Weather Radar Time Series Simulation: Improving Accuracy and Performance

CHRISTOPHER D. CURTIS

Cooperative Institute for Mesoscale Meteorological Studies, University of Oklahoma, and NOAA/OAR/National Severe Storms Laboratory, Norman, Oklahoma

(Manuscript received 11 December 2017, in final form 17 September 2018)

ABSTRACT

Time series simulation is an important tool for developing and testing new signal processing algorithms for weather radar. The methods for simulating time series data have not changed much over the last few decades, but recent advances in computing technology call for new methods. It would seem that faster computers would make better-performing simulators less necessary, but improved technology has made comprehensive, multiday simulations feasible. Even a relatively minor performance improvement can significantly shorten the time of one of these multiday simulations. Current simulators can also be inaccurate for some sets of parameters, especially narrow spectrum widths. In this paper, three new modifications to the conventional simulators are introduced to improve accuracy and performance. Two of the modifications use thresholds to optimize both the total number of samples and the number of random variates that need to be simulated. The third modification uses an alternative method for implementing the inverse Fourier transform. These new modifications lead to fast versions of the simulators that accurately match the desired autocorrelation and spectrum for a wide variety of signal parameters. Additional recommendations for using single-precision values and graphical processing units are also suggested.

1. Introduction

Time series simulation plays an important role in developing new weather radar signal processing algorithms. Having simulated data that accurately represent the weather signal characteristics is essential. In this paper, a few modifications to conventional simulators will be introduced to improve both accuracy and performance. The focus will be on simulating single-polarization weather radar data using a Gaussian spectral model for the weather (or ground clutter) signal. Although this model does not always hold for weather signals (Janssen and Van Der Spek 1985; Yu et al. 2009), it is the standard for weather radar simulations and is a reasonable starting point when assessing algorithm performance. The approaches described in this work do not require a Gaussian spectrum model and should be able to be applied to other models with some modification. Although the focus of this paper is on single-polarization time series data, two realizations of single-polarization time series data can always be combined using an approach such as the one found in Galati and Pavan (1995).

These time series simulations serve a very specific purpose for radar engineers who are developing new weather radar variable estimators and testing algorithms. A realization of simulated time series data per channel represents the radar return from a particular resolution volume. The number of samples in the time series directly corresponds to the number of pulses transmitted during the dwell time for a pulsed-Doppler radar. It is common to simulate a large number of realizations with the same signal parameters to test the performance of a particular signal processing technique. Each realization is from a complex, stationary, Gaussian random process, and the Gaussian model is a reasonable choice for representing the combined signal from a large group of hydrometeors. By varying the signal parameters, a comprehensive analysis of the overall performance of the technique can be carried out. Even though the actual time series from a weather radar does not always match the Gaussian spectral model, the model is still the standard method for simulating time series data for testing signal processing algorithms. There are also cases where only one or two realizations (two for dual-polarization signals) are simulated for a resolution volume, especially when simulating a realistic weather profile. The simulator approaches described in this paper will not be

Corresponding author: Christopher Curtis, chris.curtis@noaa.gov

DOI: 10.1175/JTECH-D-17-0215.1

significantly faster for a single realization, but they should result in a more accurate realization for some sets of signal parameters. There is a discussion later about which simulator may be slightly faster in this particular case.

There are two main types of time series simulators used for simulating radar weather data corresponding to a particular resolution volume, and they are very closely related. The most well-known example of the first type can be found in Zrnić (1975), although similar methods were introduced around the same time, such as the one described by Thompson (1973). The main idea is to start with the desired power spectral density (taking into account spectral aliasing if necessary). Next, multiply the amplitude spectrum (the square root of the power spectral density) times an appropriate set of random numbers. Finally, use an inverse discrete Fourier transform (DFT) to produce a time series realization. The inverse DFT is commonly implemented using an inverse fast Fourier transform (FFT) for efficiency. This type of simulator will be called spectral (SP), since the time series depends on the desired power spectral density. The second main type of time series simulator depends on the desired autocorrelation function, called autocorrelation (AC). The simulation steps are similar to the spectral simulator except that the power spectral density is produced by calculating the discrete Fourier transform of the autocorrelation function. An example of this type of simulator was described by Frehlich and Yadlowsky (1994), and an earlier example that was used to study hydrological and geophysical time series was mentioned in Davies and Harte (1987). There are subtle differences in the two approaches, and they will be compared in more detail in section 3.

The current simulation methods have served us well for the past few decades, so one might wonder if new approaches are truly needed. With the advent of faster computers, thousands or millions of realizations can be produced in a short amount of time, but simulations that test a wide range of weather (and sometimes ground clutter) signal characteristics can take several hours or even several days. Improving performance can have a significant effect on these multihour or multiday simulation runs. There are also cases where weather signals with narrow spectrum widths (e.g., simulations of ground clutter data) can lead to significant errors when fixed simulation lengths are utilized. Figure 1 shows the magnitude in decibels (dB) of the calculated ensemble average of the autocorrelation functions for the SP and AC simulators compared to the desired, unbiased autocorrelation function. Both autocorrelation functions were computed using 100 000 realizations with a spectrum

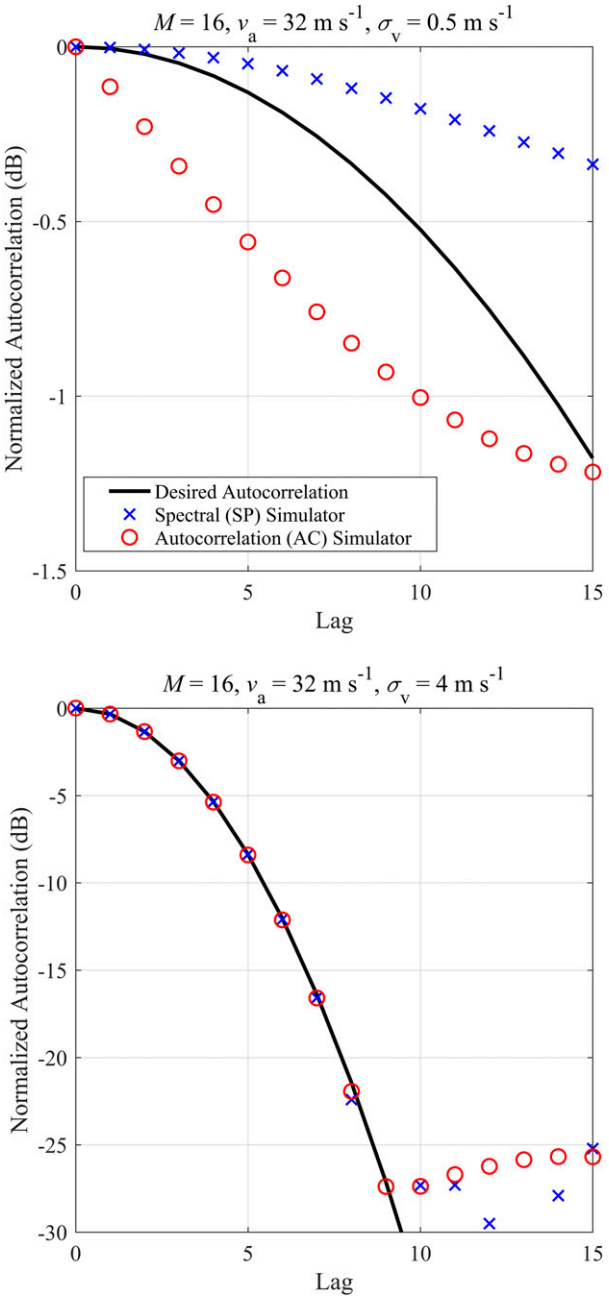

FIG. 1. Normalized autocorrelations computed from an average of 100 000 realizations of the fixed-simulation-length versions of the SP and AC simulators. The desired autocorrelation is shown for comparison. Here $M = 16$, and $v_a = 32$ m s^{-1}. The spectrum width for the top panel is 0.5 m s^{-1} and for the bottom panel is 4 m s^{-1}.

width σ_v of 0.5 m s^{-1} for the top panel and 4 m s^{-1} for the bottom panel, both with a maximum unambiguous velocity $v_a = 32$ m s^{-1}. The normalized autocorrelation is used to clearly show the level of the autocorrelation below the peak; thus, the absolute simulated power does not affect the results and is not included as a simulation

parameter throughout. The desired number of simulated samples or sample length M is 16. When simulating pulsed-Doppler radar data, M is also the number of pulses. A simulation (or FFT) length of $3M = 48$ was used for the SP simulator, and a simulation length of $2M = 32$ was used for the AC simulator. This is the number of samples simulated to produce a single realization, but only M samples are returned. Zrnić (1975) mentions that additional correlation can be produced because of the finite window (simulation length), but a specific simulation length was not recommended. The factor of 3 used here was based on a later paper by Sachidananda and Zrnić (1999). Frehlich and Yadlowsky (1994) state that a simulation length of $2M$ is required, but they also say that the simulation length must be long enough to ensure that the spectrum is positive. Even though different fixed simulation lengths may be used, these values seem to be common in actual implementations and clearly illustrate the effects of insufficient simulation length. Some additional comparisons will be made using the same fixed simulation length for both the SP and AC versions to illustrate fundamental differences in the two types of simulators.

For the narrower spectrum width of $0.5\,\mathrm{m\,s^{-1}}$, both of the simulators have significant departures from the desired autocorrelation function. This is a relatively narrow spectrum width but not excessively so. It could be used when simulating ground clutter or when testing a spectrum width estimator at narrow spectrum width values. The SP simulator shows the effects of additional correlation, but the AC simulator shows significantly reduced correlation. The most likely explanation for the reduced correlation is the truncation of the autocorrelation function. Both of the simulators perform well for the $4\,\mathrm{m\,s^{-1}}$ spectrum width, but they do depart from the autocorrelation function below about 20 dB. This behavior is common for these simulators even when the signal-to-noise ratio is high (100 dB in this case). It seems like the autocorrelation function is faithfully simulated only down to approximately 20–25 dB below its peak in all of the cases that were examined. To address these issues with fixed simulation lengths, the simulation length could be extended on an ad hoc basis for narrower spectrum widths, but an approach for finding the appropriate simulation length based on the simulated signal parameters is suggested. This can lead to longer simulation lengths for narrow spectrum widths but also shorter simulation lengths for wider spectrum widths. The key is to find the shortest possible simulation length while still accurately simulating the time series data.

Matching the autocorrelation is one important way to measure the accuracy of a time series simulator, but another is accurately matching the power spectral density (PSD). The computed PSD depends on several factors, including the signal-to-noise ratio (SNR) and the window applied to the time series. An SNR of 100 dB was utilized for these simulations because it is similar to the maximum dynamic range of current weather surveillance radars. To compute the PSD from the simulated time series, a Chebyshev window with 150-dB sidelobes was used. An aggressive window such as this one clearly shows the shape of the spectrum all the way down to the noise. Figure 2 shows the magnitude in decibels of the PSD for the SP and AC simulators compared to the desired PSD. The simulation parameters are identical to those used for Fig. 1. In fact, the same time series data were used to estimate the mean of both the autocorrelation and the PSD. The simulated velocity is $0\,\mathrm{m\,s^{-1}}$, which results in a centered spectral peak. Since changing the velocity just shifts the spectrum, a simulated velocity of $0\,\mathrm{m\,s^{-1}}$ will be used for all of the simulations in the paper. The desired PSD is the sum of the theoretical, windowed Gaussian spectrum and an appropriately scaled white noise spectrum, which incorporates the effects of added noise. The windowed Gaussian spectrum is computed by multiplying the theoretical Gaussian autocorrelation by the autocorrelation function of the window and then transforming to the spectral domain using the DFT. For the narrower spectrum width of $0.5\,\mathrm{m\,s^{-1}}$, the simulation length for the SP simulator ($3M = 48$) is sufficient to visibly match the PSD. The AC simulator does not match the spectrum below about −20 dB. This is an effect of truncating the autocorrelation function and will be discussed in more detail in section 3. Both simulators match the PSD well when $\sigma_v = 4\,\mathrm{m\,s^{-1}}$; significant mismatches tend to occur only at narrow spectrum widths. The goal of this paper is to find versions of the simulators that are both accurate and efficient for a wide range of simulation parameters. Accuracy is the primary objective, since inaccurate simulators can lead to faulty conclusions, but maximizing efficiency while maintaining accuracy is also a major objective.

The rest of the paper is structured as follows. The modifications for improving the accuracy and performance of these types of time series simulators are described in section 2. The differences between the spectral and autocorrelation simulators, including the AC spectrum mismatch from Fig. 2, are discussed in section 3. The new versions of the simulators are validated in section 4 using comparisons to the original fixed-simulation-length versions. Additional considerations such as using single-precision and graphical processing units are addressed in section 5 with section 6 containing conclusions.

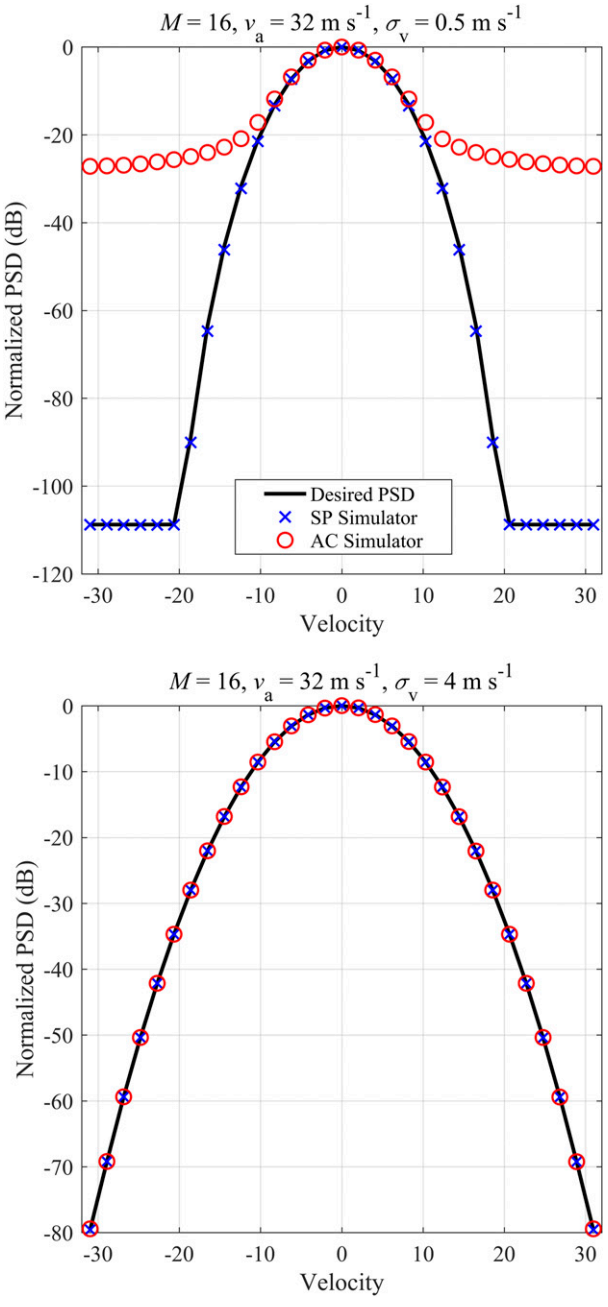

FIG. 2. Normalized PSDs computed from the average of 100 000 realizations of the fixed-simulation-length versions of the SP and AC simulators. The desired PSD is shown for comparison. The simulation parameters are the same as Fig. 1.

2. Three simulator modifications to improve accuracy and performance

The first modification to improve accuracy and performance of time series simulators was mentioned in the introduction. Setting the simulation length based on the spectrum width (and maximum unambiguous velocity) leads to more accurate simulators and can also lead to better performance. This modification is described first, but before that a slightly more detailed outline of the steps for the conventional simulators is introduced to better show how the new modifications fit into the existing simulator framework.

As described earlier, both the spectrum and autocorrelation simulators are similar except for how the desired model spectrum is calculated. Here is a procedure for both simulators with step 2-SP used for the SP simulator and step 2-AC for the AC simulator.

Inputs: desired number of samples M, signal power S_{dB} (dB), radial velocity v, spectrum width σ_v, SNR (if noise added), maximum unambiguous velocity v_a, and number of realizations R

1) Determine the simulation length M_S (fixed or parameter based).

2-SP) Calculate the spectrum directly using a formula or other method. Compute spectrum values on an extended Nyquist cointerval (from $-nv_a$ to nv_a), where n is an integer factor (usually 3 or 4). Alias the extended spectrum to produce a spectrum on the desired Nyquist cointerval (from $-v_a$ to v_a).

2-AC) Compute the length–M_S, conjugate symmetric autocorrelation function, using a formula. Calculate the DFT of the autocorrelation to produce the aliased spectrum.

3) Appropriately scale the spectrum (PSD) to obtain the desired signal power and to adjust for the power of the simulated white noise in step 4. Take the square root of the PSD to produce the amplitude spectrum (frequency response).

4) Simulate white noise to match the simulation length M_S and the desired number of realizations R. Multiply each realization of simulated noise by the amplitude spectrum.

5) Perform an inverse DFT (often with an efficient FFT algorithm) for all of the realizations to transform from the frequency domain to the time domain.

6) Return the first M samples from the M_S simulated samples to form an $M \times R$ matrix of time series data.

7) Add the desired level of white noise if necessary.

Output: $\mathbf{V}$ ($M \times R$ matrix of time series data)

The first step is to determine the simulation length M_S, and a new parameter-based approach for determining it is introduced next. The rest of the steps follow the standard simulation technique (with some minor changes) that have been described by Zrnić (1975), by Frehlich and Yadlowsky (1994), and also by Galati and Pavan (1995) when describing their fast convolution generator. The second modification for improving simulator performance is an additional thresholding step introduced between

steps 2 and 3, and the third modification replaces steps 4–6 with an alternative method for computing the inverse DFT. These changes to the simulator structure are relatively minimal, and the focus of these modifications is to keep the basic structure of the simulator (one independent realization of time series data produced for each simulated realization) unchanged.

a. Determining the simulation length

As discussed in the introduction, fixed simulation lengths of $2M$ and $3M$ have been used in the past and work reasonably well in most situations, but there are cases where these fixed simulation lengths are not sufficient (especially with narrow spectrum widths) and other situations where shorter simulation lengths could be employed to improve performance. The behavior of the autocorrelation in the bottom panel of Fig. 1 ($\sigma_v = 4\,\mathrm{m\,s^{-1}}$) suggests a possible approach to determining the simulation length. Since the autocorrelation is simulated consistently only to 20–25 dB below the peak, a threshold could be used to aid in determining the simulation length while also taking into account the circular convolution properties of the inverse DFT.

To better understand the effect of simulation length on the autocorrelation, we can look at some different simulation lengths other than those used in the bottom panel of Fig. 1. The SP simulator will be used exclusively in this section because of the mismatch between the AC simulator spectrum and the desired spectrum in Fig. 2; this mismatch issue will be addressed in section 3. With the same simulation parameters used in the bottom panel of Fig. 1 ($M = 16$, $v_a = 32\,\mathrm{m\,s^{-1}}$, and $\sigma_v = 4\,\mathrm{m\,s^{-1}}$), Fig. 3 shows calculated autocorrelations for $M_S = 16, 25$, and 32 using the SP simulator. The $M_S = M = 16$ case clearly illustrates the problem with using too short of a simulation length. By multiplying by the spectrum in the frequency domain and performing an inverse DFT, the simulated noise samples are being filtered in the time domain. Because this is implemented with a DFT, the filtering is a circular convolution and the filter aliases or wraps around. This results in the magnitude of the autocorrelation at lag 1 being equal to the magnitude at lag -1 (or lag 15 when $M_S = M = 16$), and this extends to other lags, causing the values for lags 8–14 to be larger than the desired autocorrelation. We propose to set M_S just long enough to avoid the aliasing from the filter but as short as possible for efficiency.

This aliasing effect would not necessarily cause problems when simulating data if the larger autocorrelation lags were not calculated from the time series data, but the goal of the new simulators is to accurately match the autocorrelation at all lags in addition to matching the PSD. In the future, these higher lags could be used by a new estimator, or the additional correlation could affect estimator results. The $M_S = 2M = 32$ case shows that for these parameters, a simulation length of $2M$ seems to be sufficient to accurately match the autocorrelation, but a shorter simulation length between M and $2M$ could also be sufficient. Finding this value will lead to a general procedure for choosing the simulation length.

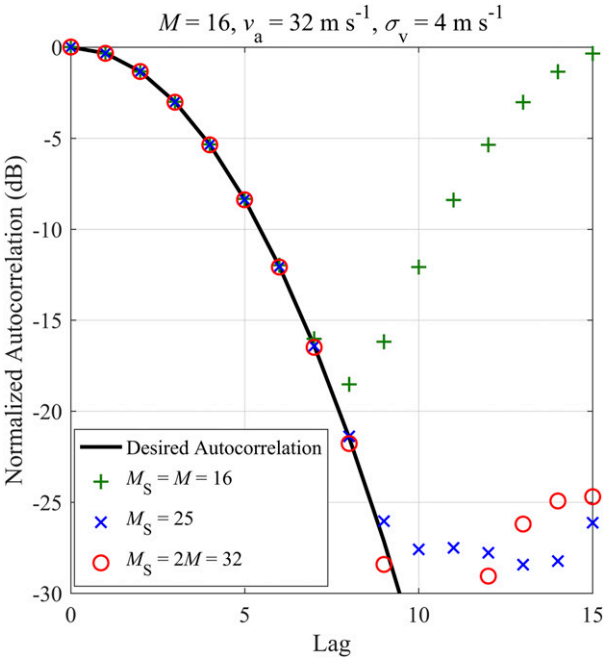

FIG. 3. Normalized autocorrelations computed from the average of 100 000 realizations of the fixed-simulation-length version of the SP simulator. Here $M = 16$, and $v_a = 32\,\mathrm{m\,s^{-1}}$. The spectrum width is $4\,\mathrm{m\,s^{-1}}$, and results for three simulation lengths are shown: 16, 25, and 32.

For the simulation parameters in Fig. 3, the desired autocorrelation drops at least 25 dB below the peak at lag 9. As mentioned previously, the autocorrelation is not measured accurately this far below the peak. The simulation length needs to be increased so that the aliasing from the circular convolution does not affect the lags from 8 to 15. Because the magnitude of the autocorrelation is symmetric, the magnitude of the autocorrelation at lag 9 equals the magnitude at lag -9. For $M_S = M = 16$, the autocorrelation at lag -9 aliases to lag $7 = 16 - 9$; this is what causes the undesired autocorrelation values from lags 8 to 15 but also does not seem to affect the value of the correlation at lag 7. If we increase M_S to 25, then the autocorrelation at lag -9 aliases to lag $15 = 25 - 9 = M - 1$. Since the magnitude of the autocorrelation is 25 dB below the peak, it should not measurably affect the autocorrelation at lag 15. As shown in Fig. 3, the calculated autocorrelation for $M_S = 25$ is very similar to the calculated autocorrelation for

$M_S = 2M = 32$. By using a threshold on the autocorrelation (e.g., 25 dB below the peak), the simulation length can be determined so that it is large enough to accurately match the desired autocorrelation but also as small as possible to make the simulator as efficient as possible. The next step is to formalize the process for a signal with a Gaussian spectrum (and a Gaussian autocorrelation).

For the Gaussian case, the autocorrelation function is a closed form expression, so the lag at which the magnitude of the autocorrelation falls below a particular threshold can be solved for directly. The normalized autocorrelation is used to simplify the calculation and is given as

$$a = \exp\left[-\frac{1}{2}\left(\frac{\pi\sigma_v t}{v_a}\right)^2\right], \tag{1}$$

where a is the magnitude of the normalized autocorrelation, σ_v is the spectrum width, v_a is the maximum unambiguous velocity, and t is a real number corresponding to the lag. There is no velocity in the equation because the velocity affects the phase but not the magnitude of the autocorrelation. The autocorrelation threshold A_T is a positive value (dB) that measures the drop in the magnitude of the autocorrelation below the peak. If the autocorrelation magnitude is set to this threshold, $a = 10^{-A_T/10}$, then the equation can be solved for t as follows:

$$t = \frac{v_a}{\pi\sigma_v}\sqrt{-2\log a} = \frac{v_a}{\pi\sigma_v}\sqrt{\frac{\log(10)}{5}A_T}, \tag{2}$$

where "log" is the natural logarithm. Since we need the first integer lag such that the autocorrelation magnitude falls below the threshold, we can set

$$k = \text{ceil}(t) = \text{ceil}\left[\frac{v_a}{\pi\sigma_v}\sqrt{\frac{\log(10)}{5}A_T}\right], \tag{3}$$

where "ceil" is the ceiling function. For the parameters in Fig. 3 ($M = 16$, $v_a = 32\,\text{m s}^{-1}$, $\sigma_v = 4\,\text{m s}^{-1}$) and an autocorrelation threshold of $A_T = 25$ dB, $k = 9$ using (3), which matches the lag from the earlier discussion. In this case, the simulation length M_S is set to $M + k$. This results in the value $M_S = M + k = 16 + 9 = 25$ that was found informally earlier. This process can also be visualized in the drawing in Fig. 4. The portion of the autocorrelation function that is above the autocorrelation threshold is shown at the top and has length $2k + 1$; the boxes represent the samples of the autocorrelation function from lags $-k$ to k. The circle on the left of the middle portion of the drawing illustrates this case. If we are using M samples from the circular convolution to form the realization, then there needs to be at least k samples between the two ends of the realization to ensure that the larger autocorrelation values do not alias. This leads to a minimum value of $M + k$ for M_S.

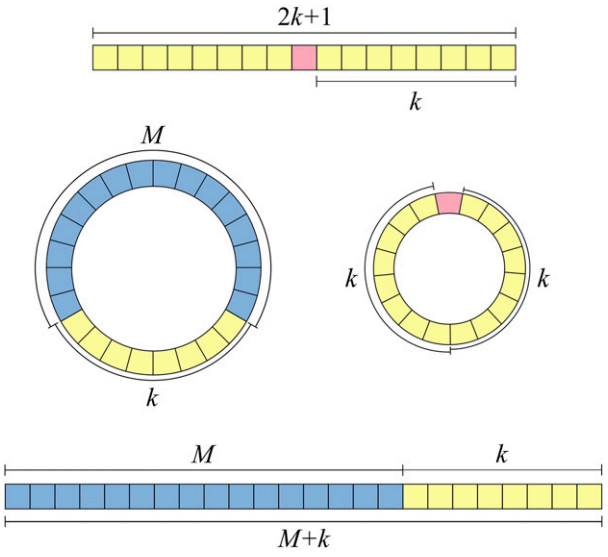

FIG. 4. Drawing showing (top) the length of the autocorrelation function with the portion of the autocorrelation function that is above A_T and (middle) the effects of circular convolution. The simulation length is either $2k + 1$ or $M + k$, whichever is larger.

There are a couple of other situations that can affect the determination of M_S. The first is straightforward and is related to the number of autocorrelation values above the threshold. The circle on the right of the middle part of the drawing shows that M_S also needs to be at least $2k + 1$ so that the autocorrelation does not wrap around and result in larger-than-expected autocorrelation values. Given a particular k computed from A_T, the simulation length should be set to the larger of $2k + 1$ and $M + k$.

The previous computations assume that the autocorrelation falls below the threshold before lag $M - 1$. There is another case that needs to be considered when the autocorrelation is still above the threshold at lag $M - 1$. This normally occurs at narrower spectrum widths. Figure 5 illustrates this situation using the same parameters used in the top part of Fig. 1 ($M = 16$, $v_a = 32\,\text{m s}^{-1}$, $\sigma_v = 0.5\,\text{m s}^{-1}$) but with different simulation lengths. In this case, the magnitude of the desired autocorrelation is less than 1.5 dB below the peak at lag $M - 1$. As shown in Fig. 1, the autocorrelation does not match when $M_S = 2M = 32$. The M_S calculated using the autocorrelation threshold $A_T = 25$ dB is 141, and the result closely matches the desired autocorrelation as expected. But, the computed autocorrelation using a simulation length of $M_S = 95$ is also illustrated in the figure and seems to match just as well as $M_S = 141$.

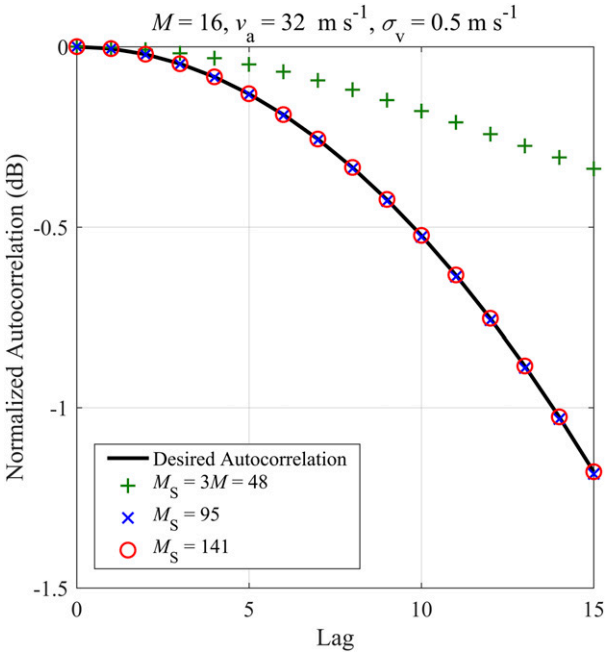

FIG. 5. Normalized autocorrelations computed from an average of 100 000 realizations of the fixed-simulation-length version of the SP simulator. Here $M = 16$, and $v_a = 32$ m s^{-1}. The spectrum width is 4 m s^{-1}, and results for three simulation lengths are shown: 48, 95, and 141.

When the autocorrelation does not fall off quickly at lag $M - 1$, the autocorrelation threshold can be set to 10 dB below this autocorrelation value instead of 25 dB below the peak.

The autocorrelation is still aliasing in this case, but the values are at least 10 dB below the value at lag $M - 1$. The effect of the aliasing is negligible, so the simulation is still accurate, but it saves a significant amount of time when simulating weather signals with narrow spectrum widths. This 10-dB threshold was found by testing the simulator with a wide variety of parameters and is validated as part of section 4. The formula for computing A_T in this case is based on the magnitude of the normalized autocorrelation from (1):

$$A_T = 10 - 10\log_{10}\left(\exp\left\{-\frac{1}{2}\left[\frac{\pi\sigma_v(M-1)}{v_a}\right]^2\right\}\right)$$
$$= 10 + \frac{5}{\log(10)}\left[\frac{\pi\sigma_v(M-1)}{v_a}\right]^2, \quad (4)$$

where "$\log_{10}$" is the base-10 logarithm, and "exp" is the base of the natural logarithm. Before determining M_S, the autocorrelation threshold A_T is chosen as the minimum of the result of (4) and the fixed value of 25 dB discussed earlier.

This leads to an algorithm for choosing M_S that can replace step 1 of the generic simulator procedure given earlier:

1) Set the simulation length $M_S = \max(2k + 1, M + k)$, where

$$k = \text{ceil}\left[\frac{v_a}{\pi\sigma_v}\sqrt{\frac{\log(10)}{5}A_T}\right].$$

The autocorrelation threshold is calculated as

$$A_T = \min\left\{25, 10 + \frac{5}{\log(10)}\left[\frac{\pi\sigma_v(M-1)}{v_a}\right]^2\right\},$$

for a Gaussian spectral model.

This leads to an M_S that is long enough to accurately match the autocorrelation but that can also be shorter than a fixed simulation length in some cases. This ensures that the simulator is accurate and also faster than the fixed-simulation-length simulator when feasible. This discussion has focused on determining M_S based on the autocorrelation rather than the PSD. From the bottom panels of Figs. 1 and 2, it seems that the autocorrelation is more sensitive to simulation length than the PSD. This ends up being true over a wide set of parameters and will be confirmed in section 4.

b. *Adding a spectral threshold*

The second approach for improving the performance of the SP and AC simulators can be thought of as extending the threshold idea from the autocorrelation domain to the spectral domain. The calculation of the PSD is different from the autocorrelation because it can be computed accurately much farther below the peak of the spectrum (especially when calculating the ensemble average). At first, this would seem to preclude the use of a threshold similar to the autocorrelation threshold, but there are practical effects that are also involved. A radar system has a particular dynamic range that ensures that the spectrum can be accurately computed only down to the system noise level. We can take this dynamic range into account and set a threshold that is based on the highest expected SNR. As mentioned earlier, a maximum SNR of 100 dB will be used throughout the paper to explore threshold levels, but the conclusions could be applied to simulations with other SNR values (or for simulating systems with higher or lower dynamic ranges).

Another way to understand the motivation for this threshold is to look at an example. Figure 6 shows the desired PSD computed without additive noise along with a PSD computed from 100 000 realizations of the

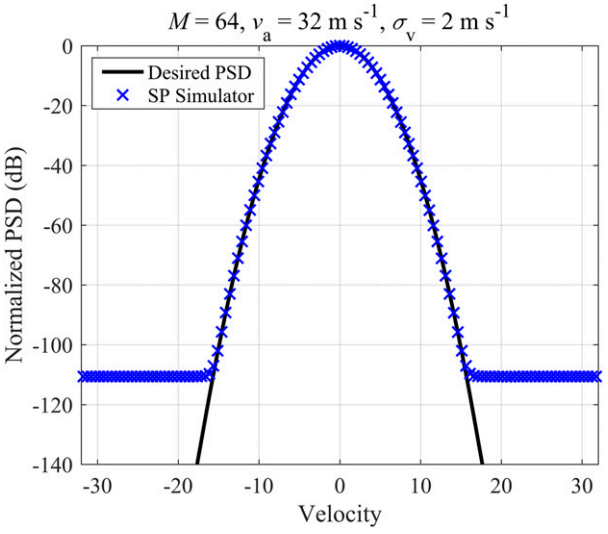

FIG. 6. Normalized PSD computed from an average of 100 000 realizations of the fixed-simulation-length version of the SP simulator. The desired PSD is computed assuming no noise is present.

SP simulation with an SNR of 100 dB and a Chebyshev window with sidelobes at −150 dB. The spectrum is normalized to the peak of the weather portion of the spectrum, which causes the noise level to show up at a value below 100 dB. If the weather spectrum were completely flat, then the noise level would show up at the expected value. The other simulation parameters are $M = 16$, $v_a = 32\,\mathrm{m\,s^{-1}}$, and $\sigma_v = 2\,\mathrm{m\,s^{-1}}$. The simulation length M_S is 82 and was computed using the procedure introduced in the previous subsection. The point of this figure is to show that using the whole amplitude spectrum when multiplying by the simulated noise (as in step 4 of the simulator outline) is not necessary. In this case, the part of the spectrum below about 110 dB is not going to affect the simulated PSD. The spectrum could be thresholded using a spectral threshold S_T to keep only the part of the PSD that is necessary for accuracy. The values of the PSD that are not needed could be set to zero, but there is also no reason to simulate noise values that are only going to be multiplied by zero. If only the nonzero part of the amplitude spectrum (computed from the desired PSD) is used, then a significant amount of time can be saved in some cases by simulating fewer than the M_S samples of white noise for each realization. This change is especially effective for narrower spectrum widths.

As with the autocorrelation threshold, the issue is what value should be used for the spectral threshold. It depends on the desired SNR, the normalized spectrum width ($\sigma_{vn} = \sigma_v/2v_a$), and the window being used to calculate the PSD. There is also a weak dependence on M. Because of the dependence on the window, different thresholds could be calculated if the window was known, but we chose to use a simple threshold that is valid for a wide variety of spectrum widths and windows. Not only does the first sidelobe level of the window affect the threshold value but also the sidelobe rolloff. Through inspection of many sets of simulation parameters, a reasonable rule of thumb seems to be setting the spectral threshold to a value 35 dB greater than the desired SNR. Determining the threshold based on both the spectrum width and window might improve performance marginally, but the simple threshold based on the SNR gives most of the benefit without knowledge of the window. That is part of the reason that a Chebyshev window with 150-dB sidelobes is used to calculate the PSD; this is an aggressive window that should reveal as much of the spectrum as possible with a simulation SNR of 100 dB. Setting S_T = SNR + 35 will lead to additional nonzero spectrum values for some parameter and window choices, but it should ensure that the expected value of the PSD matches the desired PSD in all but the most extreme cases. There are also cases where the spectrum width is wide enough that none of the spectrum will be thresholded. Setting S_T = SNR + 35 seems to be the best choice if only one value is to be used.

To see the results of using different thresholds, Fig. 7 shows the calculated PSDs using the same parameters from Fig. 6, but it also includes results from the SP simulator using two different spectral threshold values: 25 and 135 dB. Only half of the spectrum is shown to make the results easier to observe. All of the spectra level off around −110 dB like the simulated spectrum in Fig. 6 because of the added noise and the normalization. For the recommended value of S_T = SNR + 35 = 135 dB, the simulated PSD matches the desired spectrum just as well as the nonthresholded version. For S_T = 25 dB, the PSD matches closely down to around 25 dB below the peak and drops off more quickly after that. This value would be sufficient if no window were used when processing the simulated data, but the spectrum clearly does not match when a more tapered (i.e., more aggressive) window is employed. One possible way to further improve the runtime of the simulator would be to try to set the spectral threshold to the smallest value possible depending on the simulation parameters. To get an idea of the effect, the relative runtimes for the two thresholded versions with respect to the nonthresholded version are as follows: 1.40 times faster for S_T = 135 dB and 1.83 times faster for S_T = 25 dB with 50 000 realizations. A significant amount of the time reduction is achieved using the more conservative threshold, but there are additional time savings using a smaller value. The current recommendation is to set S_T = SNR + 35 to avoid inadvertently using a value that is insufficient. This is a conservative choice, and

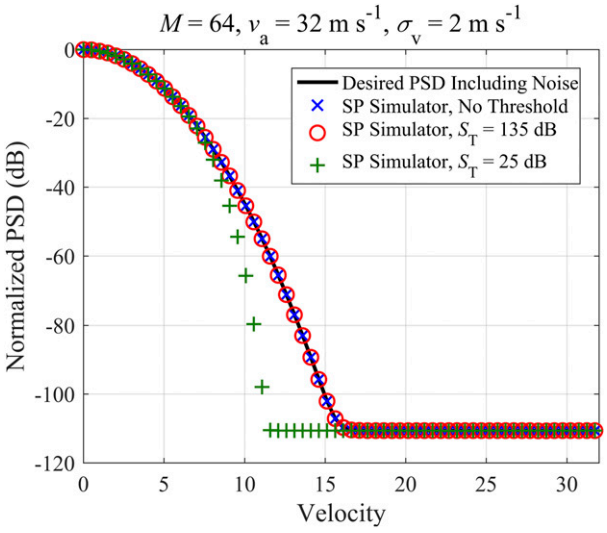

FIG. 7. Normalized PSDs computed from averages of 100 000 realizations of the fixed-simulation-length version of the SP simulator without a spectral threshold and with two fixed values: 25 and 135 dB.

additional time savings can be realized if the particular window and spectrum width are taken into account.

This new step can be captured in the simulator outline by adding a step between steps 2 and 3 (referred to as step 2.5):

2.5) If the SNR is defined, then set S_T = SNR + 35; otherwise, set S_T to a default value (e.g., 135 dB). Set all values in the spectrum to zero that are more than S_T (dB) below the peak of the spectrum.

There are times when the time series simulator is used without adding noise (e.g., when multiple simulated signals are added together). The spectrum thresholding in that case could be skipped, a default value could be used that is set based on the expected maximum SNR, or the user could pass the expected SNR (an SNR hint) to the simulator. For these simulators, the SNR hint was chosen to give the most flexibility, but the effect is not seen in the figures since an SNR of 100 dB is used throughout. Furthermore, the thresholded values are set to zero, but the processing in later steps will need to recognize the zeros in the spectrum to avoid simulating unnecessary data.

The spectral threshold also leads to a change in step 2-SP for the SP simulator: the step when the spectrum is calculated. The original step 2-SP includes an integer factor that is used for aliasing the spectrum. Instead of using a fixed factor, the spectral threshold can be utilized to determine how much the spectrum needs to be extended before calculating the aliased spectrum, since the spectrum needs to be computed only S_T (dB) below the spectral peak. This ensures that the spectrum is accurate and also that unnecessary values are not computed. The effect of the spectral threshold on the accuracy of the autocorrelation and the PSD will be assessed in section 4.

c. *Using a partial DFT to improve efficiency*

This subsection focuses on steps 4–6 from the simulator outline: multiplying simulated white noise variates by the amplitude spectrum, performing an inverse DFT to produce time series data, and returning M samples per realization. In general, it seems like the inverse DFT (using the FFT algorithm) would be the fastest way to transform the data from the frequency domain to the time domain, but there are a few reasons why a partial DFT can improve performance in some cases. The first is that only M samples of data are needed for each realization, but M_S samples are computed from the full inverse DFT. The normal process is to return the first M samples as described in step 6. If the inverse DFT is implemented using an inverse DFT matrix, then we can return the first M samples by using only the first M rows of the matrix. Even though matrix multiplication is normally slower than an FFT, this can reduce the number of computations significantly in cases of narrower spectrum widths. Another way that a partial DFT can be faster is by multiplying the white noise by the amplitude spectrum in one step rather than the two steps needed for an FFT. In the simulator outline, each length–M_S realization of the white noise is multiplied by the amplitude spectrum before the inverse DFT is performed, since it is normally implemented with an FFT instead of a DFT matrix. With the partial inverse DFT matrix, we can multiply each row of the matrix by the amplitude spectrum to produce the same effect.

It may be easier to see this with an equation. The matrix of white noise $\mathbf{N}$ that is used in step 4 is $M_S \times R$, where R is the number of realizations. For the full inverse DFT using an FFT algorithm, every column of $\mathbf{N}$ is multiplied by the amplitude spectrum, which is $M_S \times 1$ in this case. Then, the full inverse DFT is computed for each column. With the inverse DFT matrix, the inverse DFT can be performed by multiplying by $\mathbf{W}^*$, where $\mathbf{W}$ is the $M_S \times M_S$ DFT matrix and * signifies the complex conjugate. Since matrix multiplication multiplies each row of the first matrix by each column of the second matrix (and then sums the products), the amplitude spectrum can be applied by multiplying each row of $\mathbf{W}^*$ by the amplitude spectrum, which is $1 \times M_S$ when multiplying by rows instead of columns. If we call this new matrix $\mathbf{D}$, then the multiplication by the amplitude spectrum and the inverse DFT can be implemented by the matrix multiplication $\mathbf{DN}$. This shows that the multiplication of the amplitude spectrum and the inverse

DFT can be implemented using a single-matrix multiplication.

There are a couple of other modifications to this matrix that can save additional time. The matrix $\mathbf{D}$ is $M_S \times M_S$, but we are simulating only M samples. If we take only the first M rows of $\mathbf{D}$ and multiply $\mathbf{N}$ by this new $M \times M_S$ partial DFT matrix, we can directly produce the $M \times R$ matrix of time series data. One last consideration is to take into account the zeros in the amplitude spectrum and to avoid all of the multiplications and additions with zeros. Remember that the amplitude spectrum may have values set to zero from the spectrum threshold S_T that was previously discussed. If some of the values of the amplitude spectrum are zero, we can remove those columns from $\mathbf{D}$ without affecting the final result. Let $\tilde{\mathbf{D}}$ be the $M \times \tilde{M}_S$ matrix formed from the first M rows of $\mathbf{D}$ with the zero columns removed, where $\tilde{M}_S$ is the number of nonzero values in the amplitude spectrum. In the end, we can produce an $M \times R$ matrix of time series data by multiplying an $\tilde{M}_S \times R$ matrix of complex white noise $\tilde{\mathbf{N}}$ by $\tilde{\mathbf{D}}$. This minimizes the number of computations when using the DFT matrix, and it will be shown in section 4 to be faster than the inverse FFT implementation in many cases.

This new procedure replaces steps 4–6 in the simulator outline with one new step:

4–6) Form the $M_S \times M_S$ matrix $\mathbf{D}$ by taking the conjugate of the DFT matrix $\mathbf{W}$ and multiplying each row by the amplitude spectrum that has already been thresholded using S_T. Make a new $M \times \tilde{M}_S$ matrix $\tilde{\mathbf{D}}$ from the first M rows of $\mathbf{D}$ and the $\tilde{M}_S$ nonzero columns. Produce an $M \times R$ matrix of time series data by multiplying an $\tilde{M}_S \times R$ matrix of complex white noise by $\tilde{\mathbf{D}}$.

This simulator will be validated in section 4 using a wide range of simulation parameters. The runtimes will also be compared to a version of the simulator that uses the inverse FFT algorithm to determine which provides the best performance. The next section explores the differences between the SP and AC simulators.

3. Comparing spectral and autocorrelation simulators

As shown in the simulator outline, both SP and AC simulators produce a spectrum in steps 2-SP and 2-AC, respectively, that is then multiplied by white noise for each realization. For the SP simulator, that spectrum can be produced directly if there is an explicit function for the spectral model. In contrast, the AC simulator produces the spectrum by computing the autocorrelation function first and then using a DFT to transform the autocorrelation to the frequency domain. Since there is an explicit formula for a Gaussian signal in both the spectral and autocorrelation domains, either type of simulator can be used. Because of the relationship between the spectrum and the autocorrelation, it should be possible to compute one from the other, and the results should be the same. The top panel in Fig. 2 with $M = 16$, $v_a = 32\,\mathrm{m\,s^{-1}}$, and $\sigma_v = 0.5\,\mathrm{m\,s^{-1}}$ shows that this is not always the case. Even though the simulation length M_S is different, there is another issue that affects the spectrum for the AC simulator.

The autocorrelation simulator used in Fig. 2 is described in Frehlich and Yadlowsky (1994) and computes the spectrum from M_S values of the autocorrelation function (described as the desired covariance). The only restriction on the autocorrelation is that it is conjugate symmetric to ensure a real spectrum. Lags from $-\mathrm{ceil}[(M_S - 1)/2]$ to $\mathrm{floor}[(M_S - 1)/2]$ of the autocorrelation function are needed so that there are a total of M_S values when taking into account the conjugate symmetry. This autocorrelation function will be compared to the autocorrelation function calculated from the directly computed spectrum using the SP simulator. In Fig. 2, the simulation lengths are different for the AC and SP simulators. To better compare them, we can use the same parameters as in Fig. 2 but choose the same simulation length based on the procedure described earlier. This leads to a simulation length $M_S = 95$, which requires lags from -47 to 47. Figure 8 shows the magnitude of the two autocorrelation functions: one computed as described in Frehlich and Yadlowsky (1994) and the other by taking the inverse DFT of the spectrum computed for the SP simulator. The autocorrelations clearly do not match, which is the main reason that the AC and SP simulator spectra do not match in Fig. 2. Computing the autocorrelation only from lags -47 to 47 truncates the autocorrelation function. In order for the autocorrelation to match the one from the SP simulator, more lags need to be computed. The additional lags are aliased on top of the originally computed lags and lead to the smoother autocorrelation. The autocorrelation computed from the spectrum can be closely approximated by calculating the theoretical autocorrelation to 400 dB below the peak and then aliasing the autocorrelation similarly to the aliasing of the spectrum. Computing down to 400 dB seems to be necessary to approximate the spectrum as closely as possible. This value was empirically determined by testing a wide variety of signal parameters.

The actual aliasing of the autocorrelation function is nearly identical to the aliasing of the spectrum. Remember that before using the DFT to compute the

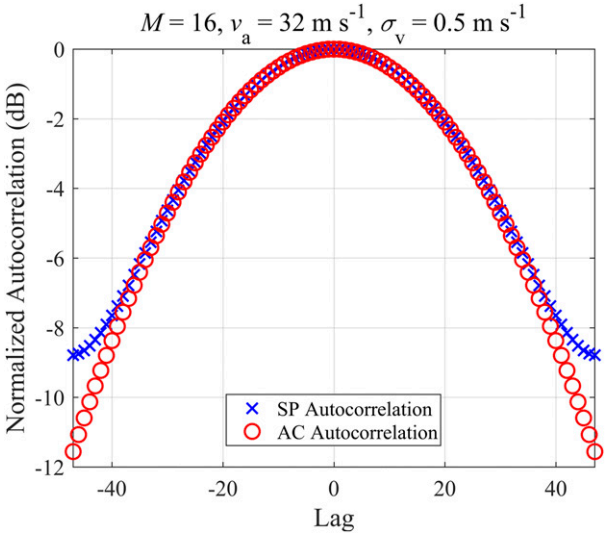

FIG. 8. Comparison of the TAC function from the original version of the AC simulator to the autocorrelation function computed by taking the inverse FFT of the spectrum computed for the SP simulator.

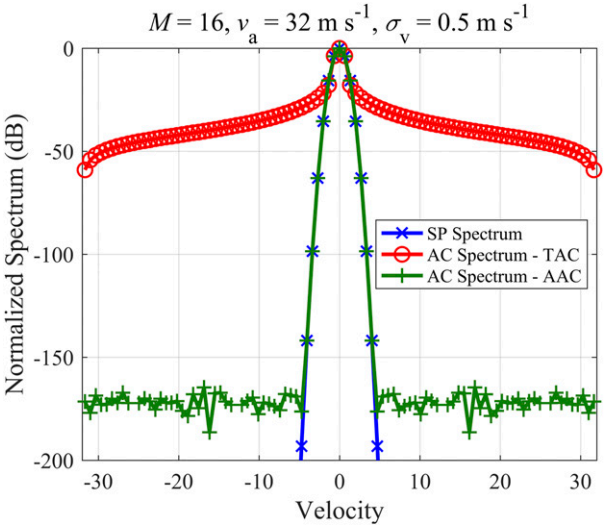

FIG. 9. Computed spectra from three different simulators: SP simulator with the spectrum calculated directly, AC simulator with TAC, and AC simulator with AAC.

spectrum, the lag-0 autocorrelation needs to be the first element of the autocorrelation array. In this implementation, the original autocorrelation array is computed with the lags from 0 to $M-1$, so the lag-0 correlation is in the correct position. Next, the lags from $-M$ to -1 are computed and added to the original array. Based on the 400-dB threshold, the process is continued for the next M lags in both the positive and negative directions until the smallest computed autocorrelation is less than the threshold. Each set of computed autocorrelations is added to the array.

The spectra produced from the three approaches (directly computed spectrum, truncated autocorrelation, and aliased autocorrelation) are shown in Fig. 9. The spectrum that is directly computed by the SP simulator is the most accurate. The spectrum produced from aliasing the autocorrelation (AAC) matches the SP spectrum to more than 150 dB below the peak. The spectrum produced from truncating the autocorrelation (TAC) is accurate only to about 15 dB below the peak. This is the main factor in the mismatched spectra from the top panel of Fig. 2. It should also be noted that the truncated autocorrelation did not lead to a mismatch when $\sigma_v = 4\,\mathrm{m\,s^{-1}}$ (the bottom panel of Fig. 2). This particular effect occurs only when simulating especially narrow spectrum widths, but aliasing the spectrum addresses the problem for any case. Because of the extra DFT and the less accurate spectrum, it is recommended that the SP simulator be used in most cases. The results should be nearly identical as long as the SNR is not larger than about 125 dB, but the extra DFT takes a small amount of additional time and could have an effect on performance when the number of realizations is small. One reason to use the AC simulator is when there is an explicit equation for the autocorrelation but not the spectrum.

The second step in the simulator outline needs to change slightly to account for aliasing the autocorrelation:

2-AC) Compute the autocorrelation so that the smallest value falls below a specific threshold; 400 dB below the peak is recommended. Alias the autocorrelation to length M_S. Calculate the DFT of the autocorrelation to produce the aliased spectrum.

We now have the necessary modifications to the simulator outline for the new simulators.

The final steps that incorporate all three modifications and the aliasing of the autocorrelation are given below:

Inputs: desired number of samples M, signal power S_{dB} (dB), radial velocity v, spectrum width σ_v, SNR (if noise added), maximum unambiguous velocity v_a, and number of realizations R

1) Set the simulation length $M_S = \max(2k+1, M+k)$, where

$$k = \mathrm{ceil}\left[\frac{v_a}{\pi\sigma_v}\sqrt{\frac{\log(10)}{5}A_T}\right].$$

The autocorrelation threshold is calculated as

$$A_T = \min\left\{25, 10 + \frac{5}{\log(10)}\left[\frac{\pi\sigma_v(M-1)}{v_a}\right]^2\right\},$$

for a Gaussian spectral model.

2-SP) Calculate the spectrum directly using a formula or other method. Compute spectrum values on an extended Nyquist cointerval (from $-nv_a$ to nv_a), where n is an integer factor based onthe spectral threshold S_T. Alias the extended spectrum to produce a spectrum on the desired Nyquist cointerval (from $-v_a$ to v_a).

2-AC) Compute the autocorrelation so that the smallest value falls below a specific threshold; 400 dB below the peak is recommended. Alias the autocorrelation to length M_S. Calculate the DFT of the autocorrelation to produce the aliased spectrum.

2.5) If the SNR is defined, set S_T = SNR + 35; otherwise, set S_T to a default value (e.g., 135 dB). Set all values in the spectrum to zero that are more than S_T (dB) below the peak of the spectrum.

3) Appropriately scale the spectrum (PSD) to obtain the desired signal power and to adjust for the power of the simulated white noise in step 4. Take the square root of the PSD to produce the amplitude spectrum (frequency response).

4–6) Form the $M_S \times M_S$ matrix $\mathbf{D}$ by taking the conjugate of the DFT matrix $\mathbf{W}$ and multiplying each row by the amplitude spectrum that has already been thresholded using S_T. Make a new $M \times \tilde{M}_S$ matrix $\tilde{\mathbf{D}}$ from the first M rows of $\mathbf{D}$ and the $\tilde{M}_S$ nonzero columns. Produce an $M \times R$ matrix of time series data by multiplying an $\tilde{M}_S \times R$ matrix of complex white noise by $\tilde{\mathbf{D}}$.

7) Add the desired level of white noise if necessary. Output: $\mathbf{V}$ ($M \times R$ matrix of time series data)

In the next section, the accuracy and performance of these new, modified simulators will be compared to the fixed-simulation-length SP and AC simulators.

4. Validating the new simulators

To validate the accuracy of the new simulators and to compare their performance, two versions of the new simulators will be compared to the fixed-length simulators from Figs. 1 and 2. The first is an SP simulator using all of the approaches described in section 2, including the partial DFT matrix approach; this simulator will be referred to as SP-DFTM. The second simulator is an AC simulator using the first two approaches in section 2 and the autocorrelation aliasing from section 3, but the DFT is computed using an FFT algorithm instead of the partial DFT matrix. This will be referred to as AC-FFT. One minor change with the AC-FFT simulator is that the simulation length is increased to the next largest multiple of four to take advantage of fast FFT algorithms. Testing shows that FFT lengths that are multiples of four lead to more consistent runtimes. The SP-DFTM and AC-FFT simulators allow the validation of both the spectral and autocorrelation approaches and also a runtime comparison between the partial DFT matrix and FFT methods. The effect of the single additional DFT in the AC-FFT simulator is negligible when looking at large numbers of realizations. Figure 10 shows the average autocorrelations and PSDs computed from 100 000 realizations of both simulators for the parameters shown in the top panels of Figs. 1 and 2 (M = 16, v_a = 32 m s^{-1}, and σ_v = 0.5 m s^{-1}). The new method for computing the simulation length results in very good accuracy for both of the simulators; the autocorrelations and PSDs match the desired ones closely. The spectral threshold does not seem to affect the accuracy as predicted. Setting the spectral threshold to a value 35 dB larger than the SNR is sufficient in these tests to match the accuracy of the unthresholded simulator.

To more comprehensively test both the SP-DFTM and AC-FFT simulators, a simulation was run using a wide variety of sample lengths and spectrum widths to test the simulators under different conditions. Four sample lengths were chosen, M = [4, 16, 64, 1024]; this includes some commonly used values [16, 64] and a couple of extreme values [4, 1024]. Eight spectrum widths were chosen, σ_v = [0.125, 0.25, 0.5, 1, 2, 4, 8, 16], with a maximum unambiguous velocity v_a = 32 m s^{-1}. Thus, the normalized spectrum widths that were tested are σ_{vn} = [1/256, 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2]. The spectral and autocorrelation simulators with fixed simulation lengths M_S of 3M and 2M, respectively, were implemented based on the original simulator outline. These simulators will be referred to as SP-3M and AC-2M, respectively. An additional simulator, AC-3M, was included to show the errors from the truncated autocorrelation that were described in section 3. By using the same fixed simulation length, 3M, for both the SP-3M and AC-3M simulators, the differences are only from the truncation of the autocorrelation and not the difference in the simulation length.

Each of the five simulators was run 100 times with 50 000 realizations to compute mean squared error (MSE) values, and runtimes were computed for all of the simulators except the AC-3M simulator. The runtime of the fixed-simulation-length simulators is almost completely dependent on the fixed simulations' length. The runtimes were also calculated for 100 simulation runs with 1000 realizations to see whether the runtime

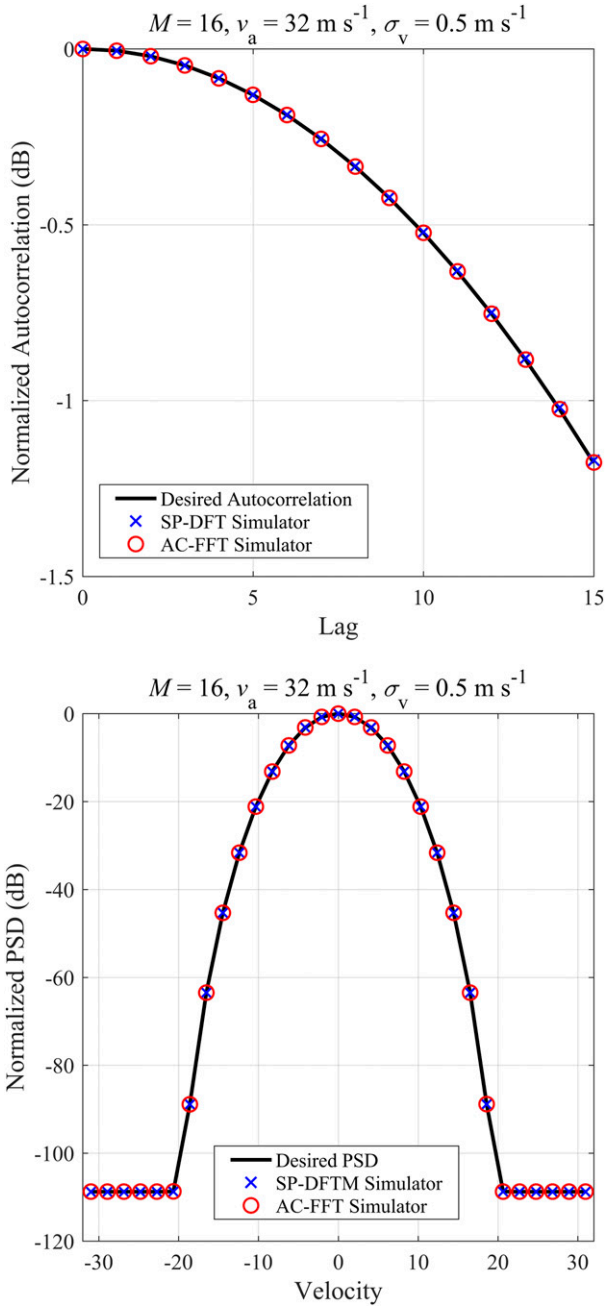

FIG. 10. Normalized autocorrelations and PSDs computed from averages of 100 000 realizations of the SP-DFTM and AC-FFT simulators with the same parameters as the top panels of Figs. 1 and 2 ($M = 16$, $v_a = 32\,\mathrm{m\,s^{-1}}$, and $\sigma_v = 0.5\,\mathrm{m\,s^{-1}}$).

results change with fewer realizations. To get the final plotted MSE value, an MSE estimate was computed for each autocorrelation and PSD with respect to the desired autocorrelation or PSD measured in decibels to allow for the smaller values to be taken into account. The desired autocorrelation was set to −50 dB when any of the values were at least −50 dB below the peak to keep the MSE from being too large, since the measured autocorrelations do not drop off like the theoretical autocorrelation. The desired PSD was not truncated, but the SNR was used to calculate a spectrum that includes the noise floor. A Chebyshev window with 150-dB sidelobes was also used when computing the desired PSD. The 100 calculated MSE estimates for each set of parameters were then averaged to get a final MSE value. Because of the way the MSE values are computed, they are not very useful in an absolute sense, but the method does seem to be sensitive to relatively small differences, even for extremely tiny values.

To validate the accuracy of the simulators, two additional simulators were included. The first simulator utilizes longer simulation lengths: either 1.25 times the M_S value computed for the SP-DFTM and AC-FFT simulators or $M_S + 25$, whichever is greater. This simulator uses an upper bound for the simulation length and is called Baseline-UB. The spectral threshold was not used, and the FFT algorithm was utilized to produce the time series. If the simulator errors for SP-DFTM and AC-FFT match the Baseline-UB simulator, then the value of M_S should be sufficient, and the use of the spectral threshold should not affect the accuracy. The comparison to the Baseline-UB simulator validates that the computed M_S is sufficient, since a simulation with a significantly longer simulation length does not result in smaller errors, but a successful test does not show that M_S is as small as possible. For this, a lower-bound baseline simulator was used. This simulator utilized shorter simulation lengths: either 0.9 times the M_S value computed for the SP-DFTM and AC-FFT simulators or M, whichever is larger. This simulator is called Baseline-LB. If the simulator errors for SP-DFTM and AC-FFT match the Baseline-LB simulator, then the simulation length could be shorter than one computed using the autocorrelation threshold. We expect to see larger errors from Baseline-LB, which will verify that the simulation length is close to being as short as possible. The simulators were run using MATLAB with double-precision output. The processer was an Intel Xeon E5-2630 with 64 GB of RAM. The performance could vary significantly on different processors, but this configuration is reasonable for showing the performance with a modern CPU.

Figure 11 shows the error results of the simulation for the five simulators previously mentioned (AC-FFT, SP-DFTM, AC-2M, AC-3M, SP-3M) and also includes the baseline simulators (Baseline-UB, Baseline-LB) for comparison. The AC simulator results are depicted in red and the SP results in blue. The AC-FFT and SP-DFTM simulators are displayed using solid lines, while the fixed-parameter simulators are displayed with

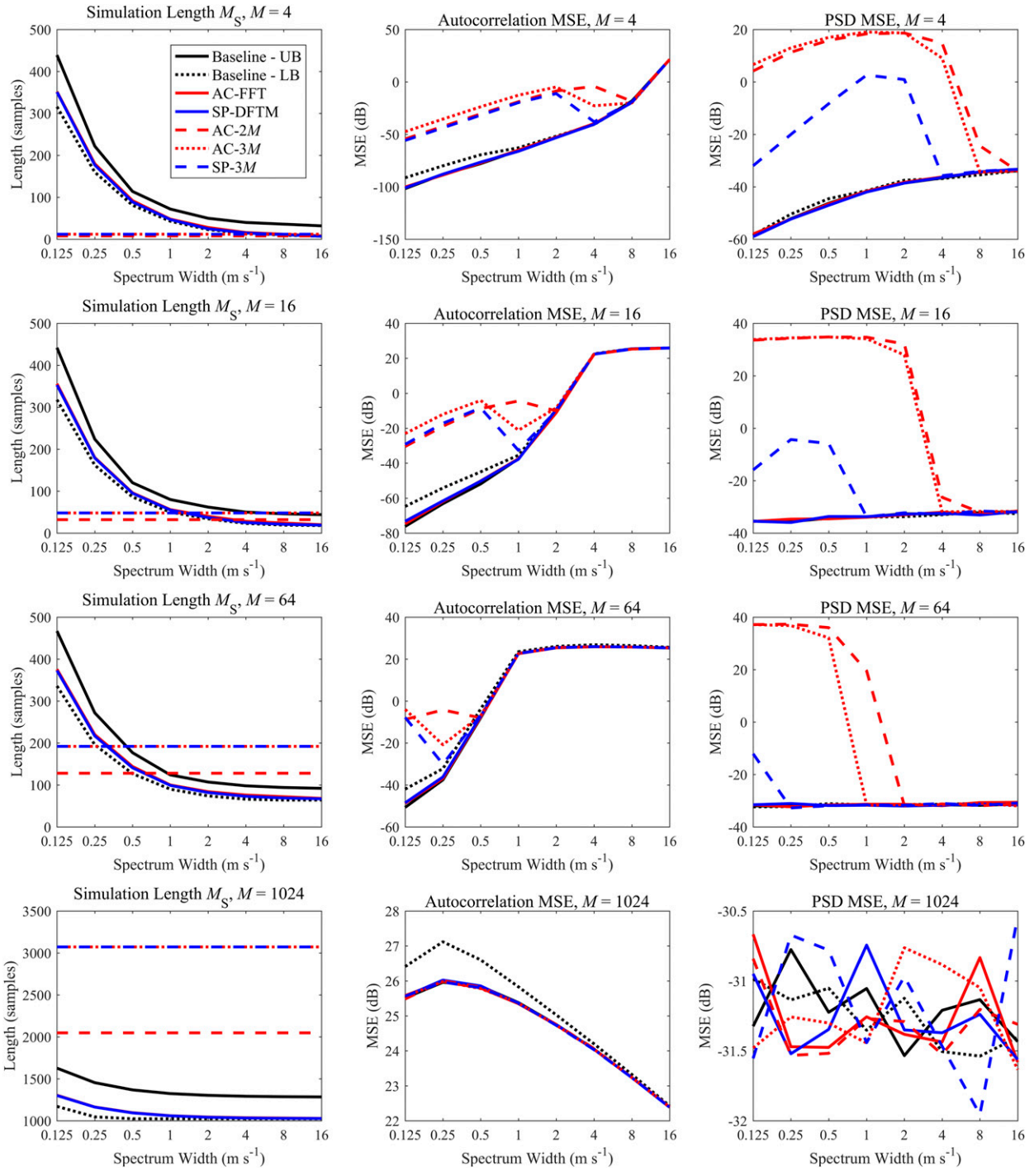

FIG. 11. Simulation lengths and errors for the SP-DFTM, AC-FFT, AC-2M, AC-3M, and SP-3M simulators along with errors for the two baseline simulators, Baseline-UB and Baseline-LB. Each row corresponds to a different value of M (4, 16, 64, and 1024). (left) Simulation length. MSE (dB) for the (center) autocorrelation and (right) PSD.

dashed or dotted lines. The baseline simulators are depicted with black lines with the UB simulator solid and the LB simulator dotted. The rows correspond to the four values of M. The first column shows the M_S for all seven simulators. For small M, the calculated simulation length is significantly longer than the fixed simulation lengths ($2M$ or $3M$) at narrow spectrum widths. For large M, the calculated simulation length is shorter.

The curves for the baseline simulations have a similar shape to the curve for the calculated simulation length but are shorter (Baseline-LB) and longer (Baseline-UB) as expected. The second and third columns show the accuracy of the autocorrelation and PSD, respectively, in terms of MSE (displayed in dB).

The fixed-simulation-length simulators—SP-3M, AC-2M, and AC-3M—have significantly larger errors than the Baseline-UB simulator for $M = 4$, 16, and 64. The accuracy issues affect most of the spectrum widths at $M = 4$ but fewer spectrum width values as M increases. This is consistent with the behavior of the simulation length plots in the first column. The fixed-simulation-length simulator accuracy matches the parameter-based simulators at the largest value of $M = 1024$, since the simulation lengths are all larger than the ones computed for the parameter-based simulators. The SP-3M simulator autocorrelation accuracy is similar to or slightly better than the AC-2M and AC-3M simulators. There is a slight anomaly where the AC-3M estimator is less accurate than the AC-2M estimator in some cases. We did not track down the cause of this, since both estimators are inaccurate in those cases. The accuracy results for the PSD are similar to the autocorrelation results. Because the autocorrelation is truncated for the AC-2M simulator, the PSD errors are significantly greater than for the fixed-length SP-3M simulator. The AC-3M simulator errors are closer to the AC-2M simulator than the SP-3M simulator, which shows that the truncation of the autocorrelation really is the main factor in the increased errors. The SP-DFTM and AC-FFT simulators are significantly more accurate than the fixed-parameter simulators, and for all of the cases except for the PSD MSE panel with $M = 1024$, the MSE curves are on top of the MSE curves for the Baseline-UB simulator. In the $M = 1024$ case, the PSD accuracy values are nearly the same for all seven of the simulators. The SP-DFTM and AC-FFT simulators appear to be just as accurate as the Baseline-UB simulator. The Baseline-LB simulator has larger autocorrelation errors than both the SP-DFTM and AC-FFT simulators for at least some spectrum widths for all values of M. The errors are only slightly larger for the PSD at $M = 4$, again showing that the autocorrelation is more sensitive to the simulation length. These results clearly show that the calculated simulation length is nearly as short as possible. The only time that the Baseline-LB errors are approximately the same is when both the calculated simulation length and the simulation length for Baseline-LB are near M.

Figure 12 shows the average runtimes for the SP-3M, AC-2M, AC-FFT, and SP-DFTM estimators. The first and second columns show the runtimes for 1000 and 50 000 realizations, respectively, leaving out the AC-3M simulator and the two baseline simulators. The average runtimes were calculated using a trimmed mean using the central 50 values of the 100 sorted values for the 100 simulator runs. A trimmed mean was used to avoid any outliers in the runtimes. The SP-DFTM and AC-FFT simulators were faster at all spectrum widths for $M = 64$ and 1024. The fixed-length AC-2M simulator is consistently faster than the SP-3M simulator because of the difference in the fixed simulation length. For $M = 16$, the SP-DFTM simulator is faster than the fixed length simulators, but the AC-FFT simulator is significantly slower for narrow spectrum widths. For $M = 4$, the SP-DFTM simulator has comparable runtimes to the fixed-simulation-length simulators and is considerably more accurate (as shown in Fig. 11). The AC-FFT simulator is again slower at narrow spectrum widths. The AC-FFT simulator is faster only when $M = 1024$ and the number of realizations is 1000 or at large spectrum widths when the number of realizations is 50 000. In the $M = 64$ case with 50 000 realizations and a spectrum width $\sigma_v = 2\,\mathrm{m\,s^{-1}}$, the SP-DFTM simulator takes 0.25 s, the AC-FFT simulator 0.31 s, the AC-2M simulator 0.56 s, and the SP-3M simulator 0.72 s. This is a set of parameters where all four simulators accurately match the autocorrelation and PSD. The SP-DFTM simulator is more than twice as fast as the AC-2M simulator and almost 3 times as fast as the SP-3M simulator. For most simulations, the SP-DFTM simulator is the fastest and most accurate and is recommended for a wide range of signal parameters.

There is a case that is not adequately addressed with the previous validation. There are times (e.g., when simulating a realistic weather profile) when only one or two realizations for a particular set of parameters are needed. In this case, the extra steps to compute the DFT matrix may cause the simulator to be less efficient than in cases with 1000 or 50 000 realizations. Figure 13 shows the runtimes for three different simulators: two that have already been introduced, SP-DFTM and AC-FFT; and a third, SP-FFT, that uses the threshold approaches from section 2 but uses the inverse FFT algorithm instead of a partial inverse DFT matrix. This simulator should be slightly faster than the AC-FFT simulator for small numbers of realizations because the single DFT is not utilized to produce the amplitude spectrum. To test under varying conditions similar to a realistic weather profile, the simulators were run with 1000 different sets of parameters, with 1–1000 realizations for each set. The total runtimes for all 1000 sets of parameters with varying number of realizations are plotted in Fig. 13. As predicted, the FFT simulators were faster for smaller numbers of realizations, but the SP-DFTM simulator becomes faster around 500 realizations. The SP-FFT simulator is about 2% faster on average than the

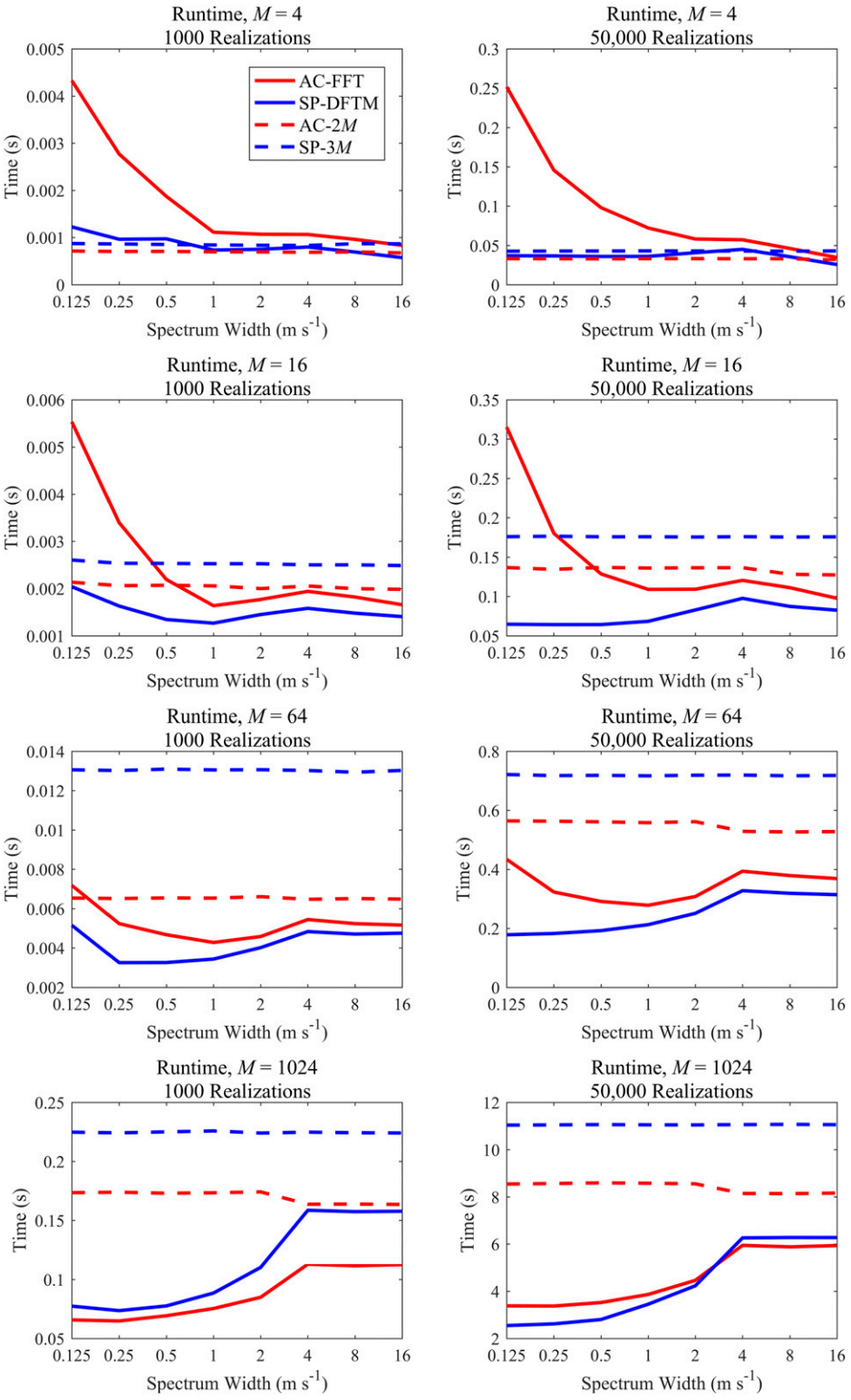

FIG. 12. Average runtimes for the AC-FFT, SP-DFTM, AC-2M, and SP-3M simulators. (left) Average runtime (s) for 1000 realizations, and (right) average runtime (s) for 50 000 realizations. The rows correspond to different values of M (4, 16, 64, and 1024).

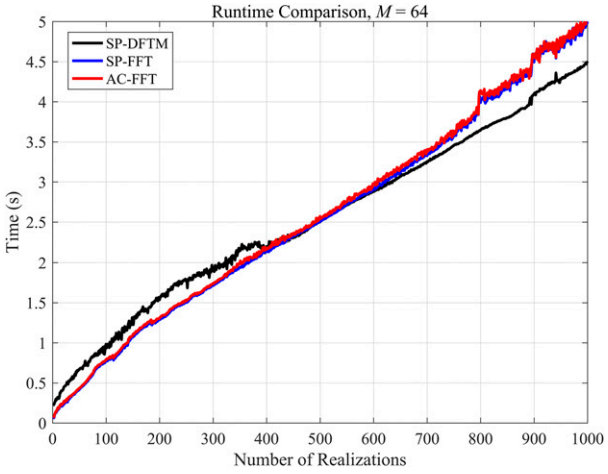

FIG. 13. Runtimes for varying numbers of realizations using 1000 different sets of parameters for the SP-DFTM, SP-FFT, and AC-FFT simulators.

AC-FFT simulator over the studied range. Based on these times, the SP-FFT simulator should be used when simulating small numbers of realizations, and the SP-DFTM simulator should be used when simulating larger numbers of realizations. The code for the simulators used in this section is available on the MATLAB File Exchange website under the name WR-TSS.

5. Additional considerations

In addition to modifying the simulators, there are a couple of other options that can improve the performance of time series simulations. The first is using single-precision values when possible. Single-precision floating point numbers are more than sufficient for most weather simulations and lead to a 20% performance improvement compared to double-precision floating point numbers in MATLAB. If a large number of realizations are needed, using a graphical processing unit (GPU) can also boost performance. Using single precision can make even more sense when using a GPU because many commercial GPUs that are not made for scientific applications have significantly faster single-precision performance but only marginally better double-precision performance.

Figure 14 shows the average runtimes and the autocorrelation and PSD errors for the SP-3*M*, AC-2*M*, and three versions of the SP-DFTM simulator. These three versions include the original double-precision version, a single-precision version, and a single-precision version that is optimized for a GPU. The 20% speedup from using single precision alone is apparent, but the biggest gains come from using the GPU. For these tests, a GeForce GTX TITAN Black GPU was utilized.

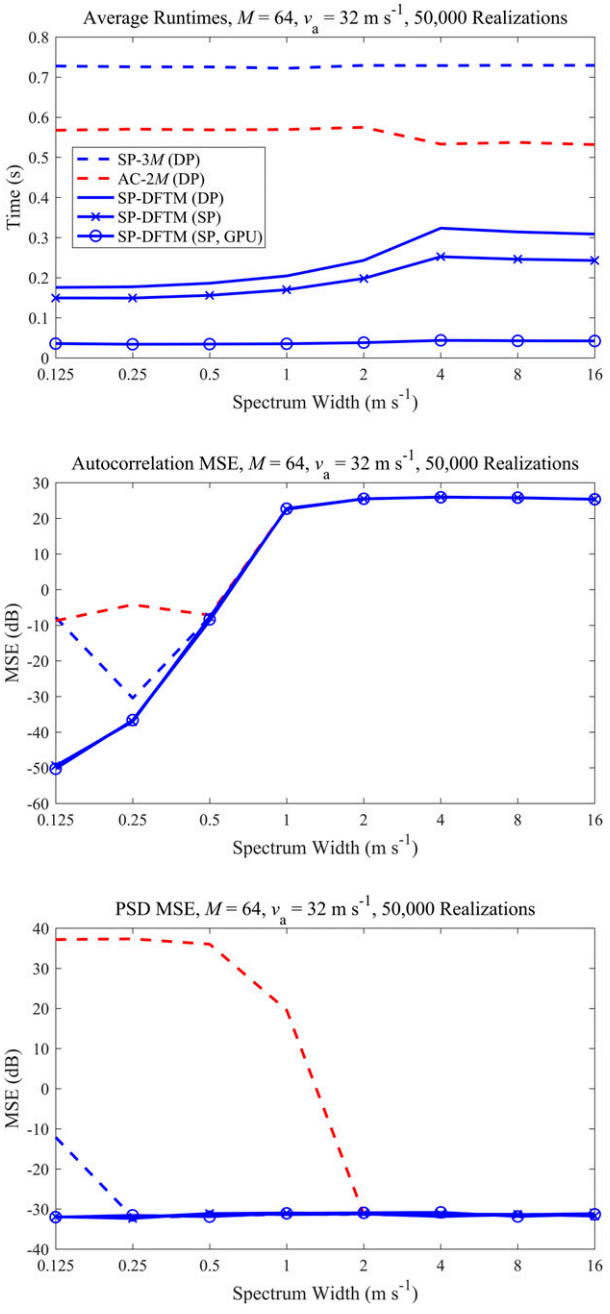

FIG. 14. (top) Average runtimes for the SP-3*M*, AC-2M, and three versions of the SP-DFTM simulator with $M = 64$ and 50 000 realizations. The (middle) autocorrelation and (bottom) PSD MSE for the same set of simulators.

The single-precision GPU version of SP-DFTM is 4–5 times faster than the single-precision version. If we go back and compare this to the original fixed-simulation-length versions, the single-precision GPU version is 12–16 times faster than AC-2*M* and 17–20 times faster than SP-3*M*. Of course, those simulators could also be sped up using single-precision GPU versions, but the SP-DFTM

simulator is accurate and faster already. The errors of all of the SP-DFTM simulators are nearly the same, which shows that using single precision or a GPU does not affect simulator accuracy. The accuracy was also checked over the same wide range of parameters shown in Figs. 11 and 12, and the simulators are just as accurate for all of the cases. The combination of the three modifications, single precision, and the GPU results in an accurate and fast simulator that can decrease simulation runtimes significantly. In some cases, a multiday simulation could run in a few hours.

Because of the data movement from the GPU, the GPU version is faster only for a large number of realizations. It is clearly significantly faster for 50 000 simulations. With this particular combination of CPU and GPU, the GPU version outperformed the non-GPU version when the number of realizations was greater than about 2000–3000 (depending on the spectrum width). Performance is system dependent, but a single-precision GPU version of the simulator may be the fastest way to produce accurate weather radar time series data for a large number of realizations. The code for the GPU versions of the SP-DFTM simulator is also available on the MATLAB File Exchange website under the name WR-TSS.

6. Conclusions

Three new modifications to the conventional weather radar time series simulators were introduced for improving accuracy and performance. Two of the modifications use a threshold, one in the autocorrelation domain and one in the spectral domain, to determine the simulation length and to limit the spectrum to only the values that are necessary to produce accurate simulations. The third modification uses a partial DFT matrix to calculate the inverse DFT instead of using an inverse FFT algorithm. These three modifications lead to accurate simulations for a wide variety of weather signal parameters, especially for narrow spectrum widths. In many cases, the performance is also significantly faster than traditional simulators with fixed simulation lengths. Using single precision is another way to speed up the simulators by about 20%, but using a GPU is the best way to significantly reduce simulation times. The single-precision simulators are still accurate, and using single precision and a GPU with the modified simulators results in fast and accurate simulators when simulating large numbers of realizations.

The previously described simulators were implemented assuming a Gaussian spectral model, but the same ideas could be applied to other spectral models. The key is determining the thresholds that are appropriate for the particular spectral model that balances accuracy and performance. For models that are similar to the Gaussian model, the recommended values of A_T and S_T would probably be a reasonable starting point. If the spectrum is not produced using a formula (e.g., using a collection of velocities to form the spectrum), it may be possible to estimate the spectrum width of the velocities and use that if the spectrum is close to Gaussian. This estimated spectrum width could be used in step 1 to determine the simulation length instead of the spectrum width from the Gaussian model.

There are surely additional modifications for improving performance while maintaining the accuracy of time series simulators, but the modifications introduced here are relatively simple and can be easily added to currently implemented versions without a large amount of additional complexity. One additional approach to improving performance that could be explored is producing more than one realization when the simulations length M_S is greater than $2M$. This approach would be applicable to the FFT versions of the simulators when only the first M values are currently returned. The decision about how independent these realizations need to be would factor into the possible amount of performance improvement. In general, it is important to preserve accuracy when simulating time series data because you never know when a new estimator or technique could expose the inaccuracies of a particular simulator.

Acknowledgments. I would like to thank my colleagues Sebastián Torres, Igor Ivić, and David Warde for providing suggestions and insights, which greatly improved this work. I also appreciated the comments and suggestions from the anonymous reviewers. Funding was provided by NOAA/Office of Oceanic and Atmospheric Research under NOAA–University of Oklahoma Cooperative Agreement NA16OAR4320115, U.S. Department of Commerce.

REFERENCES

Davies, R. B., and D. S. Harte, 1987: Tests for Hurst effect. *Biometrika*, **74**, 95–101, https://doi.org/10.1093/biomet/74.1.95.

Frehlich, R. G., and M. J. Yadlowsky, 1994: Performance of mean-frequency estimators for Doppler radar and lidar. *J. Atmos. Oceanic Technol.*, **11**, 1217–1230, https://doi.org/10.1175/1520-0426(1994)011<1217:POMFEF>2.0.CO;2.

Galati, G., and G. Pavan, 1995: Computer simulation of weather radar signals. *Simul. Pract. Theory*, **3**, 17–44, https://doi.org/10.1016/0928-4869(95)00009-I.

Janssen, L. H., and G. A. Van Der Spek, 1985: The shape of Doppler spectra from precipitation. *IEEE Trans. Aerosp.*

Electron. Syst., **AES-21**, 208–219, https://doi.org/10.1109/TAES.1985.310618.

Sachidananda, M., and D. S. Zrnić, 1999: Systematic phase codes for resolving range overlaid signals in a Doppler weather radar. *J. Atmos. Oceanic Technol.*, **16**, 1351–1363, https://doi.org/10.1175/1520-0426(1999)016<1351:SPCFRR>2.0.CO;2.

Thompson, R., 1973: Generation of stochastic processes with given spectrum. *Util. Math.*, **3**, 127–137.

Yu, T.-Y., R. R. Rondinel, and R. D. Palmer, 2009: Investigation of non-Gaussian Doppler spectra observed by weather radar in a tornadic supercell. *J. Atmos. Oceanic Technol.*, **26**, 444–461, https://doi.org/10.1175/2008JTECHA1124.1.

Zrnić, D. S., 1975: Simulation of weatherlike Doppler spectra and signals. *J. Appl. Meteor.*, **14**, 619–620, https://doi.org/10.1175/1520-0450(1975)014<0619:SOWDSA>2.0.CO;2.