[vi] | $0.016 \pm 1.8\%$ | $0.021 \pm 1.7\%$ | $0.088 \pm 9.1\%$ | $0.962 \pm 0.2\%$ |
| **Wavelet + ST**[vii] | $0.059 \pm 1.3\%$ | $0.071 \pm 1.1\%$ | $0.186 \pm 2.1\%$ | $0.830 \pm 0.3\%$ |
| **Wavelet + HT**[viii] | $0.024 \pm 2.0\%$ | $0.030 \pm 1.9\%$ | $0.112 \pm 16.1\%$ | $0.932 \pm 0.2\%$ |
| **Wavelet + AKFThresh v1** | $0.016 \pm 1.4\%$ | $0.020 \pm 1.6\%$ | $0.091 \pm 9.0\%$ | $0.965 \pm 0.1\%$ |
| **Wavelet + AKFThresh v2** | **$0.015 \pm 1.1\%$** | **$0.020 \pm 1.3\%$** | **$0.079 \pm 9.7\%$** | **$0.973 \pm 0.1\%$** |

[i] w = (11, 11, 11, 11)  [ii] w = (11, 11, 11, 11)  [iii] span = (11, 11, 11, 11)  [iv] $\sigma$ = (11, 11, 11, 11)  [v] mysize = (11, 11, 11, 11)  [vi] w = (11, 11, 20, 11),
polyorder = 4  [vii] k = (10, 10, 12, 10)  [viii] k = (12, 12, 12, 12)

**Table B9.** Same as Tables B3 and B6 (i.e., twentieth–best parameters for the parametric methods), but with PSNR of noisy signal 20. As the chosen parameters for the traditional parametric signal denoising methods deviate farther from their optimal values, both versions of our automatic algorithms start outperforming them by a larger and larger margin with respect to all four error metrics considered. See Figs. 9–12 (c) at the *x–axis* value of 20 for a graphical illustration of this table.

| | $L^1$ | $L^2$ | $L^\infty$ | *SSIM* |
|---|---|---|---|---|
| **Noisy data** | $0.028 \pm 1.0\%$ | $0.001 \pm 0.9\%$ | $0.165 \pm 6.7\%$ | $0.868 \pm 0.2\%$ |
| **Moving Mean**[i] | $0.080 \pm 0.2\%$ | $0.103 \pm 0.1\%$ | $0.354 \pm 2.8\%$ | $0.615 \pm 0.1\%$ |
| **Moving Median**[ii] | $0.069 \pm 0.7\%$ | $0.097 \pm 0.6\%$ | $0.441 \pm 2.3\%$ | $0.654 \pm 0.4\%$ |
| **Exp. Smoothing**[iii] | $0.106 \pm 0.3\%$ | $0.131 \pm 0.2\%$ | $0.380 \pm 0.9\%$ | $0.637 \pm 0.2\%$ |
| **FFT, Gaussian kernel**[iv] | *0.121 ± 0.1%* | *0.149 ± 0.0%* | *0.476 ± 0.8%* | *0.568 ± 0.0%* |
| **Wiener**[v] | $0.064 \pm 0.3\%$ | $0.079 \pm 0.4\%$ | $0.255 \pm 5.6\%$ | $0.708 \pm 0.3\%$ |
| **Savitzky-Golay**[vi] | $0.024 \pm 0.4\%$ | $0.031 \pm 0.5\%$ | $0.127 \pm 7.4\%$ | $0.918 \pm 0.1\%$ |
| **Wavelet + ST**[vii] | $0.059 \pm 1.2\%$ | $0.071 \pm 1.1\%$ | $0.185 \pm 3.1\%$ | $0.832 \pm 0.3\%$ |
| **Wavelet + HT**[viii] | $0.024 \pm 2.1\%$ | $0.030 \pm 2.2\%$ | $0.118 \pm 20.4\%$ | $0.932 \pm 0.3\%$ |
| **Wavelet + AKFThresh v1** | $0.016 \pm 1.6\%$ | $\mathbf{0.020 \pm 1.5\%}$ | $\mathbf{0.082 \pm 7.2\%}$ | $0.967 \pm 0.2\%$ |
| **Wavelet + AKFThresh v2** | $\mathbf{0.015 \pm 2.1\%}$ | $\mathbf{0.020 \pm 2.0\%}$ | $0.086 \pm 10.6\%$ | $\mathbf{0.972 \pm 0.1\%}$ |

[i] w = (21, 21, 21, 21)   [ii] w = (21, 21, 21, 21)   [iii] span = (21, 21, 21, 21)   [iv] $\sigma$ = (21, 21, 21, 21)   [v] mysize = (21, 21, 21, 21)   [vi] w = (25, 25, 16, 25), polyorder = 4   [vii] k = (12, 12, 12, 12)   [viii] k = (12, 12, 12, 12)