Comparison between Weather Signal Simulation in Frequency and Time Domain

IGOR R. IVIĆ[a,b]

[a] *Cooperative Institute for Severe and High-Impact Weather Research and Operations, University of Oklahoma, Norman, Oklahoma*
[b] *NOAA/OAR/National Severe Storms Laboratory, Norman, Oklahoma*

(Manuscript received 20 June 2021, in final form 15 February 2022)

ABSTRACT: Simulated weather time series are often used in engineering and research practice to assess radar systems behavior and/or to evaluate the performance of novel techniques. There are two main approaches to simulating weather time series. One is based on summing individual returns from a large number of distributed weather particles to create a cumulative return. The other is aimed at creating simulated random signals based on the predetermined values of radar observables and is of interest herein. So far, several methods to simulate weather time series, using the latter approach, have been proposed. All of these methods are based on applying the inverse discrete Fourier transform to the spectral model with added random fluctuations. To meet the desired simulation accuracy, such an approach typically requires generating the number of samples that is larger than the base sample number due to the discrete Fourier transform properties. In that regard, a novel method to determine simulation length is proposed. It is based on a detailed theoretical development that demonstrates the exact source of errors incurred by this approach. Furthermore, a simple method for time series simulation that is based on the autocorrelation matrix exists. This method neither involves manipulations in the spectral domain nor requires generating the number of samples larger than the base sample number. Herein, this method is suggested for weather time series simulation and its accuracy and efficiency are analyzed and compared to the spectral-based approach.

SIGNIFICANCE STATEMENT: All research articles published so far on the topic of weather time series simulation propose the use of inverse discrete Fourier transform (IDFT) when based on the desired Doppler moment values. Herein, a detailed theoretical development that demonstrates the exact source of errors incurred by this approach is presented. Also, a novel method to determine the simulation length that is based on the theoretical error computation is proposed. As an alternative, a computationally efficient general method (not using IDFT) previously developed for the simulation of sequences with desired properties is suggested for weather time series simulation. It is demonstrated that the latter method produces accurate results within overall shorter computational times. Moreover, it is shown that the use of graphics processing unit (GPU), ubiquitous in modern computers, significantly reduces computational times compared to the sole use of central processing unit (CPU) for all simulation-related calculations.

KEYWORDS: Atmosphere; Radars/Radar observations; Weather radar signal processing; Fourier analysis; Time series

1. Introduction

Numerical simulations provide a deterministic and controlled environment in which algorithms can be developed and tested prior to their application on real data. For this reason, multiple publications on the topic of simulating weather time series have been released. The simulators can be classified into two categories. One type of simulators synthesize weather radar time series by adding amplitude and phase contributions from a multitude of single scatterers present in a virtual environment (Capsoni and D'Amico 1998; Capsoni et al. 2001; Cheong et al. 2008). The scatterers can be (but are not limited to) nonspherical raindrops of various sizes that are in motion due to wind as well as turbulence and can have various fall speeds. Given an accurate representation of the physical environment, such simulators can provide useful information on how properties such as drop size distribution, their shape, and motion, impact weather radar observables. In the case of single polarized weather radar, the observables are power (S), velocity (mean velocity, in a radial direction, of the precipitation either toward or away from the radar denoted as v), and spectrum width (a measure of the velocity estimates variability due to wind shear, turbulence, and/or the quality of the velocity samples denoted as σ_v) that are referred to as the Doppler spectral moments (Doviak and Zrnić 1993; Bringi and Chandrasekar 2001). In polarimetric radars, three additional observables (known as polarimetric variables) are (Doviak and Zrnić 1993; Bringi and Chandrasekar 2001) differential reflectivity (logarithm of the horizontal to vertical powers ratio of received signals denoted as Z_{DR}), copolar correlation coefficient (the correlation coefficient between horizontal and vertical returns denoted as ρ_{hv}), and differential phase (phase difference between the horizontal and vertical returns denoted as ϕ_{DP}). Typically, such simulators are computationally extremely intensive due to a huge number of particles needed for a realistic representation of a virtual weather environment. Of interest herein are another type of statistically based simulators that generate random sequences (i.e., time series), which follow the statistical properties of observed phenomena described via predetermined values of Doppler spectral moments (Thompson 1973; Zrnić 1975; Sirmans and Bumgarner 1975; Frehlich and Yadlowsky 1994; Galati and Pavan 1995; Curtis 2018). Given a set of

Corresponding author: Igor R. Ivić, igor.ivic@noaa.gov

DOI: 10.1175/JTECH-D-21-0082.1

desired polarimetric variable values, two sets of time series may be generated using Doppler moment values and combined to produce simulated polarimetric time series with desired statistical properties (Galati and Pavan 1995). Because these simulators produce time series with desired Doppler moment and polarimetric variable values (estimates of which are the base radar products) they are well suited for conducting Monte Carlo simulations to determine statistical properties of estimates (e.g., bias and standard deviation) produced by weather radar systems. Such analysis is useful because it provides an avenue for statistical evaluation of signal processing techniques (Torres and Zrnić 2003; Cho 2005; Yu et al. 2006; Curtis and Torres 2011; Lei et al. 2012; Ivić 2014, 2019) as well as an assessment of system effects on weather radar measurements (Ivić 2017; Schvartzman and Curtis 2019).

Two more inputs needed for simulation are the unambiguous velocity (v_a) (Doviak and Zrnić 1993; Bringi and Chandrasekar 2001), and the number of samples (M) that correspond to the number of pulses transmitted during the dwell time for a pulsed-Doppler radar. Typically, the process of time series simulation starts by creating the signal power spectral density (PSD). For weather signals, the Gaussian spectral model is used. Next, the random components are added to the PSD, and the inverse discrete Fourier transform (IDFT) is applied to produce the time series with desired properties. Due to the circular convolution properties of the IDFT, this process typically requires the number of simulated samples (M_S) to be larger than M (where the excess samples are discarded). Consequently, the generation of an excess number of samples increases the simulation computational intensity and duration. An approximate method to determine M_S is proposed in Curtis (2018). In addition, a novel more accurate method to determine M_S is developed in this work.

In contrast to the IDFT process, a method that produces random sequences based on the desired autocorrelation matrix has been proposed by Johnson (1994). It does not require the use of IDFT and therefore generation of only M samples is needed to accurately simulate time series. The use of this method for the simulation of weather time series is investigated in this paper.

The paper is structured as follows. The theoretical background for the IDFT and autocorrelation matrix–based simulation methods is presented in section 2. It is followed by the comparison of simulation methods in terms of accuracy (section 3) and efficiency (section 4). The summary of the results is presented in section 5.

2. Theoretical background

The procedure for simulating the time series mth sample using the IDFT-based approach (herein denoted as SP) is

$$\mathrm{IQ}(m) = \frac{1}{M_S}\sum_{k=0}^{M_S-1}\sqrt{S(k)}N(k)e^{j2\pi mk/M_S}, \tag{1}$$

where $S(k)$ is the desired PSD, and $N(k)$ are the complex unit power Gaussian white noise draws [i.e., real and imaginary parts of each $N(k)$ are independent realizations of a Gaussian random variable where the ensemble average of $|N(k)|^2$ is one]. A typical procedure for creating $S(k)$ is to generate discrete samples of the desired PSD at integer multiples of $2v_a/M_S$ across the extended Nyquist cointerval (e.g., $-nv_a$ to nv_a where n is usually 3 or 4; Curtis 2018). Then alias the extended PSD over the desired Nyquist cointerval (from $-v_a$ to v_a), and rearrange it (to satisfy IDFT requirements) to produce $S(k)$. Given the desired signal power S, $S(k)$ samples are scaled so that

$$S = \langle|\mathrm{IQ}(m)|^2\rangle = \frac{1}{M_S^2}\sum_{k=0}^{M_S-1}S(k), \tag{2}$$

where $\langle\cdot\rangle$ denotes mathematical expectation. Using (1), the lag-l autocorrelation ensemble average $[R(l)]$ of the so obtained samples is

$$R(l) = \langle\mathrm{IQ}^*(m)\mathrm{IQ}(m+l)\rangle = \frac{1}{M_S^2}\sum_{k=0}^{M_S-1}S(k)e^{j2\pi kl/M_S}, \tag{3}$$

where the asterisk denotes complex conjugation. Using the discrete time Fourier transform (DTFT), the PSD may be expressed as a function of the desired autocorrelation coefficient $\rho(l)$ at lag l (i.e., autocorrelation normalized by its value at lag 0) as (appendix A)

$$S(k) = SM_S\sum_{n=-\infty}^{\infty}\rho(n)e^{-j2\pi kn/M_S}\Big/\sum_{p=-\infty}^{\infty}\rho(pM_S). \tag{4}$$

Using (3) and (4), the ensemble average of the lag-l autocorrelation from simulated samples is

$$\langle\mathrm{IQ}^*(m)\mathrm{IQ}(m+l)\rangle = \frac{S}{M_S}\frac{\sum_{n=-\infty}^{\infty}\rho(n)\sum_{k=0}^{M_S-1}e^{j2\pi k(l-n)/M_S}}{\sum_{p=-\infty}^{\infty}\rho(pM_S)}. \tag{5}$$

Because

$$\sum_{k=0}^{M_S-1}e^{j2\pi k(l-n)/M_S} = \begin{cases}0, & \text{if } l-n\neq pM_S\\ M_S, & \text{if } l-n=pM_S\end{cases}, \tag{6}$$

where p is an integer ranging from $-\infty$ to ∞, it follows that

$$\begin{aligned}\langle\mathrm{IQ}^*(m)\mathrm{IQ}(m+l)\rangle &= S\frac{\sum_{p=-\infty}^{\infty}\rho(l-pM_S)}{\sum_{p=-\infty}^{\infty}\rho(pM_S)}\\ &= \frac{S}{\sum_{p=-\infty}^{\infty}\rho(pM_S)}\Bigg[\rho(l)+\sum_{p=-\infty}^{-1}\rho(l-pM_S)\\ &\quad+\sum_{p=1}^{\infty}\rho(l-pM_S)\Bigg].\end{aligned} \tag{7}$$

The expression (7) shows that the autocorrelation ensemble average of the simulated samples at lag l is the infinite sum of

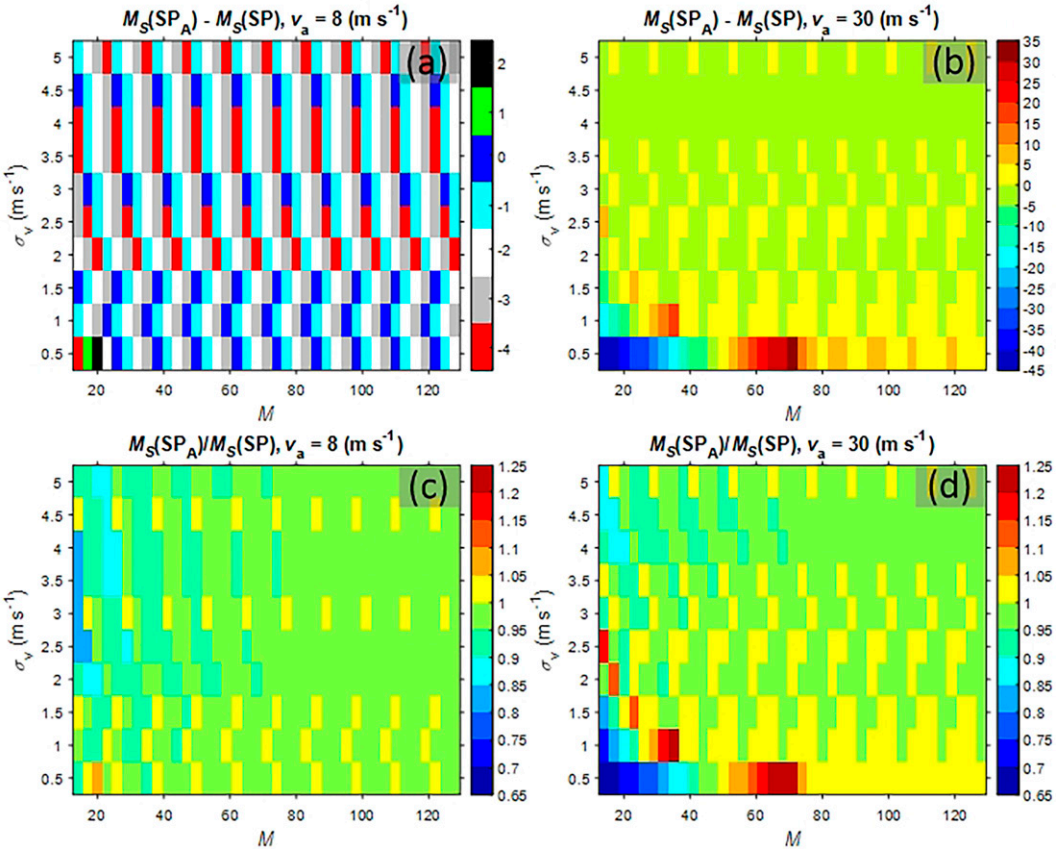

FIG. 1. Comparison of the simulation lengths between the SP and SP_A methods. (top) Difference in simulation lengths between SP_A and SP methods for (a) $v_a = 8$ m s^{-1} and (b) $v_a = 30$ m s^{-1}. Positive values denote settings where the SP_A method uses larger M_S values than the SP method and vice versa. (bottom) Ratio of SP_A vs SP simulation lengths for (c) $v_a = 8$ m s^{-1} and (d) $v_a = 30$ m s^{-1}. Values larger than one denote settings where the SP_A method uses larger M_S values than the SP method and vice versa.

the true autocorrelation function (ACF) values at lags spaced by M_S. Thus, if the desired number of simulated samples is M, then the simulation length M_S should be found so that

$$\langle \mathrm{IQ}^*(m)\mathrm{IQ}(m+l)\rangle \approx S\rho(l) \tag{8}$$

with the desired accuracy for lags $-M+1$ to $M-1$. In the case of the Gaussian ACF where

$$\rho(l) = \exp\left[-\frac{1}{2}\left(\frac{\pi\sigma_v l}{v_a}\right)^2\right]\exp\left(-j\pi l\,\frac{v}{v_a}\right), \tag{9}$$

a prescription to determine M_S is described in Curtis (2018). It is based on an approximate approach [i.e., it does not use the exact expression given in (7)] whereby the simulation length is determined by setting M_S so that the $|\rho(M_S)|$ is less than or equal to a predefined threshold (e.g., −25 dB). For Gaussian ACF, this approach is formalized as (Curtis 2018)

$$M_S = \max(2k+1, M+k), \tag{10}$$

where

$$k = \left\lceil \frac{v_a}{\pi\sigma_v}\sqrt{-\frac{\ln(10)}{5}A_T}\right\rceil, \tag{11}$$

and "ln" is the natural logarithm, the symbol $\lceil\cdot\rceil$ denotes rounding toward the next higher integer, while A_T is the autocorrelation threshold [e.g., $A_T = -25$ dB as suggested in Curtis (2018)]. For instance, if $M = 16$, $v_a = 30$ m s^{-1}, $\sigma_v = 0.5$ m s^{-1}, and $A_T = -25$ dB (e.g., narrow σ_v is often used for ground clutter simulation), this method computes $M_S = 132$. However, if $\sigma_v = 4$ m s^{-1} and all other parameters remain the same, the suggested M_S value is 28.

Contrary to the approximate approach in Curtis (2018), given the maximum allowable autocorrelation magnitude error at lag $M-1$ (since the largest error occurs at this lag), an exact method for determining M_S can be derived using (7). First, let us note that for a simulation length M_S the largest value that aliases in the ACF estimate (from simulated time series) at lag $M-1$ is $\rho(M-1-M_S)$ [as indicated by the numerator in (7)]. Hence, the minimum M_S value may be set by

first finding lag l for which $|\rho(l)|$ is smaller than or equal to a predefined threshold A_T below which autocorrelation coefficient is considered insignificant (e.g., −30 dB which corresponds to ±0.001). If $l < M$, then M_S which satisfies the following

$$\left|\frac{\sum_{p=-\infty}^{\infty}\rho(M-1-pM_S)}{\sum_{p=-\infty}^{\infty}\rho(pM_S)}\right| \le 10^{A_T/10} \tag{12}$$

may be used for simulation length. This ensures that the autocorrelation coefficient with the largest error is below the threshold A_T. If $l \ge M$, then M_S must be found so that the relative ACF error (in dB) is below the predetermined threshold as

$$\text{ACF\_err\_dB}(M-1) = 10\log_{10}\left[\left|\frac{\langle \text{IQ}^*(m)\text{IQ}(m+M-1)\rangle}{S\rho(M-1)} - 1\right|\right] \le A_T. \tag{13}$$

Using (7), the expression (13) can be written as

$$\text{ACF\_err\_dB}(M-1) = 10\log_{10}\left[\lim_{P\to\infty}\left|\frac{\sum_{p=-P}^{P}\rho(M-1-pM_S)}{\rho(M-1)\sum_{p=-P}^{P}\rho(pM_S)} - 1\right|\right] \le A_T. \tag{14}$$

Then M_S can be found by computing the left side of (14) for a set of M_S values and choosing the one that satisfies the inequality. For $A_T = -30/-25$ dB (results in autocorrelation coefficient error within ±0.001/±0.0032) and the same parameter values, as stated in the previous paragraph, this method computed M_S values of 88/82 and 24/24 for σ_v of 0.5 m s^{-1} and 4 m s^{-1}. This suggests that the SP method can be further optimized using the M_S calculation method proposed herein (hereon denoted as SP$_A$).

The absolute differences between the simulation lengths of the two methods are presented in Figs. 1a and 1b. To provide a better understanding of the relative change in the simulation lengths, the ratios of the M_S values for the two methods are presented in Figs. 1c and 1d. These are given for a range of M and σ_v values, as well as for unambiguous velocities of $v_a = 8$ m s^{-1} (Figs. 1a,c) and $v_a = 30$ m s^{-1} (Figs. 1b,d). These indicate that, in most cases, the SP$_A$ method results in either the same or slightly shorter simulation lengths when $v_a = 8$ m s^{-1}. In the case of $v_a = 30$ m s^{-1}, the SP$_A$ method results in appreciably shorter simulation lengths for narrow spectrum widths and smaller M (i.e., $\sigma_v < \sim 1$ m s^{-1} and $M < \sim 40$). At the same time, the SP$_A$ method results in longer simulation lengths when M is around 60 and $\sigma_v = 0.5$ m s^{-1} indicating that the SP method may be producing time series with ACF errors larger than the SP$_A$ method for those simulation parameters.

Additional optimization can be introduced by noting that the DFT is typically computed using the fast Fourier transform (FFT) algorithms (Cooley and Tukey 1965). These algorithms apply DFT (and IDFT) computations by combining the DFT operations of smaller sizes (than the desired DFT/IDFT) to reduce the total number of computations. The result is that a DFT/IDFT of a certain length may execute faster than another DFT/IDFT of a shorter length. In addition, the practical system dependent aspects of DFT/IDFT computations may also result in execution times that behave nonlinearly with DFT/IDFT length. To exemplify this, the SP$_A$ method execution times were measured for M_S values ranging from 10 to 512 (in steps of 1) on the system with Intel i9-9900K central processing unit (CPU) paired with Nvidia GTX 1080Ti graphic processing unit (GPU) using the MATLAB software package. The number of realizations (Q) was 5×10^5. The execution times were measured when the SP$_A$ method was executed on the CPU only and when the IDFT computations were offloaded to GPU. The results for the first case are shown in Fig. 2a, and in Fig. 2b for the second case. The plot in Fig. 2a indicates that the SP$_A$ method execution times exhibit a general linear increase with

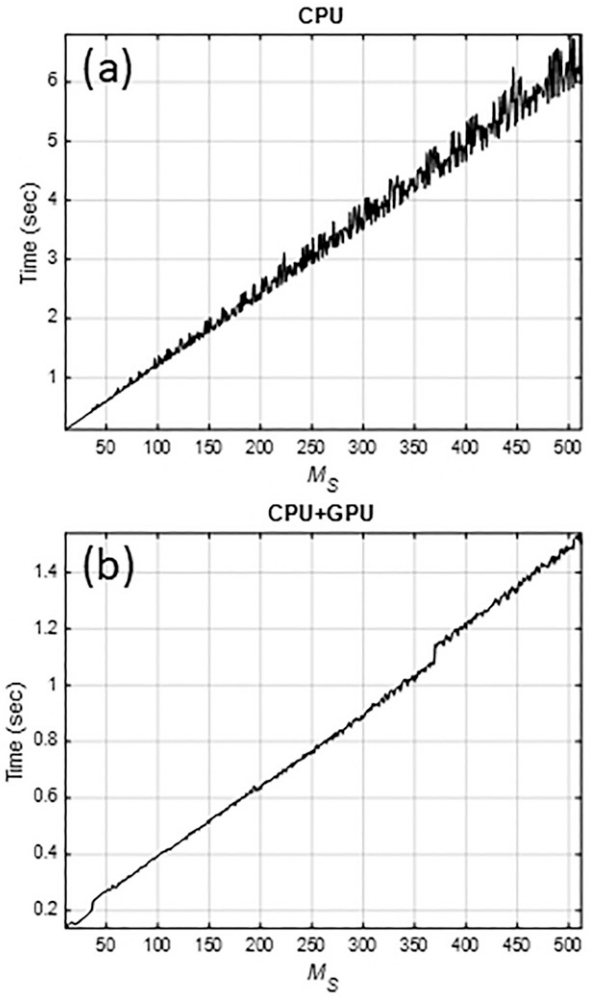

FIG. 2. The SP$_A$ method execution times for M_S values ranging from 10 to 512 in steps of 1 and the realizations number of 5×10^5. (a) The CPU performs all computations and (b) IDFT computations are performed via GPU.

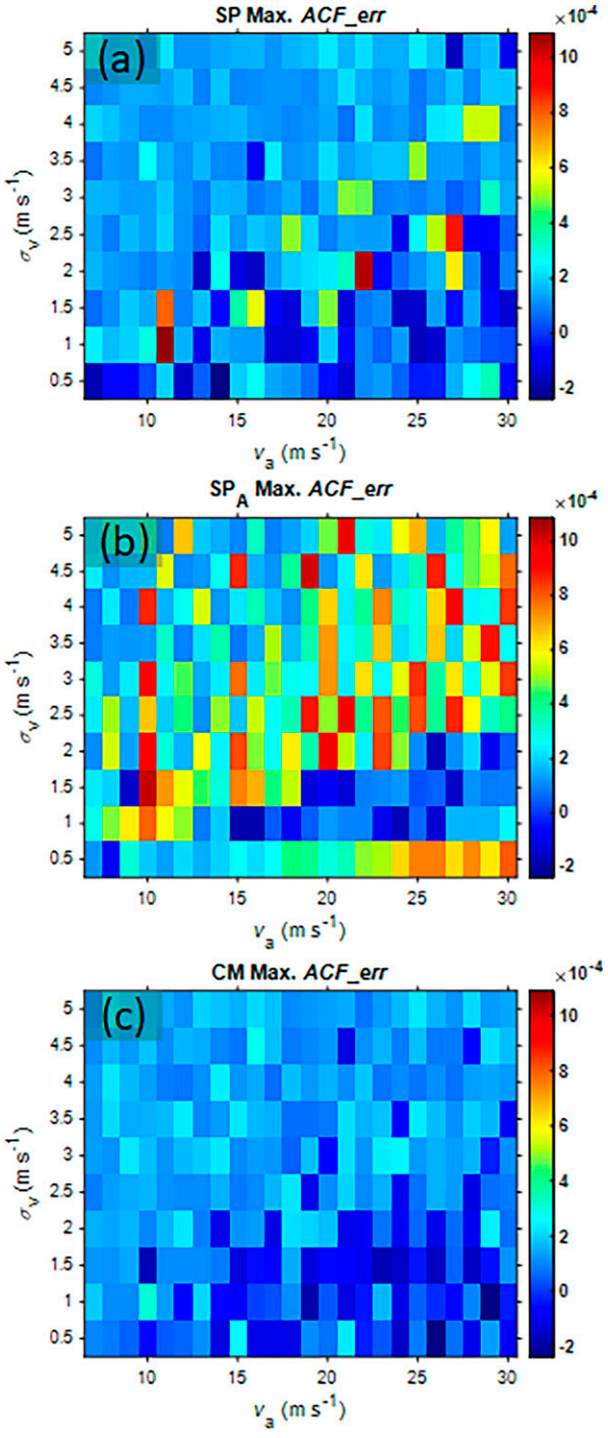

FIG. 3. The maximum ACF errors for (a) SP, (b) SP$_A$, and (c) CM methods. Color-bar values are dimensionless.

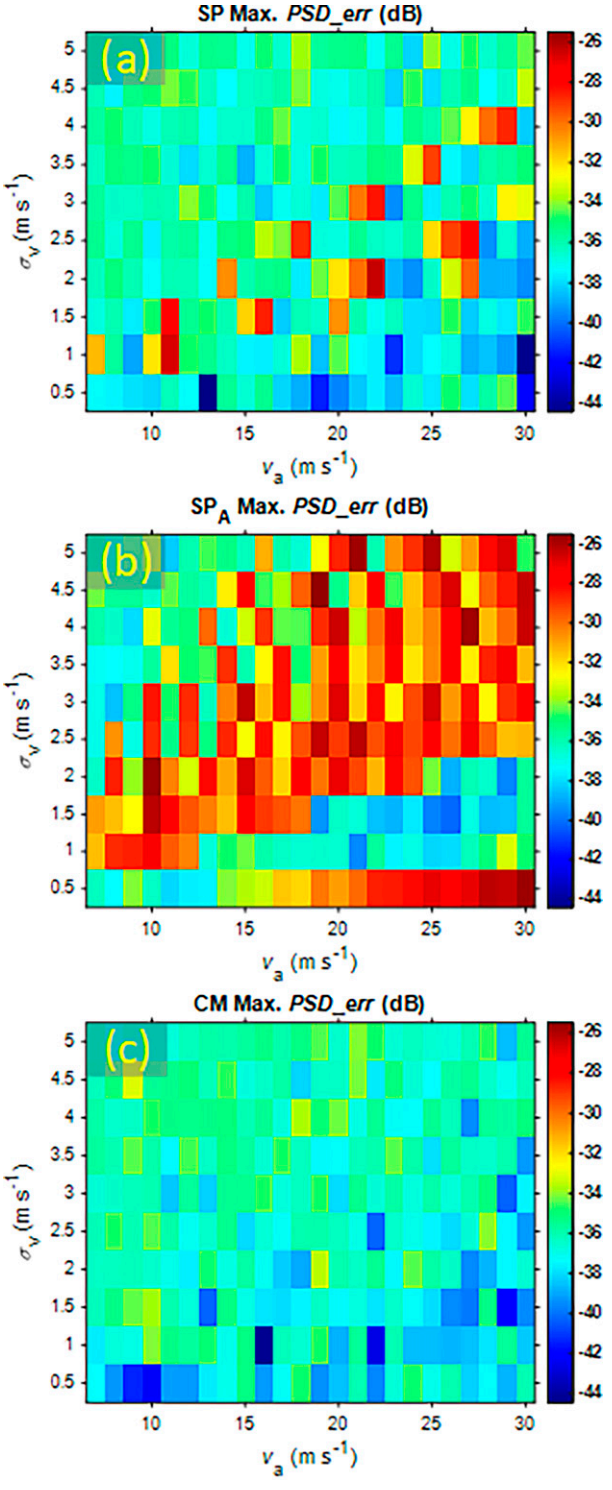

FIG. 4. The maximum PSD errors for (a) SP, (b) SP$_A$, and (c) CM methods. Color-bar values are in dB.

M_S but fluctuate appreciably for this particular system. Consequently, increasing M_S to a slightly larger value [e.g., larger than the one determined using (12) and (14)] may improve the efficiency if only CPU is used for simulation computations. Furthermore, the results in Fig. 2a suggest that this approach may produce increasing gains in efficiency as M_S becomes larger. The plot in Fig. 2b, however, shows the steady increase in computational time with M_S but very small fluctuations. This suggests that increasing M_S would not be beneficial when GPU is

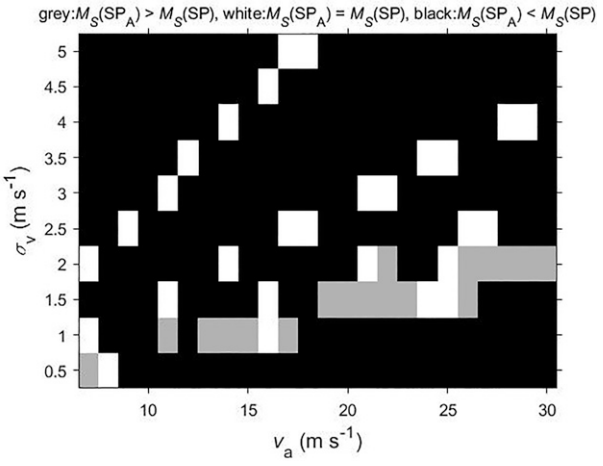

FIG. 5. Areas where the SP_A method uses M_S larger than [i.e., $M_S(SP_A) > M_S(SP)$ in gray], equal to [i.e., $M_S(SP_A) = M_S(SP)$ in white], and smaller than [i.e., $M_S(SP_A) < M_S(SP)$ in black] the SP method.

used for IDFT calculations. Consequently, the SP_A method is modified so that the M_S [determined using (12) and (14)] is increased to the next larger value that yields better efficiency as per the results in Fig. 2a (or it is left unchanged if such value does not exist) when CPU is used for IDFT computations.

The complication of determining the simulation length M_S larger than M can be avoided by following the simulation procedure proposed in Johnson (1994). To understand the logic behind this method, let us assume that the vector of M random Gaussian distributed samples **iq** with desired ACF can be produced as

$$\mathbf{iq} = \mathbf{Hn}, \tag{15}$$

where **H** is the $M \times M$ matrix of complex numbers and **n** is the vector of unit power Gaussian white noise draws. Then

$$\langle \mathbf{iq} \times \mathbf{iq}^{\mathrm{H}} \rangle = \mathbf{C} = \mathbf{H}\langle \mathbf{n} \times \mathbf{n}^{\mathrm{H}} \rangle \mathbf{H}^{\mathrm{H}} = \langle \mathbf{HH}^{\mathrm{H}} \rangle, \tag{16}$$

since

$$\langle \mathbf{n} \times \mathbf{n}^{\mathrm{H}} \rangle = \mathbf{I}. \tag{17}$$

In (16), and (17), **C** is the $M \times M$ autocorrelation matrix, the superscript H denotes the Hermitian transpose, and **I** is the identity matrix. The expression (16) shows that the Gaussian random samples with the desired ACF can be generated by decomposing **C** (e.g., using eigenvalue or LDL decomposition) and multiplying the result with the vector of independent white noise samples (hereon denoted as the CM method). In the case of weather samples, the elements of **C** can be generated using (9). If simulation of time series with non-Gaussian PSD is needed, the desired PSD can be created in the spectral domain and the corresponding ACF may be calculated using an inverse Fourier transform.

The noise samples of the desired power can be added to the time series created via the SP, SP_A, or CM method to account for the finite signal-to-noise ratio (SNR) for accurate simulation of the system effects.

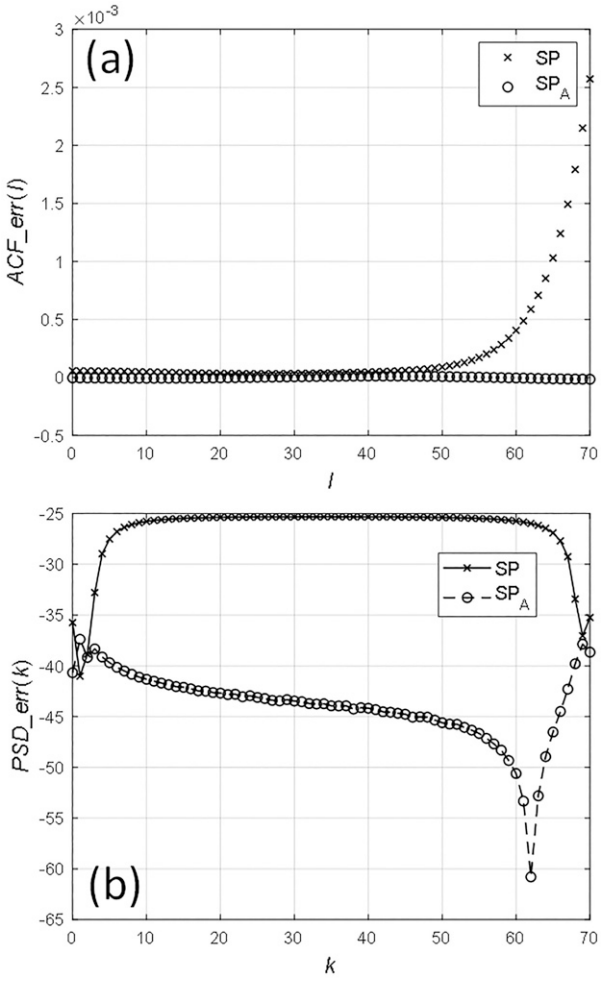

FIG. 6. (a) ACF and (b) PSD errors for the SP and SP_A method for $M = 71$, $v_a = 30$ m s^{-1}, and $\sigma_v = 0.5$ m s^{-1}.

3. Accuracy analysis

The SP [using M_S calculation described in Curtis (2018)], as well as SP_A and CM simulation methods were implemented using the MATLAB software package whereby the simulated samples were produced with single floating-point precision. The accuracy assessment is conducted by comparing the absolute value of the normalized ACF (i.e., autocorrelation coefficient) produced from the simulated time series to its theoretical counterpart in (9). The difference between the two is computed as

$$\mathrm{ACF\_err}(l) = \left| \frac{1}{Q(M-l)} \sum_{q=0}^{Q-1} \sum_{m=0}^{M-1-l} \mathrm{IQ}^{*}(m,q)\mathrm{IQ}(m+l,q) \right| \Big/ S - |\rho(l)|. \tag{18}$$

The quantity in (18) is computed for $M = 16$, and a range of v_a, as well as σ_v values. The number of realizations Q is 6×10^7. The autocorrelation threshold A_T is set to -25 dB for the SP method as this value was suggested in Curtis (2018). In the case of SP_A method, the threshold value is set to -30 dB since it limits

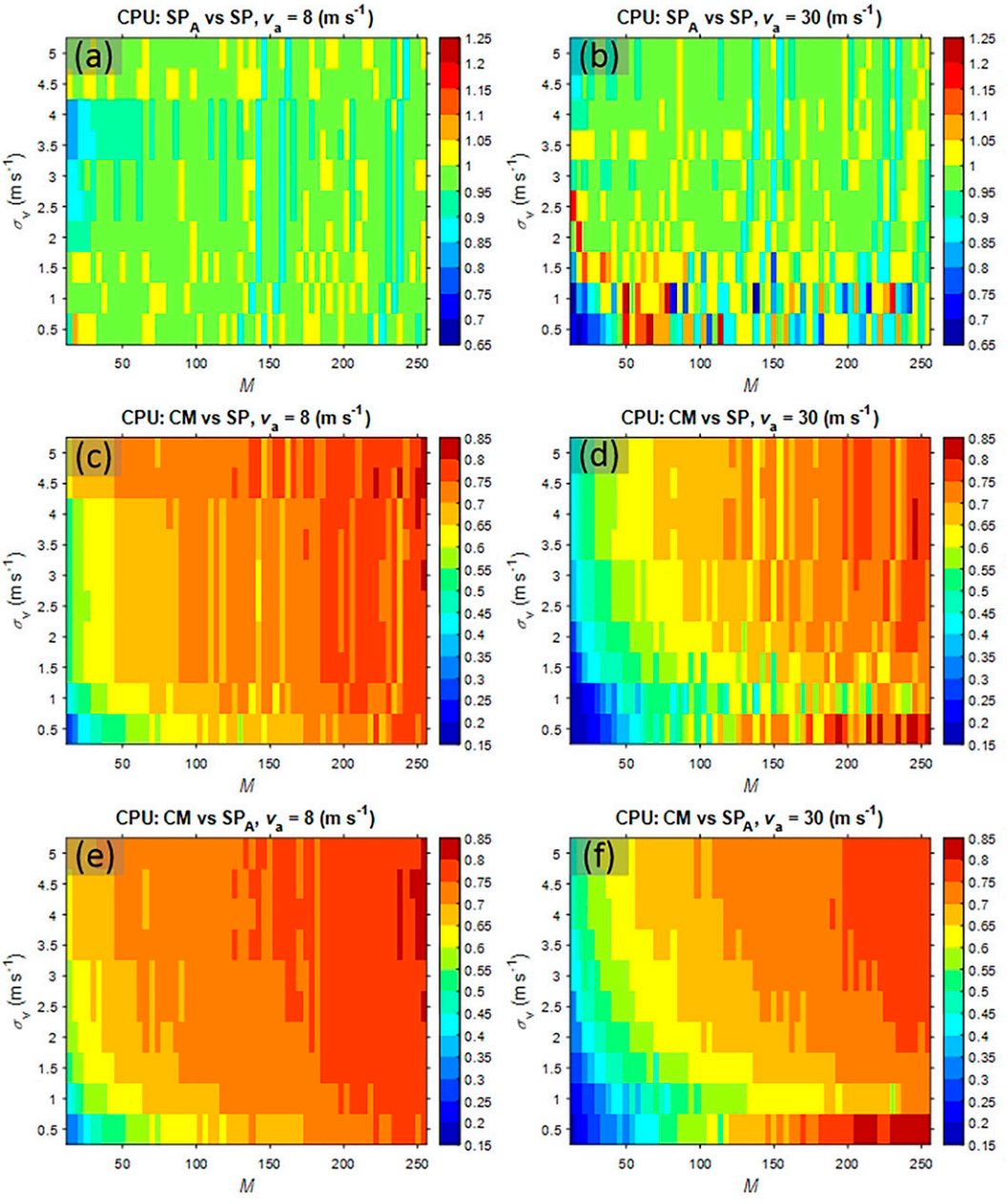

FIG. 7. The ratios of CM, SP, and SP$_A$ method execution times to simulate 5×10^5 realizations using CPU-only implementation on the system with Intel i9-9900K CPU. (a),(c),(e) v_a = 8 m s^{-1} and (b),(d),(f) v_a = 30 m s^{-1}. Color-bar values are dimensionless where values larger than 1 denote settings for which the first method in the subfigure title takes longer to execute than the second method and vice versa.

the normalized ACF error to ±0.001 which is deemed insignificant (i.e., it produces more accurate results than using A_T of −25 dB which yields errors within ±0.0032). For each set of simulation parameters, the maximum error was estimated and the results are shown in Fig. 3. In the case of the SP and SP$_A$ methods, the estimated errors are approximately between 0.0011 and −0.000 24 as well as 0.001 and −0.000 19, respectively. Visual comparison of Figs. 3a and 3b indicates that the SP$_A$ method produces more estimates with errors that are closer to the upper error limit of ~0.001 than the SP method. In the case of the CM method, the estimated errors are approximately between 0.0003 and −0.000 24. Thus, the CM method overall produces estimates with errors smaller than the other two methods. Given that the autocorrelation coefficient spans values between 1 and 0 where the values around 0.001 and below may be considered insignificant, the ACF errors produced by all three methods are random and are of levels that are not appreciable.

The next accuracy assessment is conducted by comparing the periodogram estimated from the simulated time series to the corresponding theoretical ensemble average (appendix B) for the same parameters as for the ACF accuracy analysis. This is done by computing the following:

$$\mathrm{PSD\_err}(k) = 10\log_{10}\left|\left|\frac{\hat{S}_P(k)}{\langle\hat{S}(k)\rangle} - 1\right|\right|, \tag{19}$$

where

$$\hat{S}_P(k) = \frac{1}{Q}\sum_{q=0}^{Q-1}\left|\sum_{m=0}^{M-1}\mathrm{IQ}(m.q)\exp\left(-j2\pi\frac{mk}{M}\right)\right|^2, \tag{20}$$

and

$$\langle\hat{S}(k)\rangle = 2S\sum_{l=1}^{M-1}\left\{e^{-(1/2)(\pi\sigma_v l/v_a)^2}\cos\left[\pi l\left(\frac{v}{v_a}+2\frac{k}{M}\right)\right](M-l)\right\} + MS. \tag{21}$$

Note that in (21) the contribution of noise power is omitted since the noise effects are not included in the simulated time series in this particular case. This is because the noise would obscure the PSD coefficients with powers close to or below the noise level rendering the accuracy analysis incomplete. The maximum PSD errors, as estimated by (19), are presented in Fig. 4. The overall maximum PSD errors (i.e., the maximum errors across all examined v_a and σ_v values in Fig. 4) are −26.3, −25.5, and −33 dB for the SP, SP$_A$, and CM methods, respectively. As in the case of ACF errors, a visual comparison between Figs. 4a and 4b indicates that the SP$_A$ method produces overall more estimates with PSD errors that are closer to the upper error limit than the SP method. This is the consequence of the more accurate ACF error control employed by the SP$_A$ method, whereby the SP$_A$ method results in shorter simulation lengths than the SP method while maintaining the ACF errors within the prescribed limits. This is exemplified in Fig. 5 which shows areas where the SP$_A$ method uses M_S larger [i.e., M_S(SP$_A$) > M_S(SP) in gray], equal [i.e., M_S(SP$_A$) = M_S(SP) in white], and smaller [i.e., M_S(SP$_A$) < M_S(SP) in black] than the SP method. Figure 5 shows that the SP$_A$ method chooses smaller M_S than the SP method in the majority of cases. Visual inspection of Fig. 4c suggests that the CM method produces the smallest overall errors. For the investigated settings, all three methods produce small random errors that are below −25 dB and may be considered insignificant for practical purposes.

The improved accuracy of the SP$_A$ method simulated time series is further demonstrated for settings with M = 71, v_a = 30 m s^{-1}, and σ_v = 0.5 m s^{-1}. For these settings, the SP and SP$_A$ methods determine the simulation lengths M_S of 136 and 170. The corresponding ACF and PSD errors are shown in Fig. 6. The results in Fig. 6a show that the SP method ACF errors start to rapidly increase for lags larger than 50 and eventually surpass the 0.001 value. The SP$_A$ method ACF errors, however, remain insignificant for all lags of interest. In the case of PSD errors (Fig. 6b), the SP$_A$ method results in significantly smaller errors than the SP method even though the latter method errors remain below −25 dB, which may be considered insignificant. This attests to the more precise and stable error control of the SP$_A$ method (compared to the SP method). As demonstrated, the improved error control of the SP$_A$ method may, in some cases, result in significantly longer simulation lengths than the SP method (which may result in appreciably longer execution times of the SP$_A$ method compared to the SP method).

TABLE 1. Total execution times to produce the simulation results.

v_a (m s^{-1})	System	SP (min)	SP$_A$ (min)	CM (min)
8	CPU	169.5	165.3	123.2
	CPU + GPU	38.5	38.5	31.5
30	CPU	183.3	177.9	123.7
	CPU + GPU	40.3	40.5	31.6

4. Efficiency analysis

The efficiency analysis is carried out by computing the ratio of CM, SP, and SP$_A$ method execution times for simulations that produce 5×10^5 realizations for a range of M (from 14 to 254 with the step of 4) and σ_v (from 0.5 to 5 m s^{-1} with the step of 0.5 m s^{-1}) values. For each set of M and σ_v values, the execution times are computed as an average of 10 runs. For this, the system with Intel i9-9900K CPU paired with Nvidia GTX 1080Ti GPU was used. All simulation methods were programmed to use either CPU only or CPU and GPU units. In the latter case, the IDFT, as well as the **H** matrix multiplication with **n** vector, were ported to the GPU for the three methods (using MATLAB built-in facilities). The ratios of processing times for the two implementations were computed for v_a values of 8 and 30 m s^{-1}. The first value is typical for surveillance and the second for Doppler scans.

The results for CPU-only implementation are presented in Fig. 7. For this implementation, the average execution time ratio SP$_A$/SP is 0.97 for v_a = 8 m s^{-1} (Fig. 7a) and 0.98 for v_a = 30 m s^{-1} (Fig. 7b). The more careful examination of Figs. 7a and 7b indicates that the SP$_A$ method performs faster than the SP method for M < ~30 and 2.5 < σ_v < ~4.5 m s^{-1} when v_a = 8 m s^{-1} as well as for M < ~20 and $\sigma_v \leq$ ~1 m s^{-1} when v_a = 30 m s^{-1}. In general, Figs. 7a and 7b indicate that the SP$_A$ method performs faster or slower than the SP method for some M and σ_v values but mostly similar for the majority of other values. This can be attributed to the variable hardware behavior, the fluctuations in the execution time estimates as well as to the fact that the SP$_A$ method provides better control of the simulation errors than the SP method (i.e., the SP$_A$ method may use significantly different M_S than the SP method). Thus, the settings where the SP method performs faster than the SP$_A$ method may indicate that it produces simulated time series with larger errors than the latter method (i.e., the SP$_A$ method may result in the larger M_S to keep the simulation errors within the prescribed limits as shown in the example related to Fig. 6). Figures 7c–f show that the CM method performs faster than both the SP and SP$_A$ methods whereby the performance improvement declines as M increases. The total execution times to produce the simulation results are given in Table 1. These show that while the total

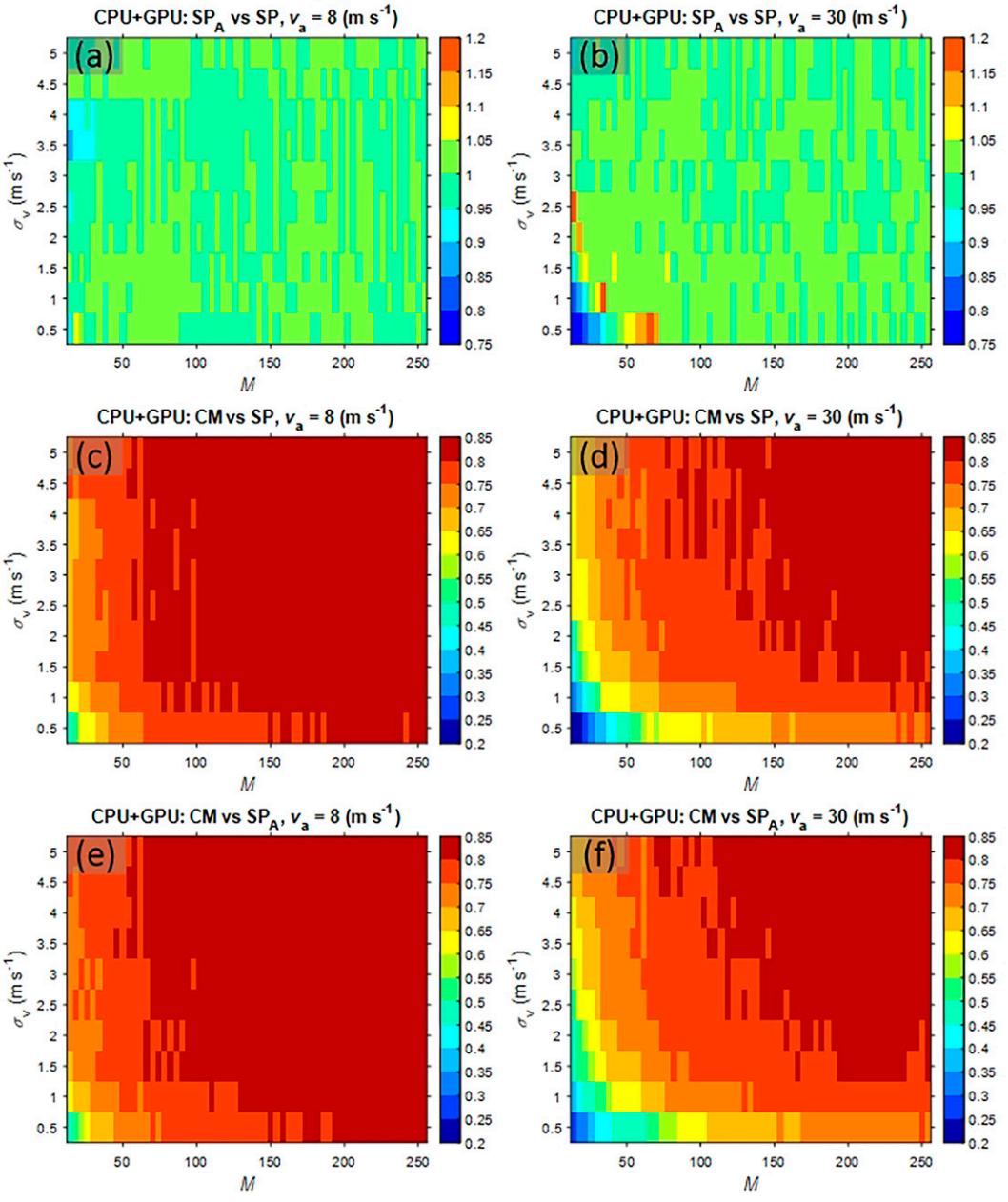

FIG. 8. The ratios of CM, SP, and SP_A method execution times to simulate 5×10^5 realizations using CPU and GPU implementation on the system with Intel i9-9900K CPU and Nvidia GTX 1080Ti GPU. (a),(c),(e) $v_a = 8$ m s^{-1}, and (b),(d),(f) $v_a = 30$ m s^{-1}. Color-bar values are dimensionless where values larger than 1 denote settings for which the first method in the subfigure title takes longer to execute than the second method and vice versa.

simulation times for the SP and SP_A methods are comparable, the CM method produced the results in appreciably less time than the other two methods.

In Fig. 8, the results in the case of CPU and GPU implementation are presented. These indicate similar efficiency results in the case of SP versus SP_A method comparison. As in the CPU-only implementation, the CM method is always executed faster than the SP and SP_A methods with the improvement tapering off as M increases. The total execution times to produce the results are given in Table 1 and show huge improvements in efficiency for all methods when GPU is used. In this case, the CM method produced the results in ~18% and ~22% less time for v_a of 8 and 30 m s^{-1} than the other two methods.

Unlike accuracy, the execution times of the investigated methods are system dependent. This is exemplified in Fig. 9.

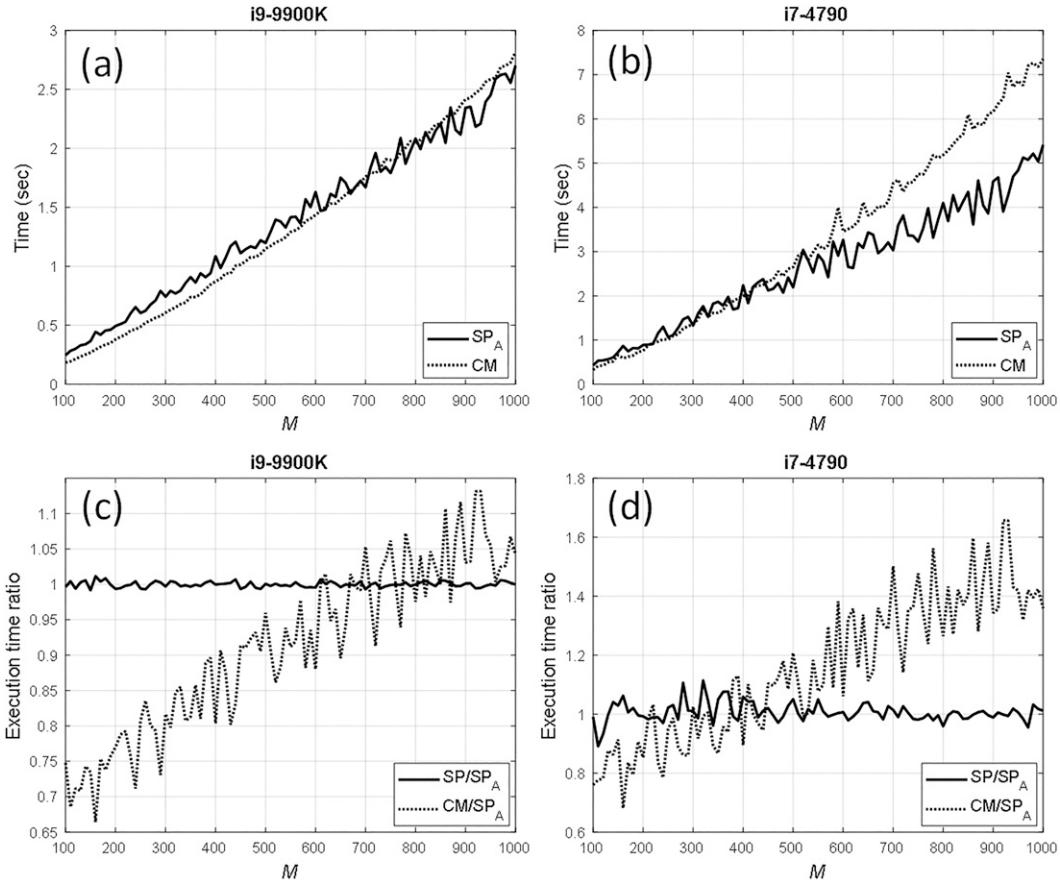

FIG. 9. The simulation times for the two systems using (a) Intel i9-9900K and (b) i7-4790 CPUs. The simulation time ratios of all three methods for the two systems using (c) Intel i9-9900K and (d) i7-4790 CPUs. The results are given for M ranging from 100 to 1000 in steps of 10, v_a of 8 m s^{-1}, and σ_v of 4 m s^{-1}.

Therein, the simulation times (Figs. 9a,b) for SP_A and CM methods, as well as their ratios (Figs. 9c,d), are presented for the two systems using Intel i9-9900K and i7-4790 CPUs. The results are given for M ranging from 100 to 1000 in steps of 10, v_a of 8 m s^{-1}, and σ_v of 4 m s^{-1}. All simulation computations were conducted using only CPUs. Apart from an expected difference in performance (i.e., i7-4790 CPU is older and less powerful than i9-9900K), an interesting observation is that the IDFT-based method (i.e., SP_A) start performing faster than the CM method for M values between 300 and 400 in the case of i7-4790 CPU, and between 700 and 800 for the i9-9900K CPU. This indicates that the IDFT methods may be more efficient for time series simulations using excessively large M values (typically used less often in practice). As demonstrated in Fig. 9, the M values at which the latter is true are system dependent.

5. Summary

In this paper, three methods for simulating realistic time series from a weather radar have been analyzed. The first two methods are based on creating the desired PSD, randomizing it, and applying the IDFT to produce time series. For accuracy, this process typically requires creating time series of a length longer than the base sample number. This increases the computational intensity and simulation duration. Therefore, the choice of simulation length (or the length of simulated time series) that achieves an optimal balance between the desired accuracy of the simulated time series and the computational intensity is of crucial importance for the first two methods. To determine the simulation length, a previously proposed approximate technique was used for the first method. For the second method, the technique proposed in this paper was used. It is based on the theoretical development presented herein and provides for the improved control of the simulation errors compared to the first method. The third method was proposed in a previous work and is based on decomposing the autocorrelation matrix and multiplying the result with a vector of Gaussian white noise samples. It retains the original simulation length and therefore circumvents the complication of determining the simulation length. A detailed analysis of the theoretical basis for all methods was presented followed by the accuracy and efficiency assessment.

The accuracy assessment was conducted by comparing the autocorrelation absolute values as well as PSD coefficients, estimated from time series, to their theoretical counterparts assuming Gaussian PSD. The results indicate that the second and the third method produce time series with sufficient accuracy in terms of autocorrelation, and PSD estimates. It is to be noted, though, that the third method produced appreciably better accuracies than the second method. Nonetheless, the errors of the second method remained within the limits below which errors were deemed insignificant. In the case of the first method, it produced accuracies equal to or better than the second method in the majority of the investigated cases. However, a case is presented where, unlike the second method, the first method produced time series with errors larger than the error significance limits. This indicates that the simulation lengths chosen by the first method may in some cases result in notably degraded accuracy compared to the second method.

To assess efficiency, all methods were executed using only the CPU but also porting part of the computations on the GPU (to improve efficiency). The results indicate that while the efficiency of the first two methods is comparable overall, the third method is significantly more economical than the other two with the difference in efficiency gradually diminishing as the number of simulated samples increases. Furthermore, it is demonstrated that the use of GPU to produce the simulated time series results in large efficiency gains.

In summary, the results of this study suggest that the autocorrelation matrix–based simulation method (i.e., the third method) produces better accuracy and efficiency compared to the other two methods for the considered cases which cover the majority of sample numbers used in practice. Nonetheless, it is shown that the application of IDFT to simulate time series is beneficial for simulation of excessively long time series where the actual lengths are system dependent (i.e., the time series lengths for which the IDFT-based methods are more efficient than the third method). When IDFT is utilized for time series simulation, the results presented herein suggest that the use of the second method may be preferable because of its improved accuracy.

The presented analysis was conducted in the case of a single-polarization time series simulation. If polarimetric time series simulation is desired, two sets of single-polarization realizations must be generated and combined based on the desired polarimetric properties. In such a case, reductions in simulation times realized by using the autocorrelation matrix–based method (vs using IDFT) would be even larger than presented here. Finally, it should be noted that the performance differences described here could have significant effects in the case of long multihour or multiday simulations.

Acknowledgments. The author would like to thank Dr. Chris Curtis and Dr. Dušan Zrnić who reviewed this manuscript and provided valuable comments that enhanced it. The statements, findings, conclusions, and recommendations are those of the author and do not necessarily reflect the views of NOAA or the U.S. Department of Commerce.

APPENDIX A

Derivation of the Simulated Time Series Autocorrelation Ensemble Average

In this appendix, a derivation to express the relation between the actual autocorrelation $[R(l)]$ of the simulated time series and the desired autocorrelation coefficient is presented. It is well known that the PSD can be found as the DTFT of the autocorrelation sequence as

$$S(f) = \sum_{n=-\infty}^{\infty} R(n) e^{-j2\pi f T_S}, \tag{A1}$$

where f is the frequency, and T_S is the sampling time. Replacing f with $k/M_S T_S$ produces PSD coefficients at discrete steps (i.e., discrete PSD coefficients created to conduct simulation) and establishes a relationship between the desired PSD and the corresponding autocorrelation of the simulated time series as

$$S(k) = \sum_{n=-\infty}^{\infty} R(n) e^{-j2\pi kn/M_S}. \tag{A2}$$

Given the condition in (2), it follows

$$\begin{aligned} S &= \frac{1}{M_S^2} \sum_{k=0}^{M_S-1} \sum_{n=-\infty}^{\infty} R(n) e^{-j2\pi kn/M_S} \\ &= \frac{1}{M_S^2} \sum_{n=-\infty}^{\infty} R(n) \sum_{k=0}^{M_S-1} e^{-j2\pi kn/M_S}. \end{aligned} \tag{A3}$$

Since

$$\sum_{k=0}^{M_S-1} e^{j2\pi kn/M_S} = \begin{cases} 0, & \text{if } n \neq pM_S \\ M_S, & \text{if } n = pM_S \end{cases}, \tag{A4}$$

where p is an integer ranging from $-\infty$ to ∞, the following ensues:

$$S = \frac{1}{M_S} \sum_{p=-\infty}^{\infty} R(pM_S). \tag{A5}$$

Because PSD coefficients are scaled so that the simulated time series have the desired power, the relation between the actual autocorrelation of the simulated time series and the desired autocorrelation coefficient is

$$R(l) = C\rho(l), \tag{A6}$$

where C is the unknown constant. It can be found using

$$S = \frac{1}{M_S} \sum_{p=-\infty}^{\infty} C\rho(pM_S) \Rightarrow C = \frac{SM_S}{\sum_{p=-\infty}^{\infty} \rho(pM_S)}. \tag{A7}$$

Hence,

$$R(l) = \frac{SM_S\rho(l)}{\sum_{p=-\infty}^{\infty} \rho(pM_S)}. \tag{A8}$$

APPENDIX B

Derivation of the PSD Coefficients Ensemble Average

In this appendix, the ensemble average of the PSD coefficients, estimated from weather time series is derived. Having weather samples

$$V(m) = V_S(m) + V_N(m), \tag{B1}$$

where $V_S(m)$ is the signal (with power S) and $V_N(m)$ is noise (with power N), each power spectrum coefficient estimate is computed as

$$\hat{S}(k) = \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} V(m)V^*(n)d(m)d(n)W^{-(m-n)k}, \tag{B2}$$

where $d(m)$ is the data window coefficient (Harris 1978), and $W = \exp(j2\pi/M)$. The ensemble average is

$$\begin{aligned}\langle\hat{S}(k)\rangle &= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\langle V(m)V^*(n)\rangle d(m)d(n)W^{-(m-n)k}\\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} R(m-n)d(m)d(n)W^{-(m-n)k}.\end{aligned} \tag{B3}$$

In the case of a weather signal, the autocorrelation is

$$R(l) = S\exp\left[-\frac{1}{2}\left(\frac{\pi\sigma_v l}{v_a}\right)^2\right]\exp\left(-j\pi l\,\frac{v}{v_a}\right) + N\delta(m), \tag{B4}$$

where $\delta(m)$ is the delta function [i.e., $\delta(m) = 1$ if $m = 0$, and zero otherwise]. Hence,

$$\begin{aligned}\langle\hat{S}(k)\rangle &= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1}[Se^{-(1/2)[\pi\sigma_v(m-n)/v_a]^2}e^{-j\pi(m-n)v/v_a} + N\delta(m-n)]\\ &\quad\times d(m)d(n)W^{-(m-n)k}\\ &= S\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}e^{-(1/2)[\pi\sigma_v(m-n)/v_a]^2}e^{-j\pi(m-n)(v/v_a+2k/M)}\\ &\quad\times d(m)d(n) + N\sum_{m=0}^{M-1}d^2(m).\end{aligned} \tag{B5}$$

By setting $m - n = l$, the following ensues:

$$\begin{aligned}\langle\hat{S}(k)\rangle &= S\sum_{l=-M+1}^{M-1} e^{-(1/2)(\pi\sigma_v l/v_a)^2}e^{-j\pi l(v/v_a+2k/M)}\\ &\quad\times\sum_{p=0}^{M-1-|l|} d(p)d(p+|l|) + N\sum_{m=0}^{M-1}d^2(m)\\ &= 2S\sum_{l=1}^{M-1}\left\{e^{-(1/2)(\pi\sigma_v l/v_a)^2}\cos\left[\pi l\left(\frac{v}{v_a}+2\,\frac{k}{M}\right)\right]\right.\\ &\quad\left.\times\sum_{p=0}^{M-1-m} d(p)d(p+l)\right\} + (S+N)\sum_{m=0}^{M-1}d^2(m).\end{aligned} \tag{B6}$$

In the case of the rectangular window

$$\begin{aligned}\langle\hat{S}(k)\rangle &= 2S\sum_{l=1}^{M-1}\left\{e^{-(1/2)(\pi\sigma_v l/v_a)^2}\cos\left[\pi l\left(\frac{v}{v_a}+2\,\frac{k}{M}\right)\right](M-l)\right\}\\ &\quad + M(S.N).\end{aligned} \tag{B7}$$

REFERENCES

Bringi, V., and V. Chandrasekar, 2001: *Polarimetric Doppler Weather Radar*. Cambridge University Press, 636 pp.

Capsoni, C., and M. D'Amico, 1998: A physically based radar simulator. *J. Atmos. Oceanic Technol.*, **15**, 593–598, https://doi.org/10.1175/1520-0426(1998)015<0593:APBRS>2.0.CO;2.

——, ——, and R. Nebuloni, 2001: A multiparameter polarimetric radar simulator. *J. Atmos. Oceanic Technol.*, **18**, 1799–1809, https://doi.org/10.1175/1520-0426(2001)018<1799:AMPRS>2.0.CO;2.

Cheong, B. L., R. D. Palmer, and M. Xue, 2008: A time series weather radar simulator based on high-resolution atmospheric models. *J. Atmos. Oceanic Technol.*, **25**, 230–243, https://doi.org/10.1175/2007JTECHA923.1.

Cho, J. Y. N., 2005: Multi-PRI signal processing for the terminal Doppler weather radar. Part II: Range–velocity ambiguity mitigation. *J. Atmos. Oceanic Technol.*, **22**, 1507–1519, https://doi.org/10.1175/JTECH1805.1.

Cooley, J. W., and J. W. Tukey, 1965: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, **19**, 297–301, https://doi.org/10.1090/S0025-5718-1965-0178586-1.

Curtis, C. D., 2018: Weather radar time series simulation: Improving accuracy and performance. *J. Atmos. Oceanic Technol.*, **35**, 2169–2187, https://doi.org/10.1175/JTECH-D-17-0215.1.

——, and S. M. Torres, 2011: Adaptive range oversampling to achieve faster scanning on the National Weather Radar Testbed phased-array radar. *J. Atmos. Oceanic Technol.*, **28**, 1581–1597, https://doi.org/10.1175/JTECH-D-10-05042.1.

Doviak, R. J., and D. S. Zrnić, 1993: *Doppler Radar and Weather Observations*. Academic Press, 562, pp.

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.

Harris, F. J., 1978: On the use of windows for harmonic analysis with the discrete Fourier transform. *Proc. IEEE*, **66**, 51–83, https://doi.org/10.1109/PROC.1978.10837.

Ivić, I. R., 2014: On the use of a radial-based noise power estimation technique to improve estimates of the correlation coefficient on dual-polarization weather radars. *J. Atmos. Oceanic Technol.*, **31**, 1867–1880, https://doi.org/10.1175/JTECH-D-14-00052.1.

——, 2017: An approach to simulate the effects of antenna patterns on polarimetric variable estimates. *J. Atmos. Oceanic Technol.*, **34**, 1907–1934, https://doi.org/10.1175/JTECH-D-17-0015.1.

——, 2019: A simple hybrid technique to reduce bias of copolar correlation coefficient estimates. *J. Atmos. Oceanic Technol.*, **36**, 1813–1833, https://doi.org/10.1175/JTECH-D-18-0226.1.

Johnson, G. E., 1994: Constructions of particular random processes. *Proc. IEEE*, **82**, 270–285, https://doi.org/10.1109/5.265353.

Lei, L., G. Zhang, R. J. Doviak, R. Palmer, B. L. Cheong, M. Xue, Q. Cao, and Y. Li, 2012: Multilag correlation estimators for polarimetric radar measurements in the presence of noise. *J. Atmos. Oceanic Technol.*, **29**, 772–795, https://doi.org/10.1175/JTECH-D-11-00010.1.

Schvartzman, D., and C. Curtis, 2019: Signal Processing and Radar Characteristics (SPARC) simulator: A flexible dual-polarization weather-radar signal simulation framework based on preexisting radar-variable data. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, **12**, 135–150, https://doi.org/10.1109/JSTARS.2018.2885614.

Sirmans, D., and B. Bumgarner, 1975: Numerical comparison of five mean frequency estimators. *J. Appl. Meteor.*, **14**, 991–1003, https://doi.org/10.1175/1520-0450(1975)014<0991:NCOFMF>2.0.CO;2.

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

Torres, S. M., and D. S. Zrnić, 2003: Whitening in range to improve weather radar spectral moment estimates. Part I: Formulation and simulation. *J. Atmos. Oceanic Technol.*, **20**, 1433–1448, https://doi.org/10.1175/1520-0426(2003)020<1433:WIRTIW>2.0.CO;2.

Yu, T. Y., G. Zhang, A. B. Chalamalasetti, R. J. Doviak, and D. Zrnić, 2006: Resolution enhancement technique using range oversampling. *J. Atmos. Oceanic Technol.*, **23**, 228–240, https://doi.org/10.1175/JTECH1841.1.

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.