INSTITUTE FOR DEFENSE ANALYSES

Implementation and Testing of Two Methods for Incorporating Concentration Fluctuations into the Hazard Prediction and Assessment Capability

Scott L. Weinrich
Stuart W. Smith
Alison E. Lawrence
Allen S. Wang

March 2019
Approved for public release;
distribution is unlimited.
Defense Threat Reduction Agency
8725 John J. Kingman Road MSC 6201
Fort Belvoir, VA 22060
IDA Paper P-10487
Log: H 19-000063

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a non-profit corporation that operates three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, project DC-6-3250, "Chemical, Biological, Radiological and Nuclear (CBRN) Analysis Support Program (ASP)," for the Joint Science and Technology Office (JSTO) of the Defense Threat Reduction Agency (DTRA). The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

For More Information:
Dr. Katherine M. Sixt, Project Leader
ksixt@ida.org, 703-575-6695
ADM John C. Harvey, Jr., USN (Ret), Director, SFRD
jharvey@ida.org, 703-575-4530

Acknowledgments
The authors wish to thank Dr. Jeffrey Urban and Dr. Nathan Platt of the Institute for Defense Analyses for their careful review of this paper.

Copyright Notice
© 2019 Institute for Defense Analyses
4850 Mark Center Drive, Alexandria, Virginia 22311-1882 • (703) 845-2000

This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013 (a)(16) [June 2013].

INSTITUTE FOR DEFENSE ANALYSES

IDA Paper P-10487

Implementation and Testing of Two Methods for Incorporating Concentration Fluctuations into the Hazard Prediction and Assessment Capability

Scott L. Weinrich
Stuart W. Smith
Alison E. Lawrence
Allen S. Wang

This page is intentionally blank.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From – To)
XX-03-2019	Final	

4. TITLE AND SUBTITLE	5a. CONTRACT NO.
Implementation and Testing of Two Methods for Incorporating Concentration Fluctuations into the Hazard Prediction and Assessment Capability	HQ0034-14-D-0001
	5b. GRANT NO.
	5c. PROGRAM ELEMENT NO(S).

6. AUTHOR(S)	5d. PROJECT NO.
Scott L. Weinrich Stuart W. Smith Alison E. Lawrence Allen S. Wang	5e. TASK NO. DC-6-3250
	5f. WORK UNIT NO.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NO.
Institute for Defense Analyses Strategy, Forces and Resources Division 4850 Mark Center Drive Alexandria, VA 22311-1882	IDA Paper P-10487 Log: H 19-000063

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR'S / MONITOR'S ACRONYM(S)
Defense Threat Reduction Agency J9CBI 8725 John J. Kingman Road MSC 6201 Ft. Belvoir, VA 22060	DTRA
	11. SPONSOR'S / MONITOR'S REPORT NO(S).

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited. Defense Threat Reduction Agency 8725 John J. Kingman Road MSC 6201 Fort Belvoir, VA 22060

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Evaluating the effectiveness of chemical and biological agent detection systems requires an accurate model of the concentration of agent in the battlespace. The time, equipment, and input data required to run computational fluid dynamics (CFD) models are often prohibitive, so analysts employ ensemble-average models, such as the Defense Threat Reduction Agency's Hazard Prediction and Assessment Capability (HPAC). Ensemble-average concentrations do not include the concentration fluctuations experienced in the field that would have an impact on the performance of detectors. In this paper, we implement and test two methods intended to leverage the concentration variance, which is also provided by HPAC, to add concentration fluctuations to the ensemble-average plume. The first-order Markov process uses a given probability density function to generate pointwise concentration time-series, and the multi-fractal method generates spatially correlated concentration fluctuations. We test these methods using data from the Fusing Sensor Information from Observing Networks (FUSION) Field Trial 2007 (FFT 07). Preliminary comparison of concentration intermittency, upcrossing frequency, upcrossing duration, and notional chemical and biological agent detection across the two methods, the FFT07 data, and HPAC and CFD model replications of an FFT07 trial give mixed results regarding which method provides the best estimates of concentration fluctuations.

15. SUBJECT TERMS

concentration variance; concentration fluctuations; first-order Markov process; fractal; multi-fractal; Gaussian puff models; Hazard Prediction and Assessment Capability; HPAC; biological warfare; chemical warfare; modeling and simulation; transport and dispersion modeling; chemical detection; biological detection; ensemble-average transport and dispersion models; FUSION Field Trial 2007; FFT07

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NO. OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Richard Fry
U	U	U	UU	106	19b. TELEPHONE NUMBER (Include Area Code) 571-616-6029

This page is intentionally blank.

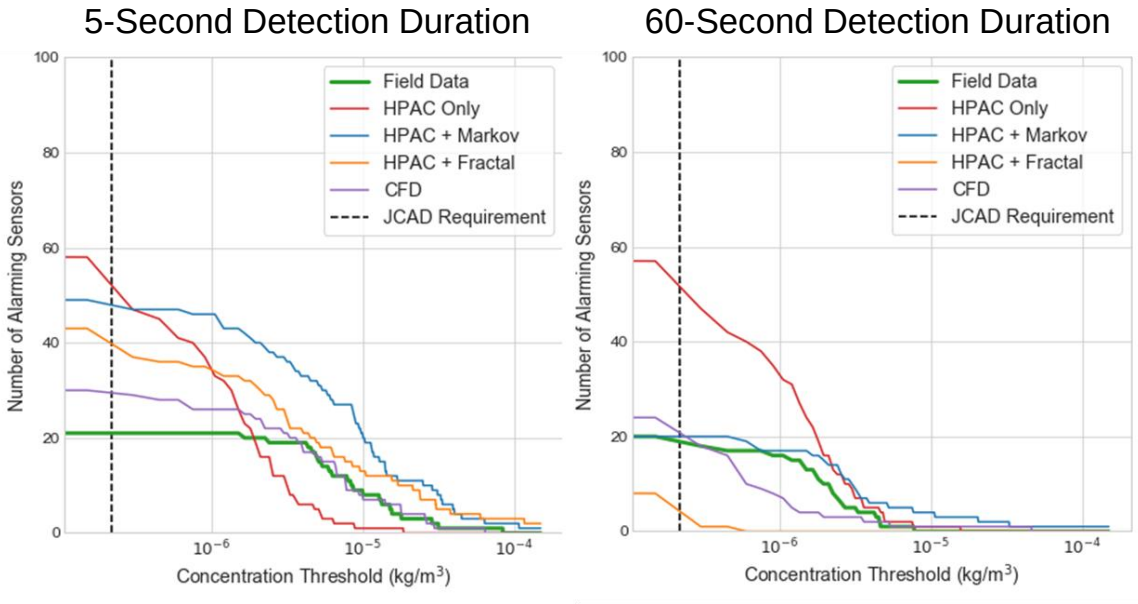
Executive Summary

Turbulent diffusion of disseminated chemical or biological material results in fluctuating concentration fields due to chaotic mixing of the air. Direct simulation of turbulent diffusion using computational fluid dynamics would produce fluctuating concentration fields, but may not be practical for many types of analyses, due to its computational intensity and the complexity of integrating meteorological observations. On the other hand, many computationally efficient Gaussian puff transport and dispersion models provide estimates of the mean concentration field with a basic characterization of the release and meteorology, but are unable to replicate concentration fluctuations. While the mean concentration field may be sufficient for modeling time-integrated concentrations, such as for modeling environmental collectors, it is a poor proxy for a fluctuating concentration field when modeling response to instantaneous concentrations, which is often needed for modeling chemical and biological detection systems.

Two methods of simulating concentration fluctuations, termed the "Markov method" and "fractal method", were investigated in an effort to improve the accuracy with which detector response can be modeled in conjunction with Gaussian puff transport and dispersion models. The Markov method employs a first-order Markov chain process to simulate fluctuating concentration time-series at discrete locations. The fractal method is founded upon fractal geometry techniques and produces a spatially correlated fluctuating concentration field. Both of these methods require three inputs: concentration mean, concentration variance, and temporal correlation; the fractal method also requires a spatial correlation factor. These two methods were applied to concentration ensembles generated from a subset of the field trials conducted during Fusion Field Trial 2007 (FFT 07), limited to continuous releases that met certain meteorological criteria, as well as Gaussian puff transport and dispersion simulation output produced by the Hazard Prediction and Assessment Capability (HPAC). Evaluation metrics were used to compare the simulated fluctuating concentrations with the actual fluctuating concentrations from individual field trials. For application to the FFT 07 ensembles, the evaluation metrics characterized the frequency and duration of upcrossing events, i.e., occurrences in which the concentration exceeds a specified threshold. These metrics were evaluated across a range of concentration thresholds. For application to HPAC outputs, a simple model of detector operation was used to quantify the total number of times a detector would alarm and the number of alarming detectors as evaluation metrics. This model of detector operation requires the concentration to exceed a given threshold for a specified duration to trigger a detector alarm. These evaluation metrics are therefore functions of both the detection threshold and detection duration used to parameterize the detector model. To provide additional context for evaluating the Markov and fractal methods, the detector model was also applied to a concentration field obtained from a large eddy computational fluid dynamics (CFD)

simulation. The CFD simulation was carefully created to replicate a specific field trial and demonstrates the fidelity of modeled results that may be obtained through considerable analytical and computational effort.

The fractal method performed marginally better than the Markov method when applied to the field trial ensembles, but this assessment is highly dependent on the particular concentration threshold of interest, and neither method replicated the metrics obtained from the actual field trials with exceptional fidelity. When coupled with HPAC, both methods yielded metrics that were generally more faithful to the field trial data than the HPAC simulation output alone, yet the methods still deviated substantially from the field trials in certain concentration threshold regimes, as observed in the following figure. The fractal method performed somewhat better than the Markov method for a detection duration of 5 seconds (typical for chemical agent detectors), but the Markov method performed remarkably well for a 60-second detection duration (typical for biological agent detectors) for the release modeled. Note that these are not the only operationally relevant detection durations, as many different detector algorithms exist. Also note that the rank ordering of methods changes with both the detection duration and the concentration threshold, which is a concern for overall model performance. Further, the FFT 07 grid is significantly denser than a typical biological or chemical sensor network, and thus the number of alarming detectors shown in the figure may not be operationally relevant in many cases.



Number of Alarming Detectors in the FFT 07 Grid for 5-Second and 60-Second Detection Durations

Contents

This page is intentionally blank.

1. Introduction

A. Background and Motivation

The Hazard Prediction and Assessment Capability's (HPAC's) primary transport and dispersion model, the Second-order Closure Integrated Puff (SCIPUFF), represents the concentration field as the superposition of collections of three-dimensional Gaussian "puffs." These Gaussian puffs arise as a solution to the Reynolds-averaged advection-diffusion equation for instantaneous releases.[1] Although Reynolds averaging allows the advection-diffusion equation to be solved analytically, information regarding fluctuations in the concentration is necessarily lost. As such, this "Gaussian puff" model provides estimates of the mean concentration field, but does not attempt to simulate the fluctuations in concentration resulting from turbulent diffusive processes. This mean concentration field mimics an ensemble mean concentration, i.e., an average over the collection of all possible turbulent realizations of the plume.

The mean concentration field alone may be reasonably adequate for estimating the amount of agent collected by equipment, such as dry filter collectors, if the duration of exposure is reasonably long, because concentration fluctuations on relatively short time scales would be expected to converge to the mean over the entire collection time. In contrast, the performance of detectors that respond to nearly instantaneous aerosol concentrations may be dramatically affected by concentration fluctuations. Using the ensemble mean concentration to model detector operation may either underestimate or overestimate the detector performance, depending on the particular scenario. If the detection threshold is above the peak ensemble mean concentration at a given detector location, then the detector performance might be underestimated because concentration fluctuations could increase the peak concentrations above the detections threshold, as depicted in Figure 1. Alternatively, the detector performance may be overestimated if the detector needs to be exposed to concentrations greater than the detection threshold for some minimum contiguous interval of time, termed the "detection duration." In such a case, rapid concentration fluctuations may cause the concentration to exceed the detection limit for intervals too short to permit detection, as depicted in Figure 2. To more accurately model the operation of such detectors, the atmospheric transport and dispersion model needs to incorporate methods to simulate realistic concentration fluctuations.

[1] R. Ian Sykes, Stephen F. Parker, Douglas S. Henn, Biswanath Chowdhury, "SCIPUFF Version 2.7 Technical Documentation," Sage Management, December 2011.

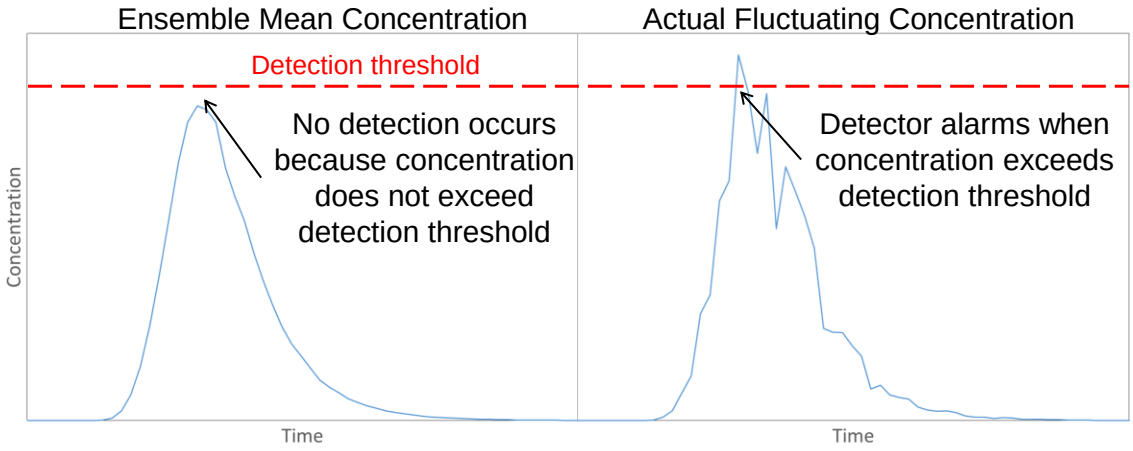


Figure 1. Notional Depiction of a Missed Detector Alarm Resulting from the Use of the Ensemble Mean Concentration

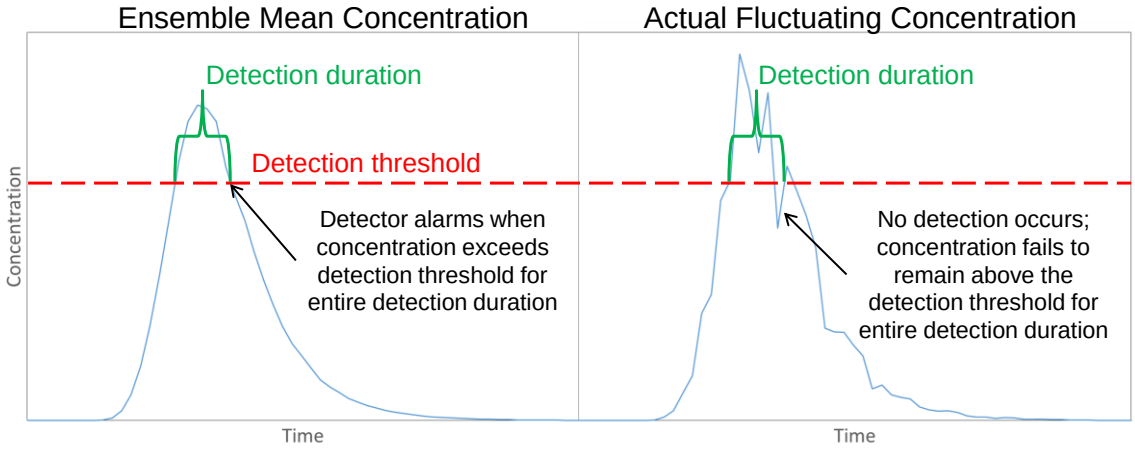


Figure 2. Notional Depiction of an Incorrect Detector Alarm Resulting from the Use of the Ensemble Mean Concentration

The Joint Effects Model (JEM) is the atmospheric transport and dispersion (AT&D) model currently accredited by the Department of Defense (DOD) for use in Chemical, Biological, Radiological, and Nuclear (CBRN) analyses.[2] JEM leverages capabilities from other models and is mostly based on HPAC.[3] HPAC itself has been accredited as a

---

2 Stephen V. Reeves, "Class Accreditation for the Joint Effects Model Increment 1," (memorandum, Washington, DC: Joint Project Manager for Information Systems, August 15, 2007).

3 Naval Surface Warfare Center, *Independent Verification and Validation of Joint Effects Model, Final Report*, (Dahlgren, VA: Naval Surface Warfare Center, March 5, 2007), 56.

Science and Technology (S&T) prototype for use by those with subject matter expertise.[4] In addition to the mean concentration field, HPAC, through SCIPUFF, provides an estimate of the concentration variance. HPAC obtains the variance estimates by relating fluctuations in the scalar concentration field to fluctuations in the wind velocity derived from turbulence theory. Together, the concentration mean and variance can be used to estimate fluctuations in a concentration field.

B. Overview of Analysis

This paper investigates two methods of simulating fluctuating concentration fields using only mean concentration and variance fields, and compares the results of those methods with actual concentrations from outdoor releases. One method employs a first-order Markov chain process to independently simulate fluctuating time series at discrete locations, as described in Chapter 2; the other method leverages fractal geometry to simulate a fluctuating concentration field, as described in Chapter 3. Both methods were applied to ensemble mean and variance statistics generated from actual outdoor field trials to investigate their ability to produce realistic concentration fluctuations. The generation of field trial ensembles is described in Chapter 4, and consists primarily of selecting continuous releases with similar meteorology, and translating and rotating them to a common coordinate system. In practical use, these methods would be applied to HPAC outputs, rather than experimental field trial ensembles. As such, a single release from Fusion Field Trial 2007 (FFT 07) was simulated in HPAC, as described in Chapter 5. The application of the Markov and fractal methods to the field trial data is described in Chapter 6. The simulated fluctuating concentrations generated by each method were then characterized by a set of metrics, described in Chapter 7 and compared to the actual fluctuating concentrations recorded during the field trials to assess the fidelity with which these methods are able to simulate concentration fluctuations. Both methods were applied to the HPAC outputs to assess how these methods perform when coupled with HPAC. The results of these comparisons are discussed in Chapter 8.

[4] Kenneth A. Myers, "Defense Threat Reduction Agency Accreditation of the Joint Effects Model, Science and Technology Prototype/Hazard Prediction and Assessment Capability 5.1," (memorandum, Fort Belvoir, VA: DTRA, September 18, 2012).

This page is intentionally blank.

2. Markov Method for Simulating Concentration Fluctuations

Markov chain models simulate stochastic (random) processes in which the probability of a given outcome at a particular time depends only on the previous state of a system.[5] This type of model lends itself intuitively to the generation of concentration fluctuations, which are stochastic by nature. This ability of a first-order, or "memoryless," Markov chain method to work with only a single state allows the reduction of a complex natural process, which can otherwise be simulated only with exacting parameterization and extensive computation, to tenable pieces. The only necessary parameter to enable the creation of Markov chains for concentration fluctuation is the choice of the probability density function (PDF) (including its functional form and parameters) that will govern the amplitude and frequency of the fluctuations. The choice of PDF and the theoretical basis of the model presented herein relies heavily on previous theoretical work[6,7,8] and lessons learned from previous implementations.[9]

A. Concentration Probability Density Functions

Multiple competing probability density functions for simulating concentration fluctuations exist in literature, though they all assume a stationary source and constant meteorology. The primary two—a shifted, clipped gamma distribution and a clipped lognormal distribution—have been shown to fit some datasets well, and each has been

5 Richard Serfozo, *Basics of Applied Stochastic Processes*, (Berlin, Germany: Springer Science & Business Media, 2009), 2.

6 Eugene Yee and R. Chan, "A Simple Model for the Probability Density Function of Concentration Fluctuations in Atmospheric Plumes," *Atmospheric Environment* 31, no. 7 (1997), 991–1002.

7 Shuming Du, David J. Wilson, and Eugene Yee, "A Stochastic Time Series Model for Threshold Crossing Statistics of Concentration Fluctuations in Non-Intermittent Plumes," *Boundary-Layer Meteorology* 92, no. 2 (1999), 229–241.

8 T.L. Hilderman and D.J. Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," *Boundary-Layer Meteorology* 91, no. 3 (1999), 451-482.

9 Ajith Gunatilaka, Alex Skvortsov, and Ralph Gailis, "High Fidelity Simulation of Hazardous Plume Concentration Time Series Based on Models of Turbulent Dispersion," (paper presented at the 15th International Conference on Information Fusion (FUSION), Singapore, July 9-12, 2012), 1838-1845.

touted as better than its competitor for a given dataset.[10,11] Many implementations of the Markov method for simulating concentration fluctuation use the shifted, clipped gamma distribution proposed by Yee and Chan as the PDF for randomly fluctuating concentrations at a fixed point from a stationary source.[12] Initially, this investigation incorporated such a PDF, but it became increasingly ad hoc as numerous issues arose and workarounds were implemented. Issues included numerical problems with the transcendental equation solver and the necessity of additional mathematical constraints and approximations to prevent the Markov chain from diverging when initialized with very large ensemble variances. Further, the nature of the shifted gamma distribution required the renormalization of the final concentration time series to the known concentration mean, which, especially in the cases of high intermittency, artificially dilated the final fluctuations. After some experimentation, the clipped lognormal PDF was used instead, as presented by Hilderman and Wilson.[13] The remainder of this section focuses on the use of the clipped lognormal PDF.

Hilderman and Wilson (1999) suggest the use of a clipped lognormal distribution, in which the formulation of the shifted concentration, $\tilde{c} = c_+ - c_{base}$, is implemented prior to the calculation of the PDF.[14] As such, the lognormal PDF and Markov process are calculated from the artificial concentration, c_+, and then shifted by c_{base} to obtain the concentration time series, $\tilde{c}$. This enables the direct calculation of the final concentration time series without the need to rescale to the original mean.

The full derivation of the lognormal PDF and associated variables for the upshifted, pre-clipped concentration time series are provided in Hilderman and Wilson (1999).[15] The pertinent distribution for the implementation used here is

$$p(c_+) = \frac{1}{\sqrt{2\pi}\sigma_{l+}c_+} \exp\left(-\frac{\left(\ln\left(\frac{c_+}{c_{50+}}\right)\right)^2}{2\sigma_{l+}^2} \right) \quad (1)$$

where σ_{l+}^2 and c_{50+} are the lognormal variance and median of the distribution, respectively. These are calculated by first calculating the fluctuation intensity, i,

---

10 Du, Wilson, and Yee, "A Stochastic Time Series Model for Threshold Crossing Statistics of Concentration Fluctuations in Non-Intermittent Plumes," 229–241.

11 Hilderman and Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," 451-482.

12 Yee and Chan, "A Simple Model for the Probability Density Function of Concentration Fluctuations in Atmospheric Plumes," 991–1002.

13 Hilderman and Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," 451-482.

14 Ibid., 451-482.

15 Ibid., 458-460, 465.

$$i^2 = \sigma_c^2 / \bar{c}^2 \tag{2}$$

which is dependent on the mean, $\bar{c}$, and variance, σ_c^2, of the stationary concentration time series. The fluctuation intensity can then be applied to the equations given in Wilson (1995)[16] which describe the empirical relationship between fluctuation intensity; conditional fluctuation intensity (fluctuation intensity for only nonzero concentrations), i_p; and intermittency, γ:

$$i_p^2 = \frac{2i^2}{2+i^2} \tag{3}$$

$$\gamma = \frac{1+i_p^2}{1+i^2} \tag{4}$$

c_{50+} can then be calculated directly as

$$c_{50+} = \frac{\bar{c}}{\gamma\sqrt{1+i^2}} \tag{5}$$

The final two quantities necessary to implement the clipped lognormal concentration fluctuation PDF, σ_{l+}^2 and c_{base}, are calculable from a system of equations derived by Hilderman and Wilson: [17]

$$\gamma = 0.5 * \left(1 - \text{erf}\left(\frac{\ln(\varphi_{base})}{\sqrt{2}\sigma_{l+}}\right)\right) \tag{6}$$

$$i_p^2 = \frac{\gamma\overline{\varphi^2}}{\Phi^2} - 1 \tag{7}$$

$$\Phi = \frac{\exp\left(\frac{\sigma_{l+}^2}{2}\right)}{2}\left(1 - \text{erf}\left(\frac{\ln(\varphi_{base}) - \sigma_{l+}^2}{\sqrt{2}\sigma_{l+}}\right)\right) - \frac{\varphi_{base}}{2}\left(1 - \text{erf}\left(\frac{\ln(\varphi_{base})}{\sqrt{2}\sigma_{l+}}\right)\right) \tag{8}$$

$$\overline{\varphi^2} = \frac{\exp(2\sigma_{l+}^2)}{2}\left(1 - \text{erf}\left(\frac{\ln(\varphi_{base}) - 2\sigma_{l+}^2}{\sqrt{2}\sigma_{l+}}\right)\right) - \varphi_{base}\exp\left(\frac{\sigma_{l+}^2}{2}\right)\left(1 - \text{erf}\left(\frac{\ln(\varphi_{base}) - \sigma_{l+}^2}{\sqrt{2}\sigma_{l+}}\right)\right) + \frac{\varphi_{base}^2}{2}\left(1 - \text{erf}\left(\frac{\ln(\varphi_{base})}{\sqrt{2}\sigma_{l+}}\right)\right) \tag{9}$$

where φ is the pseudo-concentration formed by scaling the concentration time series by its stationary mean, with Φ indicating the mean and $\overline{\varphi^2}$ indicating the total second moment. This allows for c_{base} to then be calculated by simply reversing this scaling for the conditional concentrations:

$$c_{base} = \varphi_{base} * c_{50+} \tag{10}$$

---

[16] David J. Wilson, *Concentration Fluctuations and Averaging Time in Vapor Clouds* (New York, NY: Center for Chemical Process Safety of the American Institute of Chemical Engineers, 1995).

[17] Hilderman and Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," 458-460.

B. Stochastic Time Series

A concentration time series is generated via the stochastic differential equation:

$$\mathrm{d}c = a(c,t)\mathrm{d}t + b(c,t)\mathrm{d}\xi \tag{11}$$

where the first term is the correlated portion of the differential concentration increment, and the second term is a random forcing in which $\mathrm{d}\xi$ is a Wiener process with variance dt and zero mean.[18] Physically, $a(c,t)$ is the drift coefficient, and $b(c,t)$ is the diffusion coefficient. To use this first-order Markov process, the concentration time series at a given spatial point must be continuous, with no intermittent zero concentrations (as the calculation of $b(c,t)$ requires division by the instantaneous ensemble mean concentration as seen in Equation 14), which is assumed here. As described in section 2.A., this continuous time series is later shifted and clipped as a whole to create intermittency.

The drift and diffusion coefficients are related via the Fokker-Planck equation, but the choice of either is arbitrary within certain limitations.[19] The choice of a mean-centric drift coefficient,[20]

$$a = -\frac{c-\bar{c}}{T_c} \tag{12}$$

where T_c is an appropriate time scale for concentration fluctuations, gives a calculable diffusion coefficient,[21]

$$b^2 = \frac{2}{T_c p(c)} \int_c^\infty (c-\bar{c}) p(c)\, dc \tag{13}$$

where $p(c)$ is the concentration PDF. Equation 1 can then be substituted into Equation 13 and the integral solved to obtain a formula for the diffusion coefficient:

$$b^2 = \frac{c_+}{T_{c+} p(c_+)} \left(\mathrm{erf}\left(\frac{\ln\left(\frac{c_+}{c_{50+}}\right)}{\sqrt{2}\sigma_{l+}} \right) - \mathrm{erf}\left(\frac{\ln\left(\frac{c_+}{c_{50+}}\right) - \sigma_{l+}^2}{\sqrt{2}\sigma_{l+}} \right) \right) \tag{14}$$

These equations for the drift and diffusion coefficients allow discretization of the stochastic differential equation into n time steps ($t_n = n\Delta t$) as,[22]

$$c_{n+1} = c_n + a(c_n)\Delta t + b(c_n)\sqrt{\Delta t} m_n \tag{15}$$

where m_n is a random Gaussian variate with zero mean and unit variance.

---

[18] Du, Wilson, and Yee, "A Stochastic Time Series Model for Threshold Crossing Statistics of Concentration Fluctuations in Non-Intermittent Plumes," 229–241.

[19] Ibid., 231-232.

[20] Ibid., 232.

[21] Ibid.

[22] Ibid.

To account for intermittency appropriately, the calculated concentration time series is shifted down by c_{base} and clipped below 0.

A single input is necessary to determine the parameters from Equations 3 through 10: the fluctuation intensity, which is calculated for each sensor using Equation 2.

To calculate the correlation time scale, T_c, the autocorrelation coefficient of the concentration fluctuation from the mean is calculated at each sensor in the ensemble, and then averaged across trials in the ensemble. This leaves $R(\tau)$, the average autocorrelation as a function of lag period. The time correlation scale used is then the *e*-folding time, which is defined as the time at which the autocorrelation function first reaches $1/e$, as described in Appendix B.

C. Extending to the Transient Case

The theoretical basis for the Markov method assumes stationarity: it is only empirically valid for a steadily emitting, non-moving source in an invariant wind field. Though most of the trial ensembles have a stationary source, the variable meteorology violates the condition of stationarity. The Markov method may be extended to transient cases by assuming that the condition of stationarity is satisfied across sufficiently short time intervals, as described by Gunatilaka et al.[23] In this approach, each time step in the data is considered a stationary state and is subdivided into shorter intervals (termed "transient time steps") on which the Markov method is applied. This process stabilizes the discontinuous steps in the mean concentration across the original time steps, so that the mean-centric parameter (Equation 12) remains dominant over the fluctuation parameter (Equation 14), as the mean-centric parameter affects the first-order Markov process on the order of $\Delta t/T$ and the fluctuation parameter only affects it as the square root of $\Delta t/T$ (Equation 15). Furthermore, the method from Gunatilaka et al. for creating multiple realizations concurrently[24] helps to achieve a good spread of variates from the PDF by allowing for an increased number of draws within each stationary state, which are then shifted and clipped together.

To maintain appropriate temporal correlation when the transient time steps are smaller than the temporal correlation scale, the last value of each realization is extracted, prior to shifting and clipping, to use as a seed value for the next time step. After the seed values are recorded, each set of subdivided time steps is averaged as necessary to resample the time series into the original time steps, which maintains the same temporal resolution for comparison to the field trials.

---

[23] Gunatilaka, Skvortsov, and Gailis, "High Fidelity Simulation of Hazardous Plume Concentration Time Series Based on Models of Turbulent Dispersion," 1838-1845.

[24] Ibid.

Because the fluctuation intensity is not calculable when the mean concentration is zero, the mean concentration time series to which the method is applied must be altered for use with the Markov method. The mean concentration time series are truncated to include only nonzero mean concentrations. Each value for mean concentration and variance immediately prior to the zero value is extended across the excised time period to create a non-intermittent time series over which the first-order Markov process can continuously evolve.

D. Limitations

The Markov method separates each geospatial location in the concentration ensemble into its own independent entity. Thus, spatial correlation between locations, no matter how close together, are not maintained. The lack of spatial correlation prevents a coherent plume from being reconstructed from a dense array of individual fluctuating concentration time series. If the distance between locations is larger than the spatial correlation scale, then this limitation only minimally affects the simulated concentration time series. Due to this limitation, the Markov method might be useful for modeling and simulation involving independent geospatial points, such as an investigation into optimal detector arrays, but not for modeling that requires short-range coherence or plume reconstruction, such as development of algorithms for responding to two detector alarms dependent on their relative locations.

The implementation of the Markov method detailed herein extrapolates beyond the assumption used to develop it. Approximations involved in the formulation of the equations that govern the drift and diffusion coefficients rely on the assumption of stationarity. Due to non-stationarity in the present application, the disjointed fusing together of each transient time step is non-physical and can produce large fluctuations that defy the PDF, despite efforts to smooth these interfaces.

3. Fractal Method for Simulating Concentration Fluctuations

Fractal geometry provides an apt framework for describing turbulent phenomena.[25] Fractals are constructs that exhibit some degree of geometric self-similarity across a range of scales. Turbulence involves the dissipation of energy in a cascade of eddies at decreasing scales terminating at the Kolmogorov length.[26] The cascade propagates as instabilities in eddies at a particular scale beget eddies at smaller scales. At the Kolmogorov length scale, inertial forces are comparable to viscous forces (i.e., the Reynolds number approaches unity) and the energy is dissipated as heat, rather than cascading further to smaller eddies. While the wide range of relevant spatiotemporal scales makes the direct application of the Navier-Stokes equation to solve the fluid dynamics problem computationally infeasible, the equations governing the behavior of fractals across spatial scales are comparatively simple to apply. Historically, this conception of turbulence as hierarchies of self-similar eddy structures has led to the application of fractal theory to characterize various atmospheric phenomena.[27,28,29]

A. Fractal Sum of Pulses (FSP) Approach

A method for simulating concentration fluctuations in dispersing plumes based on fractal geometry was implemented as described by Sykes et al.[30] The concentration fluctuations are modeled as the summation of randomly generated pulses, i.e., a fractal sum of pulses (FSP) approach. The pulses are located on a series of spatiotemporal grids obtained through iterative refinement to yield overlapping pulses with a range of spatial

---

25 K.R. Sreenivason and C. Meneveau, "The Fractal Facets of Turbulence," *Journal of Fluid Mechanics* 173 (1986), 382.

26 Ibid.

27 W.S. Lewellen and R.I. Sykes, "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes," *Journal of Climate and Applied Meteorology* 25, no. 8 (1986).

28 S. Lovejoy and B.B. Mandelbrot, "Fractal Properties of Rain, and a Fractal Model," *Tellus* 37A, no. 3 (1985).

29 R.I. Sykes and R.S. Gabruk, "Fractal Representation of Turbulent Dispersing Plumes," *Journal of Applied Meteorology* 33, no. 6 (1994).

30 R.I. Sykes, R.S. Gabruk, D.S. Henn, "The Small-scale Structure of Dispersing Clouds in the Atmosphere," A.R.A.P. Report No. 710 (Princeton, NJ: Titan Corporation, July 1994).

scales. These fluctuations are superimposed on the mean concentration field to yield a fluctuating concentration field.

To begin, this fractal method requires the specification of the mean concentration and variance on an initial coarse grid of locations at regular time intervals. This grid is then iteratively refined, dividing the grid spacing in half until the maximum number of refinements, N, has been evaluated. Thus, the grid spacing for each dimension at the n^{th} refinement level is given in Equations 16, 17, and 18.

$$\Delta x_n = \Delta x_0 2^{-n} \tag{16}$$

$$\Delta y_n = \Delta y_0 2^{-n} \tag{17}$$

$$\Delta t_n = \Delta t_0 2^{-n} \tag{18}$$

In this paper, implementation of the FSP model characterized the one-point probability distribution at each grid location using a lognormal distribution with mean, $\mu_{L;i,j,k}$, and variance, $\sigma_{L;i,j,k}$ where the indices $i, j,$ and k denote the grid coordinate in each dimension. LIDAR observations of dispersing plumes from industrial exhaust stacks were better characterized by a clipped-normal distribution, but large eddy simulations (LES) yielded concentration fields that demonstrate a probability distribution closer to a lognormal distribution than a clipped-normal distribution.[31,32] The clipped-normal distribution was not selected because its implementation was complicated by the lack of an analytical solution for the underlying Gaussian parameters. In attempts to implement a clipped-normal distribution, the numerical solvers used to obtain the underlying Gaussian parameters were slow and prone to failure to converge if not very precisely initialized. For the lognormal distribution, the parameters of the underlying Gaussian distribution, $\mu_{G;i,j,k}$ and $\sigma_{G;i,j,k}$, at each grid location were calculated by Equations 19 and 20.[33]

$$\mu_{G;i,j,k} = \ln\left(\frac{\mu_{L;i.j.k}^2}{\sqrt{\mu_{L;i,j,k}^2 + \sigma_{L;i,j,k}^2}}\right) \tag{19}$$

$$\sigma_{G;i,j,k} = \sqrt{\ln\left(\left(\frac{\sigma_{L;i,j,k}^2}{\mu_{L;i.j.k}^2}\right) + 1\right)} \tag{20}$$

---

31 Lewellen and Sykes, "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes."

32 Sykes, Gabruk, and Henn, "The Small-scale Structure of Dispersing Clouds in the Atmosphere," A.R.A.P. Report No. 710.

33 Lewellen and Sykes, "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes."

Triangular pulses are then generated for each grid location at each refinement of the grid. The triangular pulse functions in each direction are given by Equations 21, 22, and 23.

$$P_x\left(x, X_{n,i}, \Delta x_n\right) = \begin{cases} 1 - \frac{|x - X_{n,i}|}{\Delta x_n}, |x - X_{n,i}| < \Delta x_n \\ 0 \qquad\qquad , |x - X_{n,i}| \geq \Delta x_n \end{cases} \tag{21}$$

$$P_y\left(y, Y_{n,j}, \Delta y_n\right) = \begin{cases} 1 - \frac{|y - Y_{n,j}|}{\Delta y_n}, |y - Y_{n,j}| < \Delta y_n \\ 0 \qquad\qquad , |y - Y_{n,j}| \geq \Delta y_n \end{cases} \tag{22}$$

$$P_t\left(t, T_{n,k}, \Delta t_n\right) = \begin{cases} 1 - \frac{|t - T_{n,k}|}{\Delta t_n}, |t - T_{n,k}| < \Delta t_n \\ 0 \qquad\qquad , |t - T_{n,k}| \geq \Delta t_n \end{cases} \tag{23}$$

The centroid of each pulse, $(X_{i,n}, Y_{j,n}, T_{k,n})$, is drawn from a uniform distribution around the associated grid point, as shown in Equations 24, 25, and 26.

$$X_{n,i} \in U\left(x_{n,i} - \frac{\Delta x_n}{2}, x_{n,i} + \frac{\Delta x_n}{2}\right) \tag{24}$$

$$Y_{n,j} \in U\left(y_{n,j} - \frac{\Delta y_n}{2}, y_{n,j} + \frac{\Delta y_n}{2}\right) \tag{25}$$

$$T_{n,k} \in U\left(t_{n,k} - \frac{\Delta t_n}{2}, x_{n,k} + \frac{\Delta t_n}{2}\right) \tag{26}$$

The amplitude of each pulse, $A_{n,i,j,k}$, is drawn from a standard normal distribution, as shown in Equation 27.

$$A_{n,i,j,k} \in N(0,1) \tag{27}$$

To replicate the ensemble statistics, the amplitude of the pulses must scale with standard deviation at each grid location. Because overlapping pulses from each refinement level are summed, the standard deviation at each refinement level must sum to yield the overall standard deviation at each grid location, $\sigma_{G;i,j,k}$, i.e., the variance at each location can be expressed as a series expansion across the range of spatiotemporal scales. The standard deviation of the initial grid, $\sigma_{G;n=0,i,j,k}$, is related to the ensemble standard deviation, $\sigma_{G;i,j,k}$, in Equation 28.

$$\sigma_{G;n=0,i,j,k} = \sqrt{\frac{3}{2}\sigma_{G;i,j,k}^2 \frac{1 - 2^{-2\alpha}}{1 - 2^{-2(N+1)\alpha}}} \tag{28}$$

The α parameter defines how the amplitude of fluctuations decreases with increasing grid resolution and is calculated from the fractal dimension, D, as shown in Equation 29 for a three-dimensional grid. Studies of various atmospheric phenomena have suggested

that a fractal dimension of 1.30-1.40 is appropriate for describing cloud structures, rainfall patterns, and large-eddy simulations of dispersing plumes.[34,35,36,37,38]

$$\alpha = 3 - D \tag{29}$$

The standard deviation at the n^{th} refinement level, $\sigma_{G;n,i,j,k}$, can be calculated from the standard deviation of the initial grid, $\sigma_{G;n-1,i,j,k}$, as shown in Equation 30. With each refinement of the grid, each parent cell on the grid is divided into eight daughter cells, half of which are randomly selected to have a factor p and the other half of which have a factor $1 - p$. The variable p' in Equation 30 represents the factor that is randomly selected from the set $\{p, 1 - p\}$ for each grid point.

$$\sigma_{G;n,i,j,k} = \sigma_{G;n-1,i,j,k} 2^{\frac{1-2n\alpha}{2}} p'^{\frac{1}{2}} \tag{30}$$

The concentration fluctuations are evaluated by summing the contribution of each pulse with the amplitude of each pulse scaled by the local standard deviation, $\sigma_{G;n,i,j,k}$, at each grid location. The fluctuating concentration field, $C(x, y, t)$, is then obtained by exponentiating the sum of the fluctuations with the Gaussian mean field, $\mu_{G;i,j,k}$, as shown in Equation 31.

$$C(x, y, t) = e^{\mu_{G;i,j,k} + \sum_n \sum_i \sum_j \sum_k A_{n,i,j,k} \sigma_{G;n,i,j,k} P_x(x, x_{i,n}, \Delta x_n) P_y(y, y_{j,n}, \Delta y_n) P_t(t, t_{k,n}, \Delta t_n)} \tag{31}$$

B. Bulk Translation of Dispersing Plume

By itself, the fractal method does not replicate the bulk translation of coherent structures in a dispersing plume. The plume structures directly emerging from the fluctuating concentration field generated by the fractal method are stationary in nature. To mitigate this limitation, the initial concentration and variance fields may be converted to a Lagrangian coordinate system prior to application of the fractal method. This is achieved by shifting the plume, such that the centroid of the mean concentration field at each time step is fixed at the center of the spatiotemporal grid. The plume is then effectively stationary when the fractal method is applied. To revert to an Eulerian coordinate system, the shift performed at each time step is then inverted on the fluctuating concentration field generated by the fractal method.

---

[34] Sykes, Gabruk, and Henn, "The Small-scale Structure of Dispersing Clouds in the Atmosphere," A.R.A.P. Report No. 710.

[35] Sreenivason and Meneveau, "The Fractal Facets of Turbulence," 382.

[36] Lewellen and Sykes, "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes."

[37] Lovejoy and Mandelbrot, "Fractal Properties of Rain, and a Fractal Model."

[38] Sykes and Gabruk, "Fractal Representation of Turbulent Dispersing Plumes."

C. Limitations

As implemented, the temporal and spatial correlation scales used to define the initial grid resolution of the fractal method are fixed across the entire domain of individual simulations. This may be inconsistent with the dynamic nature of an expanding plume, and the temporal correlation scales reported by HPAC vary across time and location. Soon after release or very near the release location, it is anticipated that the spatiotemporal correlation scales of concentration fluctuations would be smaller, as the physical dimensions of the plume are relatively small. Long after the release has ended or far downwind, the spatiotemporal correlation scales may be much larger because the plume has expanded and may have developed relatively large coherent structures. A fine spatiotemporal scale may be needed to accurately characterize the fluctuations near the release location, but could present a significant computational burden if the scales are fixed across the entire domain of the simulation. Conversely, a coarse spatiotemporal scale may be sufficient farther from the release and be computationally tractable, but it may lead to inaccurately large concentration fluctuation correlation distances very near the release. The fractal method could be improved by adapting it for use with variable time scales. Presumably, the mean field could be preserved with variable time scales if the pulses are truncated at adjacent grid points to prevent a pulse with a long time scale from overlapping multiple adjacent temporal grid points; further research must be conducted to verify this presumption.

The fractal method does not enforce mass conservation. As such, it is possible for the concentration across the majority of the spatial domain to simultaneously fluctuate in the same direction which would be non-physical, but this is statistically unlikely for domains with many grid points. If the fractal method were applied to a full four-dimensional concentration field (three spatial dimensions and one temporal dimension), the total mass of the plume could be scaled to enforce mass conservation. A three-dimensional concentration field (two spatial dimensions and one temporal dimension) does not have a requirement for mass conservation; presumably, the plume also fluctuates in the vertical dimension, thereby changing the total mass represented in the two-dimensional spatial slice as it evolves through time.

The eddy structures that manifest in turbulent flow exhibit vorticity that is not replicated by the fractal method.[39] The vorticity of fluctuations result in spatial correlations that are a function of both distance and direction. Although the fractal method replicates the spatial correlations as a function of distance, spatial correlations that are a function of distance *and* direction may not be accurately represented. This is unlikely to significantly affect the modeling of arrays of independent detection systems, but could cause inaccurate

[39] Philip J.W. Roberts and Donald R. Webster, "Turbulent Diffusion," chap. 1 in *Environmental Fluid Dynamics: Theories and Applications,* eds. Hayley H. Shen, Alexander H.D. Cheng, Keh-Han Wang, Michelle H. Teng, Clark C. Liu, (Reston, VA: ASCE Publications, 2002).

behavior of linked sensor arrays that have some directional dependence in their response algorithms.

The fractal method can be computationally intensive, particularly when the spatial or temporal region of interest is large. Unlike the Markov method, the fractal method computations scale linearly with the size of the simulation's spatiotemporal domain, rather than the number of detectors. This limitation may be mitigated if the entire concentration field is not needed (for example, when only specific detector locations are of interest). If the distance between detectors is greater than the initial grid spacing, the fractal method does not permit any spatial correlation in the concentration fluctuations at different detectors. In such a case, it is only necessary to simulate the fluctuations in the region local to each detector, rather than the entire spatial domain. In such an implementation, the computations would scale with the number of detectors. A greater computational problem may be presented by the exponential increase in computations with additional spatiotemporal grid refinements. If the detector's sampling rate and detection algorithm warrant simulating fluctuations at very short time scales, the computational burden of the fractal method may become untenable.

Detailed information regarding the spatial and temporal correlations in concentration fluctuations is needed to parameterize the fractal method. Presumably, experiments leveraging a dense grid of high frequency sensors would be needed to adequately characterize the spatial and temporal correlations. The correlations in concentration fluctuations may be highly dependent on the specific nature of the release, terrain, and meteorological conditions, so broad application of generic parameters may not be possible and extensive experimental data across a range of conditions may be needed to support the parameterization and use of the fractal method.

4. Generation of Field Trial Ensembles

Field experiments involving outdoor releases of chemical agent simulants offer a wealth of data by which the methods of simulating concentration fluctuations can be evaluated. Individual field trials conducted under similar conditions can be grouped to construct ensembles from which the concentration mean and variance fields can be computed. To be useful for this analysis, the trials must have employed fast-response sensors to capture concentration fluctuations. Ideally, the sensor array in the trials would be large enough to encompass most of the disseminated plume and dense enough to characterize the shape of the plume. Also, having a large number of trials conducted under similar release and meteorological conditions allows for the creation of larger ensembles, which leads to more meaningful ensemble statistics.

A. Selection of Field Trial Data

Several field experiment datasets are described in the literature, including Project Prairie Grass (1956),[40,41] CONFLUX experiments (1990s),[42] Mock Urban Setting Test (2001) (MUST),[43] Fusion Field Trial (2007) (FFT 07),[44] and Project Sagebrush (2013).[45] Each of these datasets was investigated for use in the evaluation of the Markov and fractal methods.

[40] Morton L. Barad, *Project Prairie Grass, A Field Program in Diffusion: Vol. I*, AFCRC-TR-58-235(I) GRP-59-VOL-1 (Hanscom AFB, MA: Air Force Cambridge Research Labs, 1958), 280.

[41] Morton L. Barad, *Project Prairie Grass, A Field Program in Diffusion: Vol. II*, AFCRC-TR-58-235(II) GRP-59-VOL-2 (Hanscom AFB, MA: Air Force Cambridge Research Labs, 1958), 209.

[42] Christopher A. Biltoft, *Concentration Fluctuation Modeling of Chemical Hazards (Assess Vulnerability)*, (Dugway, UT: U.S. Army Dugway Proving Ground, 1993).

[43] Christopher A. Biltoft, *Customer Report for Mock Urban Setting Test*, DPG Document No. WDTC-FR-01-121 (Dugway, UT: West Desert Test Center, U.S. Army Dugway Proving Ground, 2001).

[44] D.P. Storwald, *Detailed Test Plan for the Fusing Sensor Information from Observing Networks (Fusion) Field Trial (FFT-07)*, Tech. Rep. Document No. WDTC-TP-07-078 (Dugway, UT: Meteorology Division, West Desert Test Center, U.S. Army Dugway Proving Ground, 2007).

[45] D. Finn et al., "Project Sagebrush Phase 1," NOAA Technical Memorandum OAR ARL-268 (College Park, MD: National Oceanic Atmospheric Administration, 2015).

Project Prairie Grass was conducted in 1956 to determine the rate of diffusion of a sulfur dioxide tracer as a function of different meteorological conditions.[46] Sensors were arranged in arcs at a range of 50, 100, 200, 400, and 800 meters from the release.[47] The sensors were designed to measure the accumulated dose, from which a single average concentration per release was calculated. These data, therefore, have low time resolution, and thus are unsuitable for evaluating methods of simulating fluctuating concentrations.[48]

Data from the CONFLUX experiments were not available and may be lost.[49]

The MUST dataset was designed to test dispersion in an urban environment using shipping containers to simulate city blocks.[50] The project included 63 continuous releases and 5 multiple puff releases, resulting in 16 hours of continuous releases and 4.75 hours of puff release data, all timed for relatively similar meteorological conditions.[51] Seventy-four fast response samplers—48 digital photoionization detectors (DPIDs) and 26 ultraviolet ion collectors (UVICs), both at 50 Hz, and 22 sonic anemometers to measure wind speed at 20 Hz—were used to sample a propylene tracer.[52] To avoid complicating the present analysis with the effects of urban dispersion on concentration fluctuations, the MUST dataset was not leveraged.

FFT 07 was designed with the specific goal of providing a data set for the testing and evaluation of source term estimation algorithms as part of the Sensor Data Fusion project for the Defense Threat Reduction Agency (DTRA). Instantaneous and continuous releases were simulated in daytime and nighttime conditions. The use of fast response DPIDs, along with the minimal variation of release parameters—including flow rate, height of release, and duration of release—between many of the numerous trials in the project make this dataset particularly suitable for evaluating the methods for simulating concentration fluctuations. Hence, the FFT 07 data were deemed acceptable for this analysis.

Project Sagebrush was designed to revisit Project Prairie Grass and included 60 two-hour trials, including 150 bag samplers (10-minute sampling intervals) and 6 fast response samplers (0.5 second intervals) using a SF_6 tracer.[53] Wind data and stability categories were

---

46 Barad, *Project Prairie Grass, A Field Program in Diffusion: Vol. I*, AFCRC-TR-58-235(I) GRP-59-VOL-1, 1-3.

47 Ibid., 62.

48 Ibid., 57.

49 Eugene Yee, email communication to the authors, 6 July 2017.

50 Christopher A. Biltoft, *Customer Report for Mock Urban Setting Test*, DPG Document No. WDTC-FR-01-121 (Dugway, UT: West Desert Test Center, U.S. Army Dugway Proving Ground, 2001).

51 Biltoft, *Customer Report for Mock Urban Setting Test*, DPG Document No. WDTC-FR-01-121.

52 Ibid.

53 Finn et al., "Project Sagebrush Phase 1," NOAA Technical Memorandum OAR ARL-268.

measured every five minutes.[54] The small number of fast response samplers limits the utility of this dataset, so it was not leveraged for this analysis.

Among the field campaigns considered in this analysis, only FFT 07 had sufficient fast-response concentration sensor data for open-terrain (i.e., not urban) dispersion. Hence, the FFT 07 data were selected to evaluate the Markov chain and fractal methods of simulating concentration fluctuations.

B. Fusion Field Trial 2007 Data Description

The FFT 07 project included 82 15-20 minute trials using a 475 meter x 450 meter grid of 100 high-frequency (50 Hz) DPID samplers for a propylene tracer, as well as 20 UVIC detectors.[55] For this analysis, only the data from the DPIDs were available. High-frequency wind, humidity, and temperature data were measured every 10 seconds from 50 meteorological stations. Figure 3 shows the locations of the concentration samplers (left pane) and meteorological sensors (right pane).

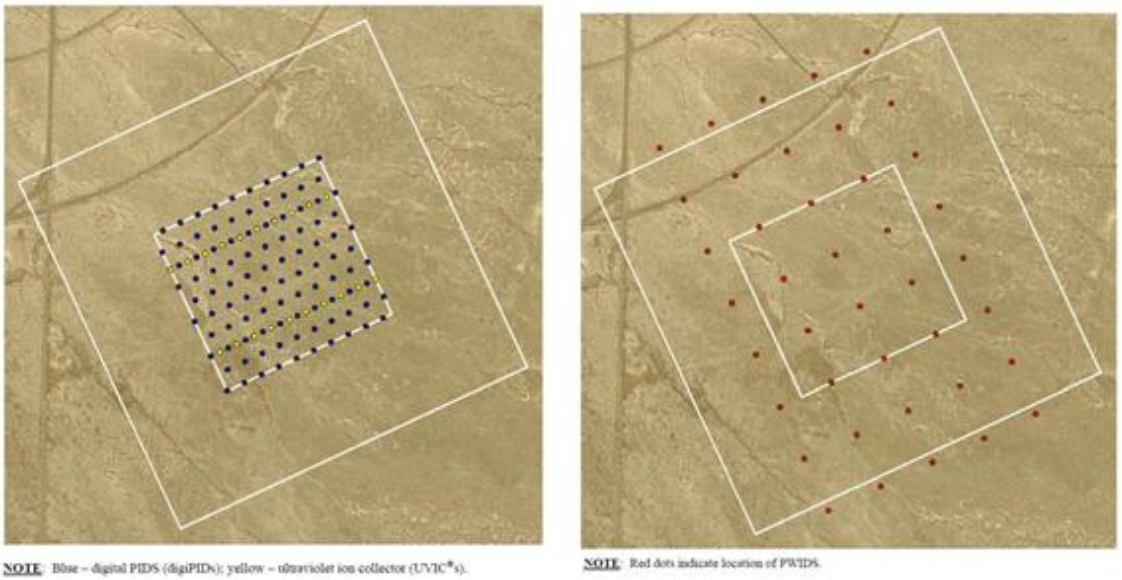


Figure 3. Sensor Grid Layout for Fusion Field Trial; Left Pane: Concentration Sensors; Right Pane: Meteorological Sensors

Although the FFT 07 release locations were changed from one trial to the next to satisfy the original purpose of creating a dataset to test source-locating algorithms, the release locations were all near one another on the same side of the grid, due to the prevailing

54 Finn et al., "Project Sagebrush Phase 1," NOAA Technical Memorandum OAR ARL-268.

55 Nathan Platt, Steve Warner, and Steve M. Nunes, "Evaluation Plan for Comparative Investigation of Source Term Estimation Algorithms Using FUSION Field Trial 2007 Data," *Hrvatski Meteorološki Časopis* 43, no. 43/1 (2008), 224-229.

winds in the area. Figure 4 shows an illustrative wind rose with a typical arc of wind directions over the course of a single FFT 07 release. All of the average wind vectors for the trials used herein are included in Table A–1 in Appendix A.

Concentration data are included for one minute before the beginning of each release and for five minutes after the end of each release. All continuous releases were 10 minutes long, so 16 minutes of data at 50 Hz are available for each DPID sensor in each continuous release trial. Note that some data are missing, because some sensors malfunctioned during some releases. Logs are also included and detail the location, flow rate, and number of individual release locations, among other parameters, for each trial.

All of the data from FFT 07 underwent an intensive quality control (QC) process and QC flags are included for each data point, both for concentrations and meteorology.[56] These flags differentiate good data from suspect, bad, or missing data.

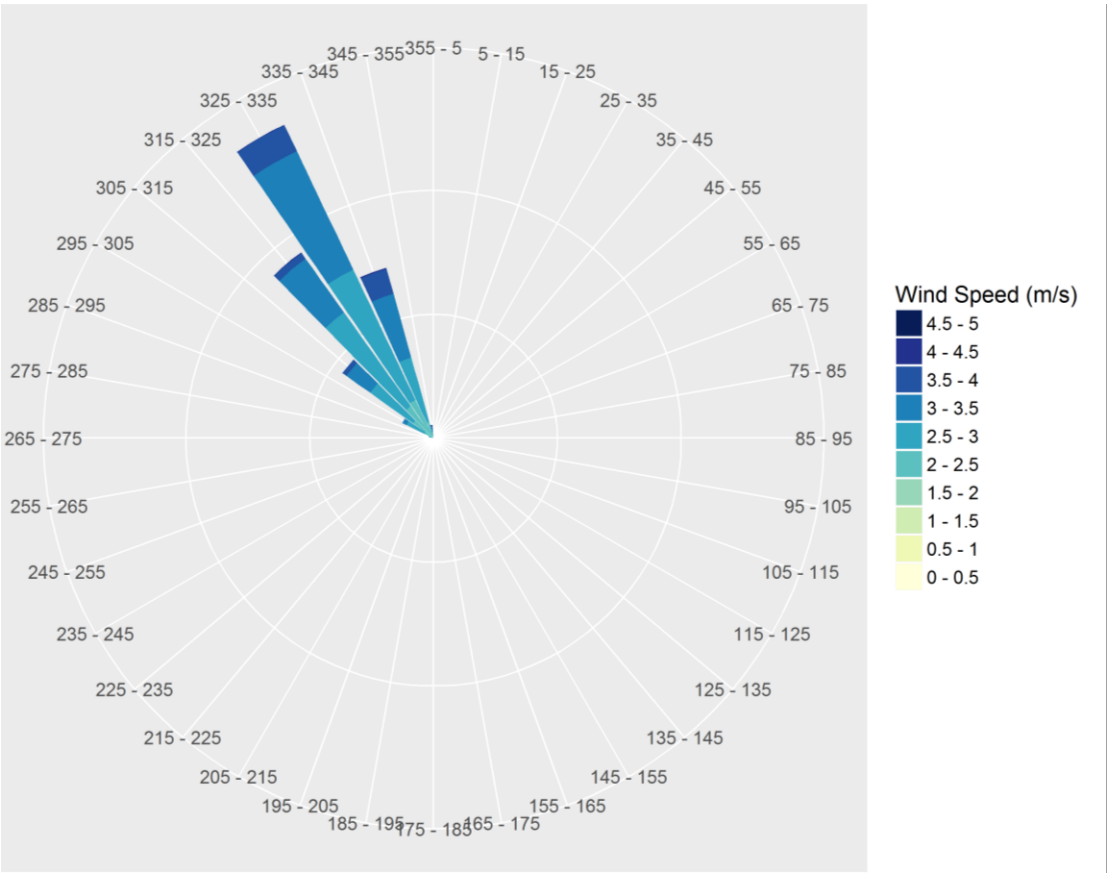


Figure 4. An Illustrative Wind Rose from FFT 07: Trial 54

[56] Platt, Warner, and Nunes, "Evaluation Plan for Comparative Investigation of Source Term Estimation Algorithms Using FUSION Field Trial 2007 Data," 224-229.

C. Standardizing Fusion Field Trial 2007 Data

Data from FFT 07 consist of release logs, meteorological sensor readings, and concentration sensor readings. Together, these data sources constitute the full trial data that are then aggregated into ensembles. Any trials that were aborted prematurely, are missing a significant portion of concentration data, or lack a release log were eliminated from consideration. Also, the wind during each continuous release (10 minutes) was examined for each trial to ensure that the wind vector at each time step, averaged across all of the meteorological sensor stations, pointed toward the sensor field for the entire duration of the release. The trials that were removed from consideration for one or more of the reasons listed above are 1, 2, 4, 13, 14, 15, 17, 61, and 64, leaving a total of 73 trials for the analysis.

Next, concentrations and meteorology were read and formatted appropriately for each trial. The release log was used to determine the starting time for each release, and that time was used to reset the time series for both concentration and meteorological sensors so that the release time corresponded to 00:00:00. Note that the times for concentration and meteorological data were in Coordinated Universal Time (UTC) and release logs were in local time (UTC-6). Meteorological sensor readings occurred once every 10 seconds, but many of these readings were deemed suspicious or possibly erroneous. To rectify this, any readings that were marked with any of the project's rigorous quality control flags, including suspect data, bad data, or missing data, were removed from the data set. Concentration sensor readings were output every 20 milliseconds, which were averaged to 1 second intervals to create a more manageable dataset. The necessary temporal resolution depends on the detector algorithm that will ultimately be modeled. One-second resolution is adequate for this analysis because, as described in section 7.B, this analysis used detectors that required upcrossing durations of 5 or 60 seconds.

To accurately represent intermittency in the concentration data, it is necessary to remove any background concentration or sensor bias in the DPID sensors. The sensor readings for the minute before each release were used as a baseline for that sensor's output for zero concentration for the given release. The mean and standard deviation of this baseline were calculated, and any sensor readings lower than the low end of the operational range of the DPID sensors (6×10^{-8} kg/m^3)[57] or lower than three standard deviations above the mean of the baseline were set to zero. Finally, although some concentration sensors in some trials continued recording for more than 15 minutes after the beginning of the release, the majority of sensors shut down at that point; hence, all sensor readings were truncated at the 15-minute mark so that each trial was equal in length across every sensor.

[57] Storwald, *Detailed Test Plan for the Fusing Sensor Information from Observing Networks (Fusion) Field Trial (FFT-07)*, Tech. Rep. Document No. WDTC-TP-07-078.

D. Grouping Fusion Field Trial 2007 Trials into Ensembles

Trials are grouped into ensembles based on their release characteristics and meteorological conditions.[58] Only continuous release trials were leveraged in this analysis. First, the trials were split into four groups corresponding to the number of simultaneous release locations used in the trial. A mean wind vector was then determined for each trial by first averaging wind speed and direction (speed averaged as a scalar, direction as a vector) across all sensors at each time step, then taking the time average to get an overall mean wind vector for the 15-minute trial. The trials are grouped into ensembles, such that the mean wind values differ no more than about 1.5 m/s and are within a 90 degree or smaller directional arc.[59] In this way, five ensembles were determined, which are briefly described in Table 1. Appendix A includes the properties of every release in each ensemble. As shown in Table 1, the number of trials in each ensemble is limited.

Table 1. Description of Ensembles Created Using FFT 07 Data

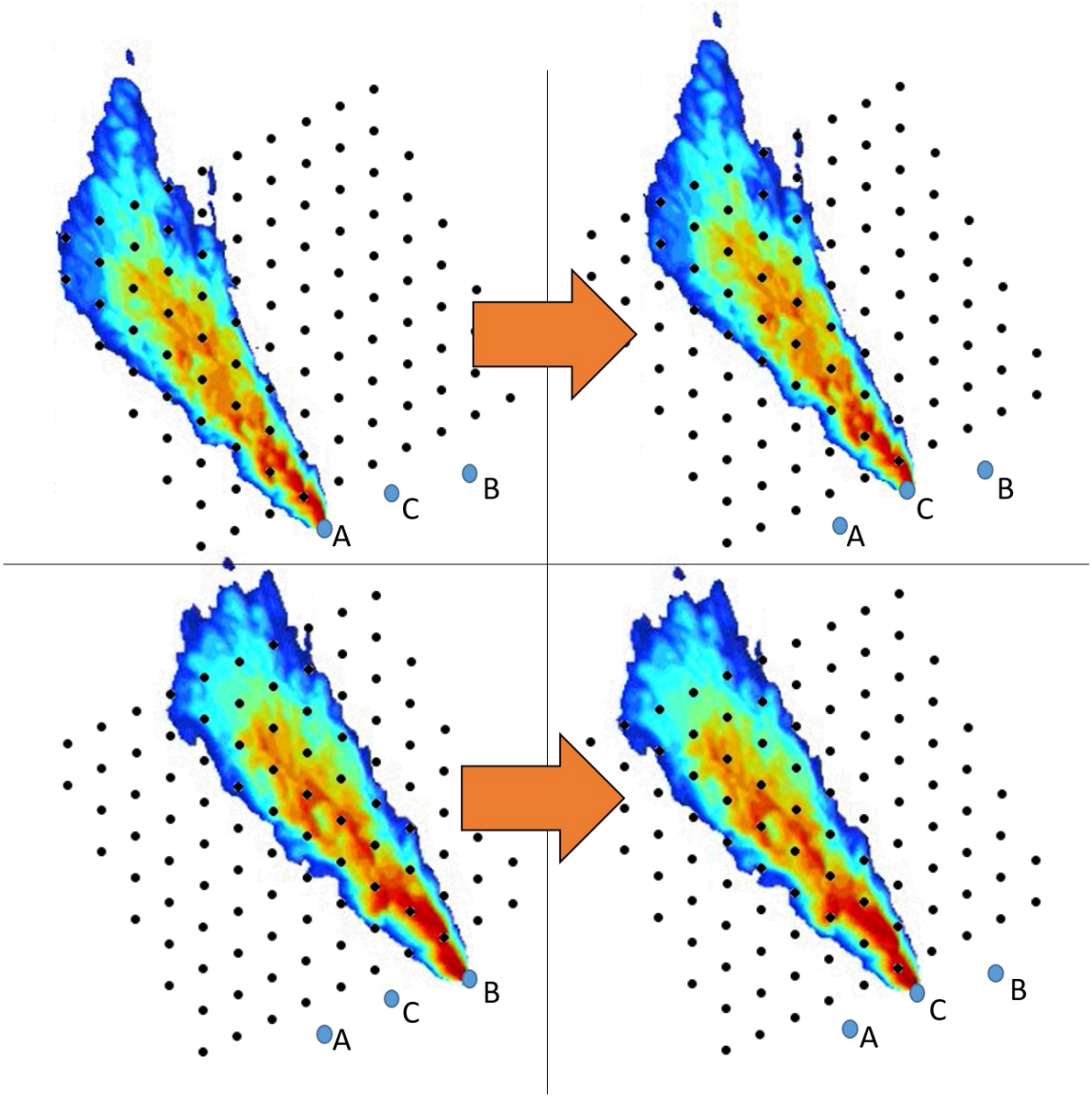
Ensemble	Number of Release Locations	Number of Trials	Range of Wind Speeds (meters/second)	Range of Wind Directions (degrees from North)
1	1	14	1.39–2.88	105–167
2	2	5	1.19–2.33	120–147
3	2	6	2.70–3.24	128–157
4	3	8	1.60–2.82	110–172
5	4	3	1.52–2.84	122–144

In trials with multiple releases, the release locations are fixed relative to one another, but the coordinates of the releases relative to the sensor grid may vary between trials. Hence, the sensor grid for each trial was shifted to align the release locations within each ensemble. The necessary offset was found by averaging the latitude and longitude of the west-most release location in each trial and determining the difference vector required to bring the release location to that coordinate. The sensor grid for each trial was shifted by the respective difference vector, and was then overlaid onto the static ensemble grid, which was based on the mean release location. This process is illustrated for two trials in Figure 5. Each sensor in the ensemble grid then uses the average of the nearest four trial grid sensor readings, inversely weighted by distance, as its concentration. If data were

[58] Ideally, all meteorological conditions would be considered when creating ensembles. Due to the limited number of continuous trials in FFT 07 with the same number of releases, only wind speed and wind direction were considered.

[59] These constraints, especially the acceptable directional arc, allow a significant amount of variation within an ensemble. Ideally, this would be more limited, but to have a reasonable number of trials in each ensemble, the similarity of the trials had to be sacrificed to some degree.

missing at one or more of the nearest four trial grid sensors and that sensor or sensors would not contain more than 25% of the ensemble grid sensor's concentration data according to the weighting scheme, then the closest three, or even two, nearest trial grid sensors were averaged instead. This process ensures that each trial's concentration data is translated into a single ensemble coordinate system with minimal loss of data.



Note: The grid is shifted for each release, such that initial release points A and B (top left and bottom left) are shifted to the same release location at point C (top right and bottom right).

Figure 5. Illustration of Plume Shifting to Average Release Coordinate for Ensemble Generation

Because the disseminator flow rate may differ between trials within an ensemble, the concentration data for each trial was normalized to the mean flow rate across all of the trials in the ensemble. This scaling assumes that the increase of concentration at each point is linearly dependent on the release mass, which is consistent with various solutions to the advection-diffusion equation.[60] Finally, for each of the 100 sensors and each time step, the concentration mean and variance are calculated across all trials in an ensemble. As variance requires at least two trials, any sensor that had its mean concentration calculated from only a single trial at any time, due to malfunction or loss of data, is excluded from the ensemble data set for those times.[61]

[60] T. Tirabassi, "Solutions of the Advection-Diffusion Equation," *Transactions on Ecology and the Environment* 21, (1997).

[61] Preferably, many trials would be used to calculate the mean and variance, but given sensor malfunctions or loss of data and the small number of trials in each ensemble, this analysis included the mean and variance for any sensor location and time that had data from at least two sensors.

5. Simulation of a Field Trial in HPAC

In the absence of field trial data for a scenario, the methods of simulating concentration fluctuations would be applied to mean and variance data calculated by HPAC, through SCIPUFF, at discrete detector locations. As such, the ability of HPAC to accurately represent mean and variance data directly affects the fidelity of simulated concentration fluctuations. A single trial from FFT 07 was emulated in an HPAC simulation to produce outputs on which methods for simulating concertation fluctuations were then applied.

Trial 54 from the FFT 07[62] data set was chosen based on several factors: (1) the trial uses only a single release location; (2) the trial has a middling mass release rate; (3) sensors on the crosswind edges of the grid reported no nonzero concentration data, indicating that the plume was completely bounded by the grid in the crosswind direction; and (4) 20 realizations of the trial had already been calculated on large-eddy computational fluid dynamics (CFD) software for a previous study.[63] Selecting this specific field trial to simulate in HPAC allows the results from the Markov and fractal methods to be evaluated in the context of results from the significantly more complex large-eddy CFD model. HPAC simulations were performed, in which concentration samplers and release locations were initialized at the locations of the DPID sensors and releases reported in the FFT 07 literature. All parameters for the release that were changed from their defaults in HPAC are listed in Table 2. The disseminator flow rate was determined by converting the given flow rate for the trial (in standard liters per minute) to kilograms per second using propylene's density and the air pressure recorded at the site:

$$mass\ flow\ rate = volumetric\ flow\ rate * density * local\ air\ pressure$$

with density = 1.81 kg/m^3 [64] and local air pressure in atmospheres. The above calculation assumes, based on the units of measure given in the data, that standard liters per minute are

---

62 Storwald, *Detailed Test Plan for the Fusing Sensor Information from Observing Networks (Fusion) Field Trial (FFT-07)*, Tech. Rep. Document No. WDTC-TP-07-078.

63 Nathan Platt, Dennis DeRiggi, Steve Warner, Paul Bieringer, George Bieberbach, Andrzej Wyszogrodzki, and Jeffrey Weil, "Method for Comparison of Large Eddy Simulation Generated Wind Fluctuations with Short-Range Observations," *International Journal of Environment and Pollution* 48, no. 1-4 (2012).

64 PubChem, "Propylene," accessed August 31, 2018, https://pubchem.ncbi.nlm.nih.gov/compound/Propene.

recorded at standard temperature and pressure and that the change in temperature from standard at the release site is negligible.

Table 2. HPAC Parameters Used To Recreate FFT 07 Trial 54

Parameter	Value
Maximum time step	30 seconds
Surface roughness	0.03 meters[A]
Material	PROPYLENE_GAS
Release duration	600 seconds
Flow rate	0.0095 kg/s
Release height	2 meters
Active fraction	1
Droplet distribution	Vapor Phase
Conditional averaging	600 seconds

[A] Christopher A. Biltoft, Shayes D. Turley, T. B. Watson, G. H. Crescenti, and R. G. Carter, *Over-Land Atmospheric Dispersion (OLAD) Test Summary and Analysis*, WDTC-FR-99-016 (Dugway, UT: U.S. Army Dugway Proving Ground, 1999).

The meteorological data, taken from the 40 meteorological stations used in FFT 07 at 10-second intervals, was used in its entirety[65] over the 15-minute HPAC simulation by setting the weather observations time bin and SWIFT wind field update interval parameters to match the meteorological sampling rate of 10 seconds. Accordingly, Large Scale Variability was turned off in the weather settings in HPAC, as turbulent velocity fluctuations should already be accounted for in the well-resolved meteorological data. Finally, the Bowen ratio and albedo of the terrain were set to the HPAC defaults for a dry desert, and the surface roughness was set to 0.03 meters to mimic the flat desert terrain.

Instantaneous concentration samplers were instantiated in HPAC at each of the DPID sensor locations used for FFT 07 at a height of 2 meters to match their real counterparts. These samplers output mean concentration, concentration variance, and the temporal correlation scale for each sampler at each time step. A brief analysis of various HPAC time steps, from 1 to 30 seconds, revealed little difference in the evaluation metrics for this release, as long as the basic structure of the concentration time series was preserved, so the HPAC time step was set to 30 seconds for expedience.

[65] The meteorological data included wind speed, wind direction, temperature, relative humidity, and pressure.

6. Application of Concentration Fluctuation Models

The Markov chain method and the fractal method for calculating concentration fluctuations were directly applied to the concentration mean and variance calculated from five field trial ensembles. The application to field trial ensembles allows these methods to be directly evaluated without complications introduced by the involvement of transport and dispersion models, though it does introduce possible complications and uncertainties from other sources. Because these methods would be coupled with atmospheric transport and dispersion models in practice, they were also applied to mean and variance estimates from an HPAC simulation of an individual field trial. Numerous fluctuating concentration realizations were simulated with each method, so that the evaluation of the methods could be based on their limiting behavior and not the specific random outcome of a single realization.

A. Application of Markov Method

One hundred realizations were created via the Markov chain process for both the field trial ensembles and HPAC outputs. The values of the metrics showed negligible change when increasing from 90 to 100 realizations. A constant temporal correlation scale was used across the entire duration of the simulation when applied to field trial ensembles, whereas the variable HPAC temporal correlation scale output was used for application to HPAC outputs.[66]

For application to the HPAC outputs, each of the 30-second output intervals was assumed to be a quasi-stationary state. The transcendental equations that stem from the concentration PDF were solved once for each HPAC time step, and the individual realizations at each HPAC sampler were allowed to evolve at a smaller fixed time step within each HPAC time step. Intervals between 200 milliseconds and 1 second were considered for the smaller time steps within each quasi-stationary HPAC time step with little observed difference. As such, the Markov process was simulated over one-second time steps within each quasi-stationary HPAC time step to more closely approximate the minimum time scale of fluctuations that was used in the fractal method.

[66] While a dynamic temporal correlation scale is preferable, only a constant temporal correlation scale is calculable for the field data because its empirical calculation requires leveraging the entire empirical concentration time series. (See Appendix B).

B. Application of Fractal Method

One hundred realizations were simulated using the fractal method for both the field trial ensembles and the HPAC outputs. As with the Markov method, 100 realizations were deemed sufficient because the values of the computed metrics were negligibly different from those computed across only 90 realizations. The spatial and temporal correlation scales for each field trial ensemble were used to initialize the spatiotemporal grid spacing. The calculation of the spatial and temporal correlation scales used for each field trial ensemble is described in Appendix B. The spatiotemporal grid was iteratively refined four times until the temporal grid spacing was one to two seconds. Although additional grid refinements simulate higher frequency fluctuations, further refinements presented a significant computational burden. The fractal dimension was selected to be 1.35 because that is the middle of the range, 1.30-1.40, found to characterize various atmospheric phenomena.[67,68,69,70,71]

Although HPAC provides a variable temporal correlation scale output, the current implementation of the fractal method can only use a constant temporal correlation scale to initialize the temporal spacing of the grid. For application to HPAC outputs, the initial spatial grid spacing was 25.84 meters and the initial temporal grid spacing was 23.71 seconds. This initial spatiotemporal grid spacing reflects the average spatial and temporal correlation scales across the five field trial ensembles. Again, the spatiotemporal grid was iteratively refined four times until the temporal grid spacing was one to two seconds.

No attempt was made to model the bulk translation of coherent structures using the fractal method. Due to the sensor array not completely encompassing the plume and the relatively low numbers of field trials in each ensemble, the mean concentration centroids of the field trial ensembles did not travel in a direction that was consistent with the mean wind field. Hence, the erratic movements of the mean field centroid were not considered representative of the bulk movement of the dispersed material in the plume. A sensor grid larger than the dimensions of the plume is needed to accurately replicate the bulk translation of the plume using this method. If the sensor grid were large enough to fully encompass the plume, then inclusion of this approach to handling the bulk translation is expected to increase the frequency and decrease the duration of the modeled fluctuations

---

67 Sykes, Gabruk, and Henn, "The Small-scale Structure of Dispersing Clouds in the Atmosphere," A.R.A.P. Report No. 710.

68 Sreenivason and Meneveau, "The Fractal Facets of Turbulence," 382.

69 Lewellen and Sykes, "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes."

70 Lovejoy and Mandelbrot, "Fractal Properties of Rain, and a Fractal Model."

71 Sykes and Gabruk, "Fractal Representation of Turbulent Dispersing Plumes."

at any given stationary sensor location. Further, application of this method may increase the spatial correlation between sensors in the direction of the wind.

This page is intentionally blank.

7. Evaluation Metrics

The metrics used to evaluate each method of simulating concentration fluctuations were selected to be directly relevant to modeling detector response. Simple models of detector response are often characterized by a threshold concentration and required duration that the concentration must exceed the threshold.[72] Therefore, the metrics in section 7.A focus on quantifying upcrossing events that occur for a given threshold, i.e., events in which the concentration rises above a specified threshold. Section 7.B provides results for operational metrics that emulate analyses of detector performance.

In the calculation of each of the following metrics, the concentration time series at each sensor in the ensemble is truncated at the first and last nonzero value. Consequently, each sensor has a different length time series. This was performed to exclude times in which the plume was not physically present in the general vicinity of a sensor, i.e., before the plume traveled far enough to reach a given sensor location, and times after the plume is no longer present at that sensor location. This also effectively excluded sensors that did not record concentrations above the DPID detection limit.

A. Evaluation Metrics for Comparison with Field Trial Ensembles

For the application to field trial ensembles, the intermittency, upcrossing frequency, and upcrossing duration were the metrics used to evaluate the methods of simulating concentration fluctuations. For these metrics, all concentration time series in the ensemble for all sensor locations were included in the summations. Intermittency was selected as a metric because it is ubiquitous in the literature. Upcrossing frequency and duration were employed because they directly examine the fluctuations that need to be reproduced when modeling detector operation.

1. Intermittency

Intermittency is the fraction of time at each point within the agent cloud that concentration is nonzero.[73]

---

[72] Alison E. Lawrence, Forrest R. Smith, John Alex Vig, and Charles E. Snyder, *User Manual for the Chemical and Biological Attack Consequence Estimator Version 1.1*, IDA Document D-8505 (Alexandria, VA: Institute for Defense Analyses, November 2017).

[73] Albert A. Townsend, *The Structure of Turbulent Shear Flow* (Cambridge, UK: Cambridge University Press, 1980), 211.

$$i = \frac{\sum t_{c\neq 0}}{\sum t_{all}} \tag{32}$$

Here, $t_{c\neq 0}$ is the total time that the concentration is nonzero and t_{all} is the total time of the between the first and last nonzero concentration. Recall that all recorded concentrations were clipped at the DPID detection limit, so the intermittency properly represents the fraction of the time that the concentration was actually measurable by a given DPID sensor while the agent cloud was present in its general vicinity.[74] Intermittency has been used to quantitatively characterize concentration fluctuations.[75,76]

2. Concentration Upcrossing Frequency

The upcrossing frequency, which has been used as a metric to characterize concentration fluctuations previously, is the average rate at which the concentration at a given sensor location rises above a threshold.[77,78,79,80,81] For a given location, the upcrossing frequency, *f*, can be expressed as:

$$f = \frac{n_{upcrossing}}{\sum t_{all}}. \tag{33}$$

As before, t_{all} is the total time of the truncated time series and $n_{upcrossing}$ is the total number of times the concentration rises above a particular concentration threshold, i.e., the number of upcrossings. The number of upcrossings is directly dependent on the

---

74 Here, the DPID detection threshold is considered the maximum of the nominal detection threshold and three standard deviations greater than the mean recorded concentration over the interval one minute prior to the release.

75 Steven R. Hanna, "Concentration Fluctuations in a Smoke Plume," *Atmospheric Environment (1967)* 18, no. 6 (1984), 1091-1106.

76 Hilderman and Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," 451-482.

77 L. Kristensen, J. C. Weil, and J. C. Wyngaard, "Recurrence of High Concentration Values in a Diffusing, Fluctuating Scalar Field," *Boundary Layer Studies and Applications* (Dordrecht, Netherlands: 1989), 263-276.

78 Eugene Yee, P. R. Kosteniuk, G. M. Chandler, C. A. Biltoft, and J. F. Bowers, "Recurrence Statistics of Concentration Fluctuations in Plumes within a Near-Neutral Atmospheric Surface Layer," *Boundary-Layer Meteorology* 66, no. 1-2 (1993), 127-153.

79 Eugene Yee, R. Chan, P.R. Kosteniuk, G.M. Chandler, C.A. Biltoft, and J.F. Bowers, "Measurements of Level-Crossing Statistics of Concentration Fluctuations in Plumes Dispersing in the Atmospheric Surface Layer," *Boundary-Layer Meteorology* 73, no. 1-2 (1995), 53-90.

80 Shuming Du, David J. Wilson, and Eugene Yee, "A Stochastic Time Series Model for Threshold Crossing Statistics of Concentration Fluctuations in Non-Intermittent Plumes." *Boundary-Layer Meteorology* 92, no. 2 (1999), 229-241.

81 Hilderman and Wilson, "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives," 451-482.

concentration threshold used to define an upcrossing event. Therefore, the upcrossing frequency at each sensor location is a function of the concentration threshold.

3. Concentration Upcrossing Duration

The upcrossing duration is the average length of time that the concentration remains above a given concentration threshold and has been used as a metric characterizing concentration fluctuations previously.[82] A high upcrossing frequency and low upcrossing duration would indicate a quickly fluctuating concentration, while a low upcrossing frequency with a low upcrossing duration would indicate a few spikes in concentration. A very high upcrossing duration is expected only with a low upcrossing frequency, and would indicate that the concentration infrequently falls below the given threshold. The upcrossing duration, *d*, is calculated as:

$$d = \frac{\sum t_{c>threshold}}{n_{upcrossing}} \tag{34}$$

Here, $t_{c>threshold}$ is the total time that the concentration exceeds a given threshold. Hence, the upcrossing duration at each sensor location is also a function of the concentration threshold.

B. Evaluation Metrics for Applying the Models to an HPAC Simulation

A simple model of detector response was implemented to obtain the evaluation metrics for the application of the methodologies to the HPAC simulation outputs. These evaluation metrics relate to detector alarm events in which a threshold concentration is exceeded for a specified duration, termed the "detection duration." This approach is particularly informative because the metrics are directly relevant to analyses in which models of detector response are used to evaluate detector performance in operational environments.

1. Total Alarm Count

The total alarm count reflects the total number of upcrossing events in which the upcrossing exceeded the detection duration. Multiple alarms from the same detector count. This metric is summed across all detector locations and was calculated for two detection durations: 5 seconds and 60 seconds. The 5-second detection duration is consistent with

[82] Yee et al., "Measurements of Level-Crossing Statistics of Concentration Fluctuations in Plumes Dispersing in the Atmospheric Surface Layer," 53-90.

the sampling rate of typical chemical detectors,[83] while the 60-second detection duration is intended to typify biological detectors.[84]

2. Number of Alarming Detectors

The number of alarming detectors reflects the number of detector locations at which an upcrossing event exceeded the detection duration at least once. In contrast to the total alarm count metric, only one alarm from a given detector is counted. Operationally, repeated alarms over a short interval from the same detector are not regarded as any more informative than a single alarm from a given detector, as personnel would presumably respond to the first detector alarm. Thus, this metric is more relevant to how personnel may respond to detector alarms than the total number of alarms metric. Because the detectors are located on a grid, this metric is proportional to the land area over which detectors might detect a release. This metric was also calculated for two detection durations: 5 seconds and 60 seconds.

---

[83] The Joint Chemical Agent Detector (JCAD) has a requirement for a five-second sampling rate while in monitor mode. *Capability Production Document for Joint Chemical Agent Detector Increment: I* (Washington, DC: Joint Requirements Office for CBRN Defense, 4 June 2008).

[84] In chamber experiments, many detectors alarm within 60 seconds of exposure to agent. As a result, biological detectors have been previously modeled with a 60-second detection duration.

Alison E. Lawrence, Scott L. Weinrich, Robert L. Cubeta, and John A. Vig, *Interpretation of Joint Biological Tactical Detection System (JBTDS) Requirements*, IDA Paper P-4870 (Alexandria, VA: Institute for Defense Analyses, 2013). SECRET//NOFORN (Only Unclassified Data Used Herein).

Alison E. Lawrence, Scott L. Weinrich, Jeffrey H. Grotte, Douglas P. Schultz, Nafis J. Upshur, W.M. Christensen, Christina M. Patterson, Eleanor L. Schwartz, Megan L. Prophet, and John N. Bombardt, *Joint Biological Tactical Detection System Analysis of Materiel Alternatives: Phase II*, IDA Paper P-4624 (Alexandria, VA: Institute for Defense Analyses, March 2011). SECRET//NOFORN (Only Unclassified Data Used Herein).

8. Results

As a basis for assessing the fidelity of each concentration fluctuation simulation method, the evaluation metrics were computed for the simulated fluctuating concentration time-series compared to the actual field trial time series, using the metrics described in section 7.A. This comparison provides insight into the nature of the simulated concentration fluctuations, but does not directly indicate how incorporation of these methods may affect analyses in which they are coupled with transport and dispersion models. Hence, evaluation metrics were obtained from applying these methods to HPAC outputs, as described in section 7.B, to give a practical sense of how these methods may perform in analyses in which detector response is modeled.

A. Results of Applying the Models to Field Trial Ensembles

The concentration intermittency was computed for each sensor location independently across each of the 100 realizations that were simulated by each method. After intermittency had been calculated for each simulated data set, the intermittency was similarly computed independently from the FFT 07 data for each location across all of the field trials in a given ensemble. Figure 6 and Figure 7 directly compare each sensor's demonstrated intermittency from the field trials with its simulated counterpart. Thus, for the ideal model, each point would lie on the diagonal, depicted with a dotted line in the figures. Note that these plots are intended to be exploratory, and later results, which employ a factor of two criterion, provide a more critical evaluation of the ability of these methods to reproduce various metrics of the field trial data. Results produced for Ensembles 1 and 3 are featured, because Ensemble 1 is composed of the most trials, and meteorological conditions were the most similar across the trials in Ensemble 3. Results for Ensembles 2, 4, and 5, which follow similar trends to those shown here, can be found in Appendix C. Overall, these results show that the Markov method generally under predicts intermittency, while the fractal method has less bias but a lot of scatter, though Figure 7 shows that the fractal method has a tendency to erroneously predict intermittencies between 0.8 and 1 for many of the sensors in Ensemble 3.

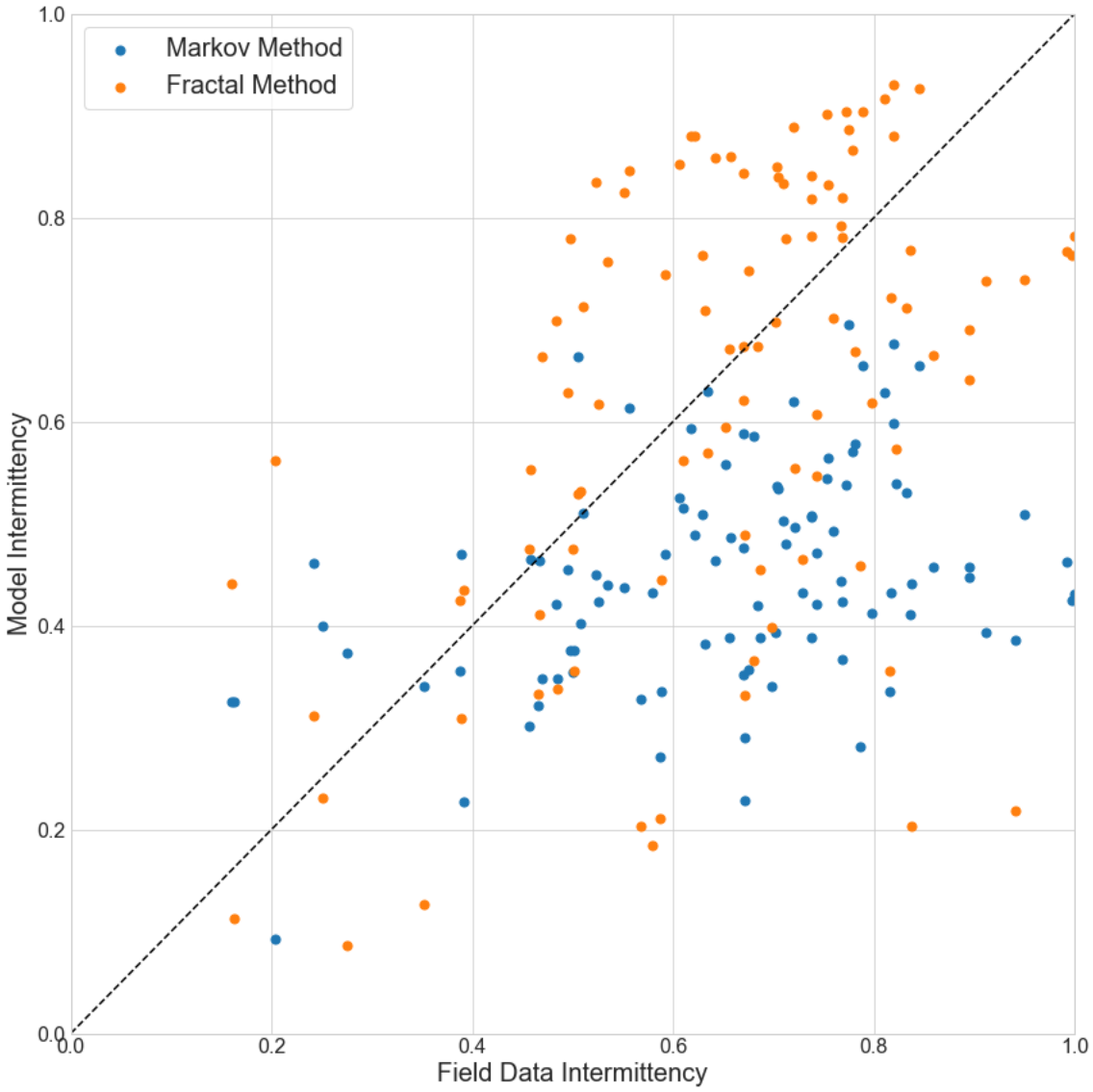


Figure 6. Intermittency Comparison for Ensemble 1

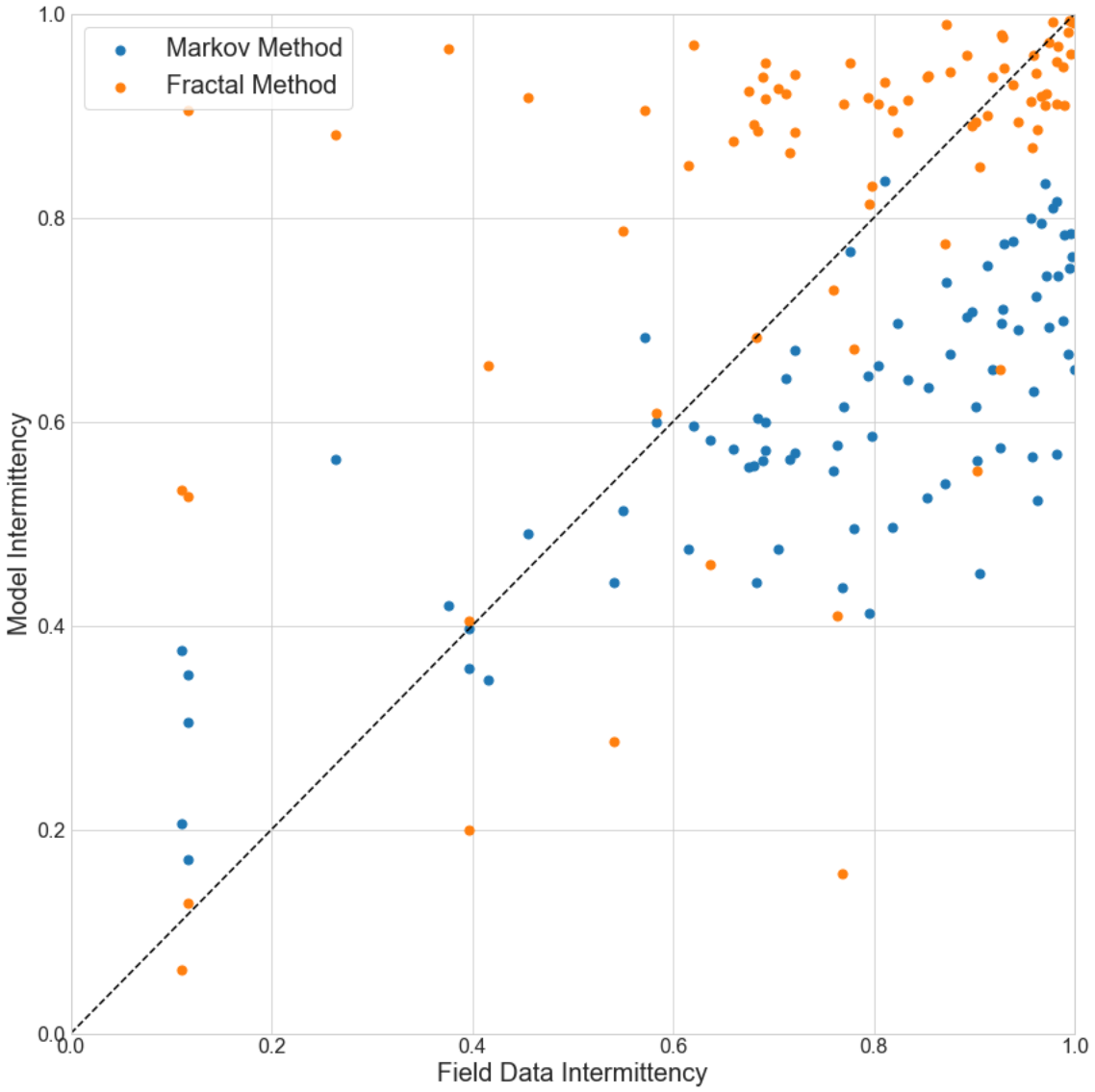


Figure 7. Intermittency Comparison for Ensemble 3

Table 3 provides a summary of the difference between the modeled and observed intermittency (i.e., the error) for selected percentiles. Negative error values indicate model under prediction of intermittency, while positive error values indicate over-prediction. Table 3 shows that, for all five ensembles, the Markov method under predicts intermittency. For many cases, even the 90th percentile values are slightly negative, indicating that the Markov method under predicts intermittency at more than 90 of the 100 sensors. The fractal method has less overall bias.

Table 3. Error Percentiles for Intermittency for the Markov and Fractal Methods

Ensemble	Simulation Method	Minimum	10th Percentile	50th Percentile	90th Percentile	Maximum
1	Markov	-0.573	-0.438	-0.190	-0.004	0.220
	Fractal	-0.723	-0.268	0.014	0.217	0.358
2	Markov	-0.616	-0.335	-0.147	0.069	0.403
	Fractal	-0.562	-0.097	0.124	0.287	0.585
3	Markov	-0.455	-0.332	-0.165	0.033	0.300
	Fractal	-0.612	-0.095	0.019	0.258	0.789
4	Markov	-0.589	-0.400	-0.217	-0.069	0.136
	Fractal	-0.492	-0.122	0.032	0.155	0.277
5	Markov	-0.995	-0.389	-0.172	-0.058	0.074
	Fractal	-0.995	-0.259	0.017	0.149	0.276

Note: Error = Predicted Value from Model – Observed Value from Field Trial; negative values indicate model under-prediction of intermittency; positive values indicate model over-prediction of intermittency.

Concentration upcrossing frequency and upcrossing duration were calculated at each sensor location across a range of concentration thresholds. Selected results are summarized in two ways: 1) upcrossing metrics averaged over sensor locations at each concentration threshold, and 2) fraction of sensors with upcrossing metrics within a factor of two of the field trial at each concentration threshold. Figure 8 and Figure 9 show the results for Ensembles 1 and 3, respectively.[85] As the concentration threshold increases, fewer sensors exceed the threshold to be included in the comparison. The small number of sensors at high concentration thresholds leads to the erratic behavior of this metric at those high thresholds.

The Markov and fractal methods show similar trends in their upcrossing metrics for Ensembles 1 and 3. Both tend to over predict upcrossing frequency for lower concentration thresholds and under predict it at higher concentration thresholds. Overall, the fractal method yields upcrossing frequencies somewhat closer to that of the field trial ensembles across much of the range of concentration thresholds, though neither method is exceptionally accurate.

Both models under predict the upcrossing duration at lower concentration thresholds. The average upcrossing duration of the fractal method is more accurate than the Markov method for concentration thresholds below approximately 10^{-5} kg/m^3, but over predicts the upcrossing duration at higher concentration thresholds. The upcrossing duration produced by the Markov method is relatively flat across the range of concentration thresholds, but approaches that of the field trial ensembles for concentration thresholds above approximately 10^{-5} kg/m^3.

[85] The literature did not provide a performance criterion applicable to upcrossing frequency or duration. A factor of two analysis allowed us to visually interpret error across different threshold concentrations.

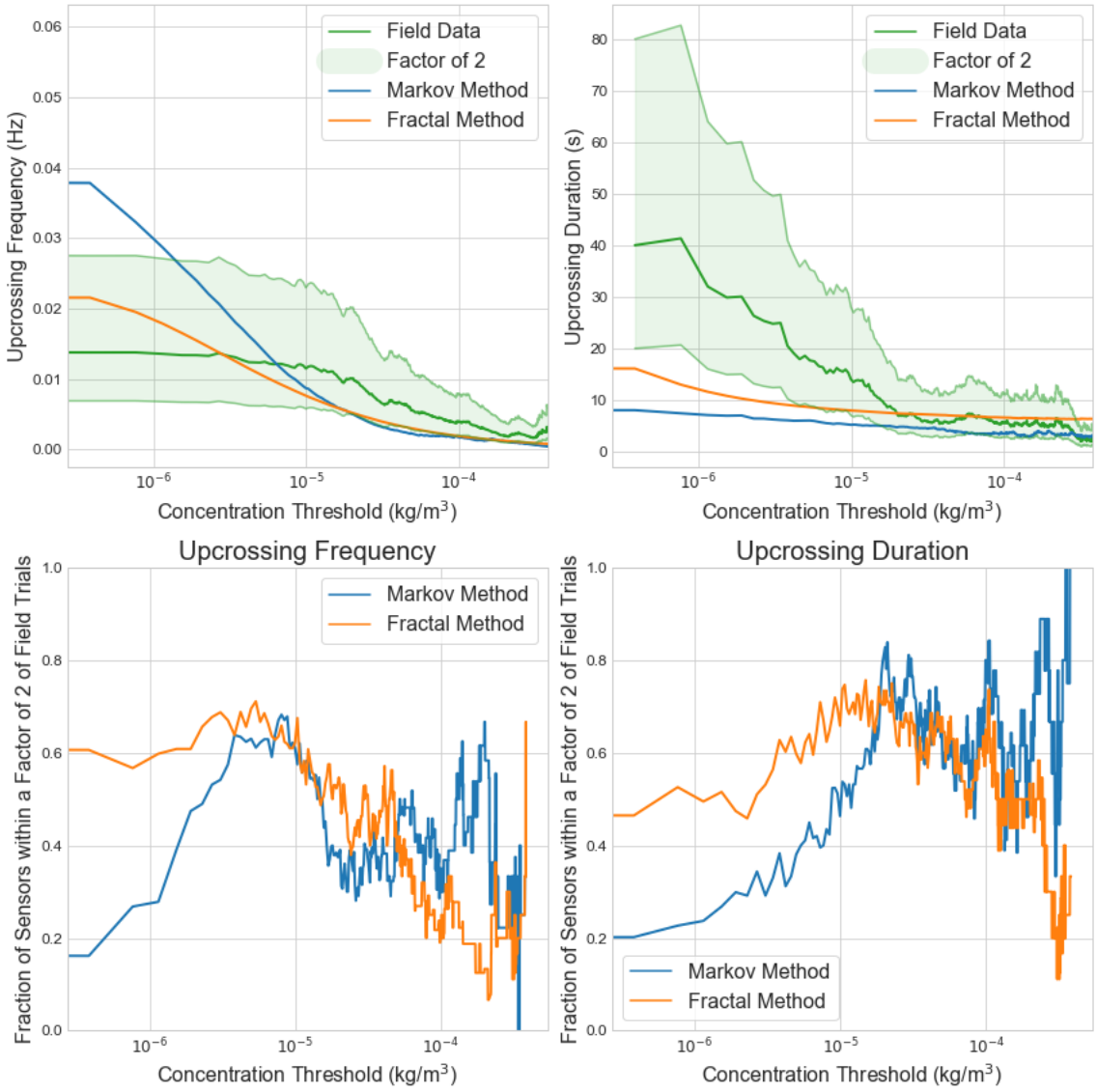


Figure 8. Upcrossing Frequency and Duration Summary Statistics for Ensemble 1

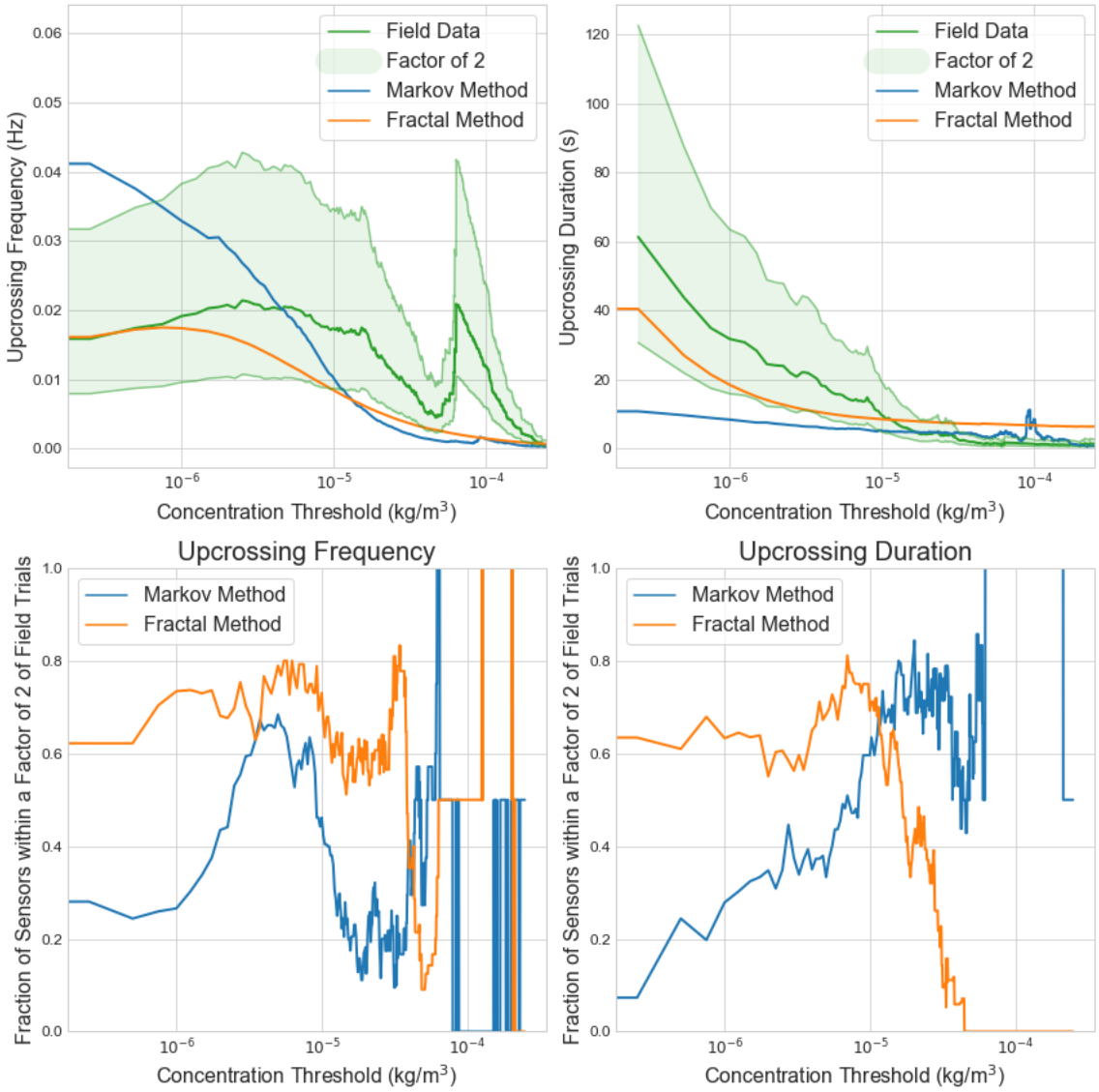


Figure 9. Upcrossing Frequency and Duration Summary Statistics for Ensemble 3

Overall, these results leave significant room for future improvement—being within a factor of two of the field data is only a modest marker of success at best, and it was not consistently met by either method. While both models are able to roughly approximate the general trends in the field trial metrics, a significant proportion of the predicted upcrossing frequencies and durations were more than a factor of two different from the observed values. Based on these three metrics, neither method adequately characterizes the concentration fluctuations seen in the field trial data. Some of these issues may be due to the limited amount of data supporting the ensembles themselves—even the largest ensemble had only 13 trials and the meteorology varied distinctly between trials, due to the choice to prioritize the number of trials per ensemble over the goodness of fit of the trial within the ensemble. The small numbers of releases available for each ensemble may have

artificially limited the concentration variance calculated at each sensor, especially where the ensemble mean was near zero.

B. Results of Applying the Models to HPAC Simulation Output

For trial 54 of the FFT 07 data, the Markov and fractal methods applied to HPAC output were compared with the field trial data, HPAC mean output alone, and a large-eddy CFD simulation (modeled with the VTHREAT software[86]) designed and run to match trial 54 as closely as possible. The CFD simulation was performed as part of an analysis by Platt et al., which includes further details on both the VTHREAT software and the parameters and resolution of the simulations in section 3.[87] The CFD results are compared with the results of the two methods examined in this paper to provide context regarding the quality of results that may be obtained through considerably more analytical and computational effort than that required to run HPAC. To allow direct comparison to trial 54, only a single realization of each of the Markov and fractal methods, chosen at random, was used in the comparison. Similarly, a single CFD realization was selected at random for use in the comparison. Figure 10 displays results for a 5-second detection duration, which is consistent with the sampling rate of typical chemical detectors, and Figure 11 uses a 60-second detection duration, which is intended to typify biological detectors.

[86] G. Bieberbach, P. E. Bieringer, A. Wyszogrodzki, J. Weil, R. Cabell, J. Hurst, and J. Hannan, "Virtual chemical and biological (CB) agent data set generation to support the evaluation of CB contamination avoidance systems," The Fifth Symposium on Computational Wind Engineering (CWE 2010), 2010.

[87] Christopher Czech, Nathan Platt, Jeffry Urban, Paul Bieringer, George Bieberbach, Andrzej Wyszogrodzki, and Jeffrey Weil, "A comparison of hazard area predictions based on the ensemble-mean plume versus individual plume realizations using different toxic load models, Paper 2.5 at the Special Symposium on Applications of Air Pollution Meteorology," *91st Annual American Meteorological Society Meeting,* Seattle, Washington. 2011.

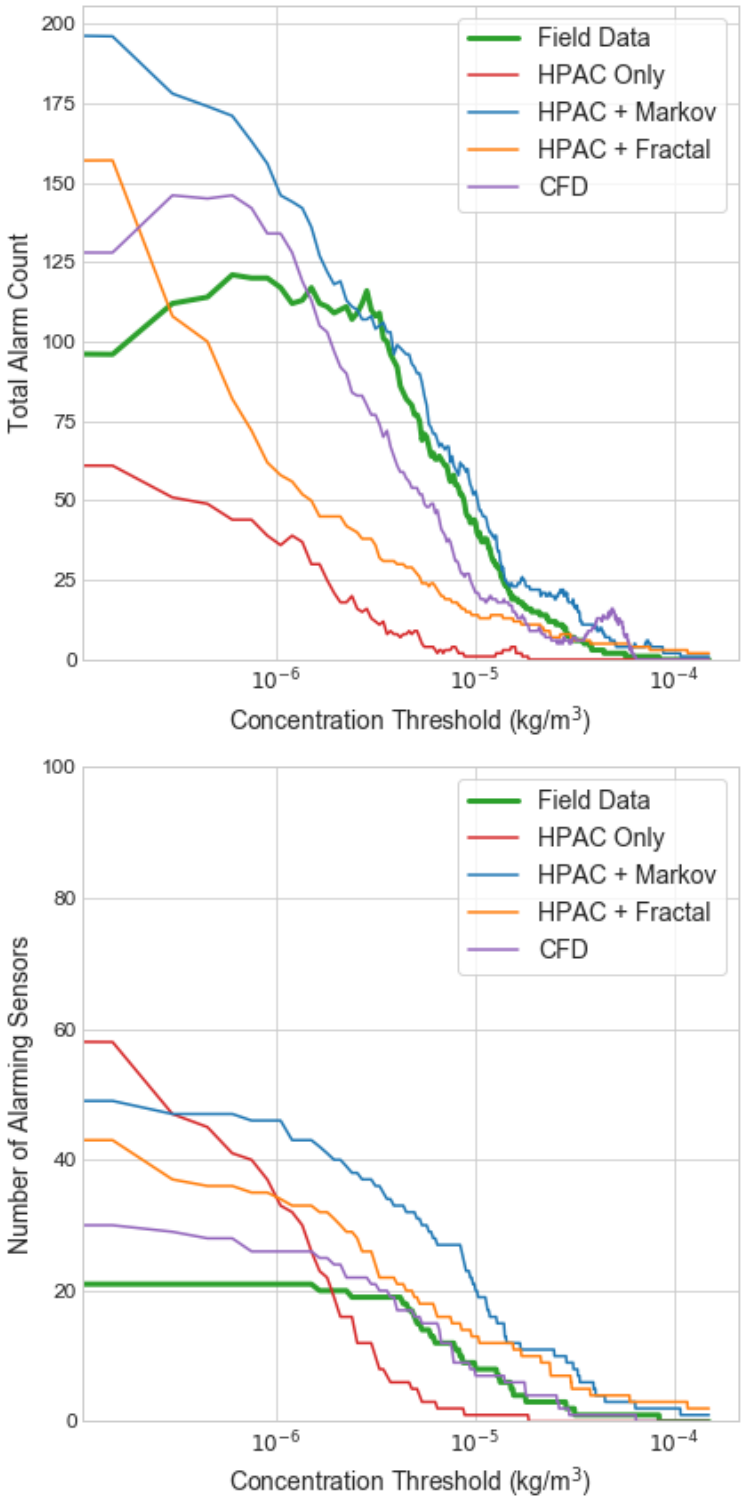


Figure 10. Detector Alarms for Various Models with 5-second Required Upcrossing Duration

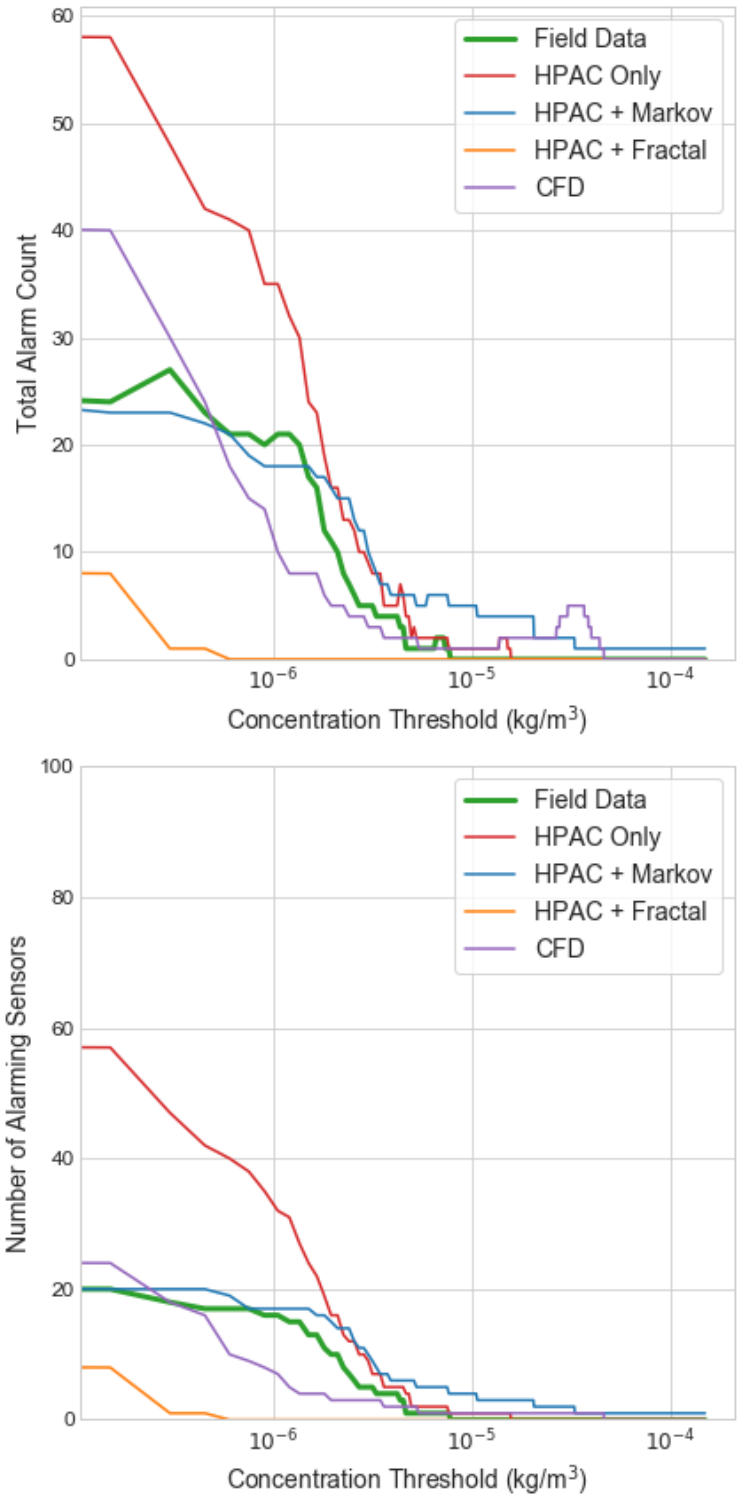


Figure 11. Detector Alarms for Various Models with 60-Second Required Upcrossing Duration

For a five-second detection duration, HPAC alone over predicts the number of alarming detectors for lower concentrations, but under predicts at higher concentrations, due to a lack of fluctuations above the ensemble mean. However, when HPAC is used in conjunction with either the Markov method or the fractal method, the number of alarming detectors show better correspondence with the field trials. Both models over predict the number of alarming detectors (lower plots, Figure 10 and Figure 11), though both get quite close to the field data at higher concentration thresholds, with the fractal method performing slightly better in this regime. The two models show opposite trends for total alarm count (upper plots, Figure 10 and Figure 11), with the Markov method over predicting and the fractal method under predicting compared to the field data. For the total alarm count, the Markov outperforms the fractal method for concentration thresholds greater than approximately 10^{-6} kg/m^3. Overall, the CFD simulation performs better than either the Markov or fractal methods for the five-second detection duration and provides remarkably accurate estimates of the number of alarming detectors for concentration thresholds above approximately 4×10^{-6} kg/m^3.

When a 60-second detection duration is required (Figure 11), both the total detector alarms and number of alarming detector metrics obtained from the use of HPAC alone are substantially greater than the field trial data across almost the entire range of concentration thresholds, approaching the field trial data results for the largest concentration thresholds. The Markov method matches the field data remarkably well for a 60-second detection duration. For some concentration thresholds, it even matches the field trial metrics better than the CFD simulation. The fractal method significantly under predicts by both metrics. This under prediction is likely because the initial temporal grid spacing of 23.71 seconds, selected to match the temporal correlation scale, was much smaller than the detection duration. The initial temporal grid defines the maximum time/distance between sensor readings in which the fluctuations are correlated in any way. For the concentration to exceed the threshold for the entire 60-second detection duration, none of the random amplitudes drawn for multiple consecutive pulses can be substantially negative. A longer initial temporal grid spacing might improve these metrics. When the initial temporal grid resolution is finer than the detection duration, multiple pulses necessarily span the detection duration. This requires that the multiple random draws that define the pulse amplitude throughout the detector duration be above the concentration threshold for a modeled detector to alarm. A longer initial temporal grid resolution requires fewer consecutive pulses to have randomly drawn amplitudes above the threshold.

These results, though more promising than those seen above for the FFT 07 ensembles, still leave something to be desired. The CFD marker shows what can currently be done with increased computational complexity and detailed input data. The CFD results mirror the field trial data reasonably well. The Markov and fractal methods do seem to be an improvement over HPAC alone, but only match the field trial data sporadically across

both metrics. There is no clear pattern of over or underestimation, and no clear rank ordering of results, indicating that the close matching seen in the Markov method may not be robust. Testing of the methods over more field trials is needed to determine if there is any consistency to these results.

This page is intentionally blank.

Appendix A.
Ensembles from Fusion Field Trial 2007 Data

This appendix provides, in Table A–1, information on each trial of Fusion Field Trial 2007 (FFT 07) that was used to create each ensemble of similar trials.

Table A–1. Characteristics of FFT 07 Trials by Ensemble

Ensemble	Trial	Number of Release Locations	Release Flow Rate (liters per minute)	Wind Speed (meters per second)	Wind Direction (degrees from North)
1	22	1	175	1.39	151.75
	63	1	150	1.69	166.91
	46	1	175	1.71	104.96
	20	1	200	1.85	144.23
	39	1	150	2.29	132.28
	7	1	200	2.39	156.60
	42	1	200	2.39	140.15
	53	1	125	2.41	136.70
	34	1	200	2.65	146.60
	30	1	125	2.72	144.16
	11	1	200	2.74	156.04
	29	1	175	2.85	141.21
	54	1	150	2.88	144.52
			Averages:	**2.30**	**143.55**
2	18	2	150	1.13	124.80
	19	2	175	1.19	119.60
	62	2	150	1.71	146.39
	40	2	150	2.18	145.64
	27	2	150	2.33	146.66
			Averages:	**1.71**	**136.62**
3	52	2	175	2.70	139.28
	32	2	175	2.78	127.97
	8	2	200	2.79	155.38
	12	2	200	2.85	146.25
	48	2	175	3.01	141.61
	24	2	175	3.24	157.46
			Averages:	**2.90**	**144.66**
4	47	3	175	1.60	109.70
	41	3	150	1.61	121.84
	26	3	150	2.16	147.53
	10	3	200	2.41	172.28
	49	3	200	2.74	136.21
	51	3	200	2.74	153.38
	28	3	175	2.79	141.42
	6	3	200	2.82	154.07
			Averages:	**2.36**	**142.03**
5	31	4	150	1.52	121.97
	43	4	200	2.01	133.22
	50	4	200	2.84	143.88
			Averages:	**2.12**	**133.02**

Appendix B.
Derivation of Spatial and Temporal Correlation Scales

The spatial/temporal correlations in the fluctuating concentration field generated by the fractal method are determined by the grid spacing. Locations more distant than the coarsest grid spacing will exhibit no correlation in the concentration fluctuations, though the concentration may still be correlated due to correlations in the mean field. To parameterize the fractal method, the pairwise spatial correlation of the concentration fluctuations was calculated for every combination of sampler locations in the ensemble. An exponential model was then fit to the spatial correlations as a function of distance. An exponential model was selected because it was a simple functional form that yielded unity at zero distance and converges to zero correlation at large distances, as would be expected for a spatial correlation function. Following conventions in the literature, the spatial correlation scales were taken as the e- folding distance,[88] i.e., the distance at which the exponential fit of the correlation coefficient falls below $\frac{1}{e} \approx 0.37$. Although the spatial correlation scales obtained in this manner are less than the minimum distance between sensors (50 meters), there is still an appreciable positive correlation between sensors up to approximately 100 meters apart that allowed the exponential model to be fit to the data. A denser grid of sensors would have allowed for a better characterization of the spatial correlation at closer distances.

Similar to the calculation of the spatial correlation scale, the temporal scale, T_x, was directly computed from the autocorrelation coefficient, $R(\tau)$, as a function of the time lag, τ:[89]

$$T_x = \int_0^{\tau_e} R(\tau) d\tau \qquad \text{(B-1)}$$

where τ_e is e- folding time, i.e., the time lag at which the autocorrelation function falls below $\frac{1}{e} \approx 0.37$.

---

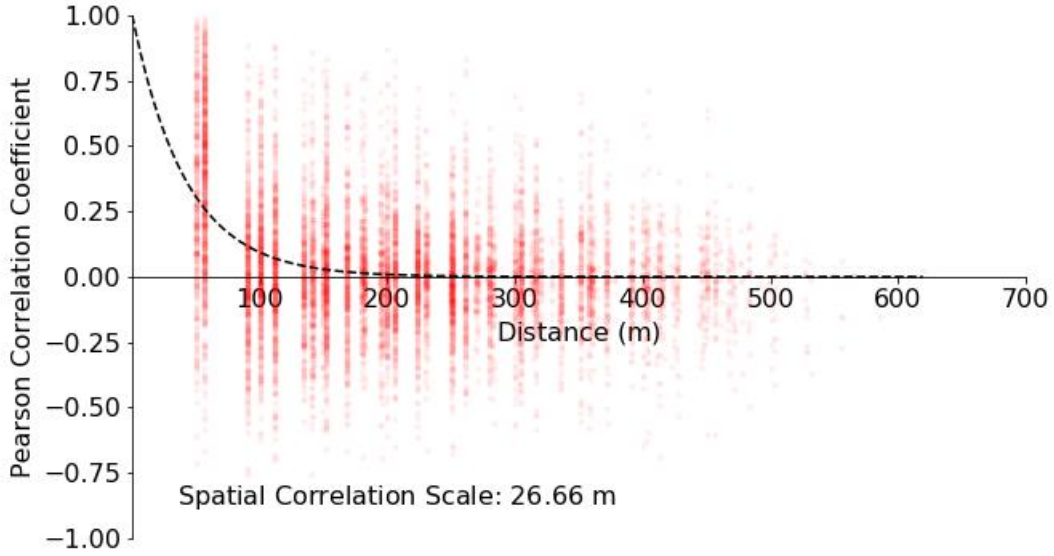
88 Eugene Yee, R. Chan, P.R. Kosteniuk, G.M. Chandler, C.A. Biltoft, and J.F. Bowers, "Experimental Measurements of Concentration Fluctuations and Scales in a Dispersing Plume in Atmospheric Surface Layer Obtained Using a Very Fast Response Concentration Detector," *Journal of Applied Meteorology and Climatology* 33, no. 8 (August 1994), 1012.

89 Ibid.

Stemming from a desire to maintain a consistent methodology for parameterizing each method, the e- folding scheme was used to define the spatial and temporal scales as used by both the Markov and fractal methods. Using the e- folding scheme to define the coarsest grid resolution in the fractal method results in no correlation in fluctuations beyond the e- folding time/distance. In reality, positive autocorrelation occurs over significantly longer temporal and spatial scales, up to approximately 100 seconds and 100 meters in the field trial data. Therefore, this e- folding scheme artificially reduced the correlation distance of simulated concentration fluctuations generated by the fractal method.

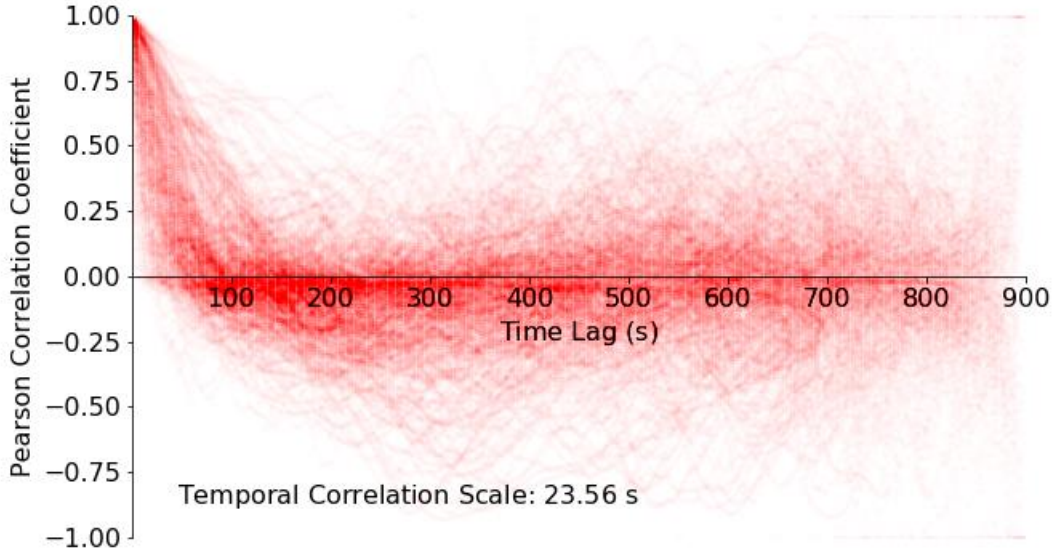
Figures B-1 through B-10 depict the determination of the spatial correlation scales and temporal correlation scales for each of the ensembles generated from Fusion Field Trial 07 (FFT 07).

A. FFT 07 Ensemble #1



Note: m = meters.

Figure B-1. Spatial Correlations for FFT 07 Ensemble #1



Note: s = seconds.

Figure B-2. Temporal Correlations for FFT 07 Ensemble #1

B. FFT 07 Ensemble #2

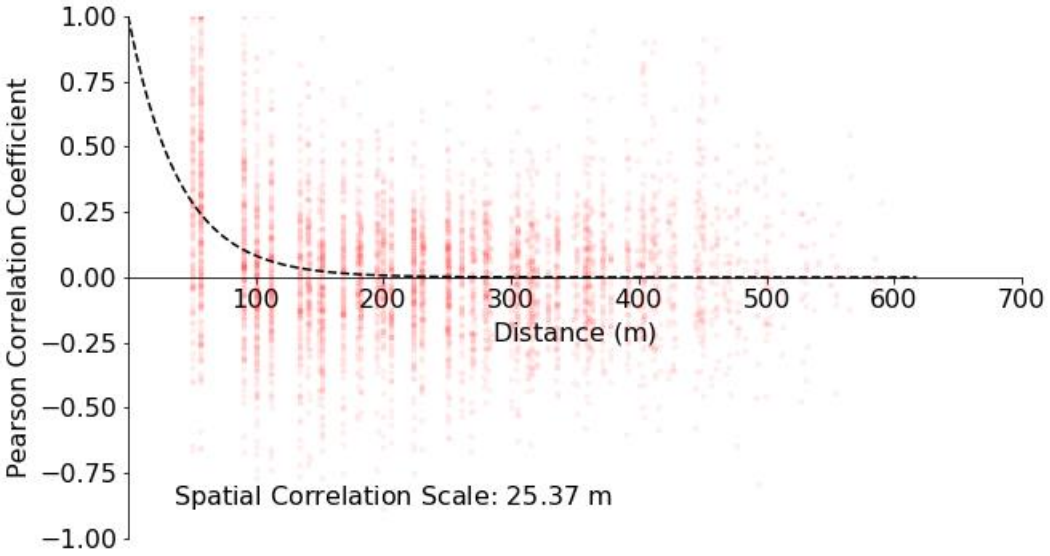


Figure B-3. Spatial Correlations for FFT 07 Ensemble #2

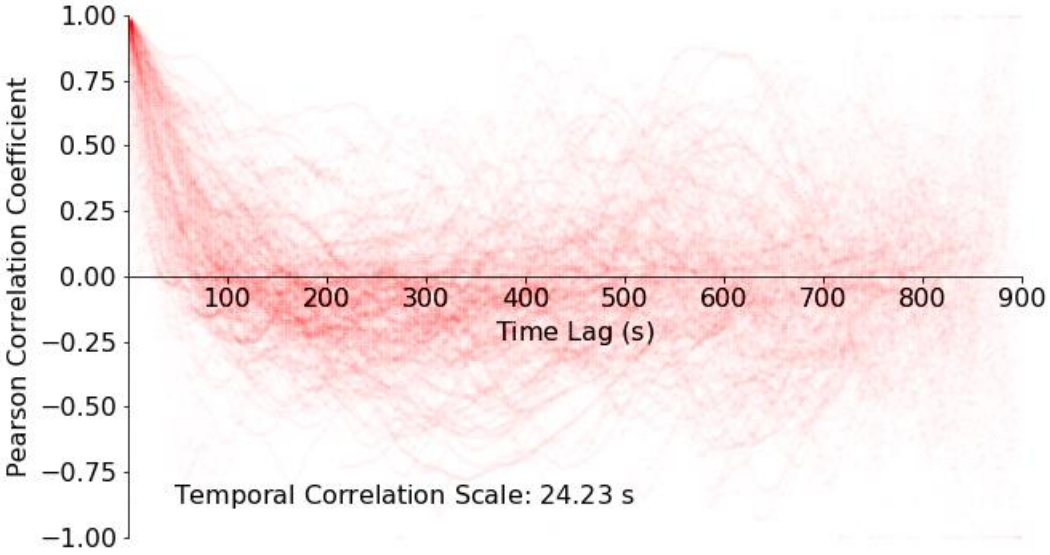


Figure B-4. Temporal Correlations for FFT 07 Ensemble #2

C. FFT 07 Ensemble #3

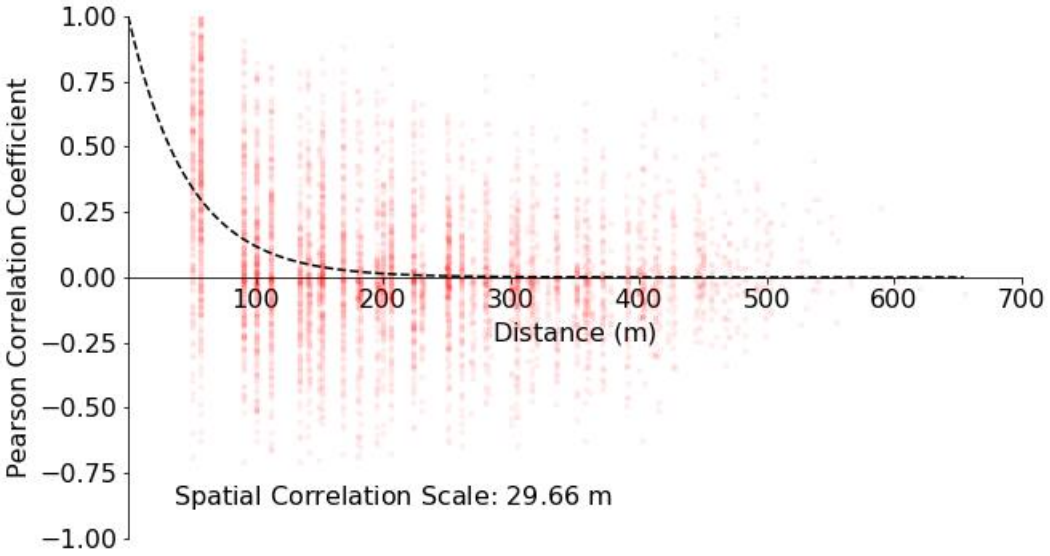


Figure B-5. Spatial Correlations for FFT 07 Ensemble #3

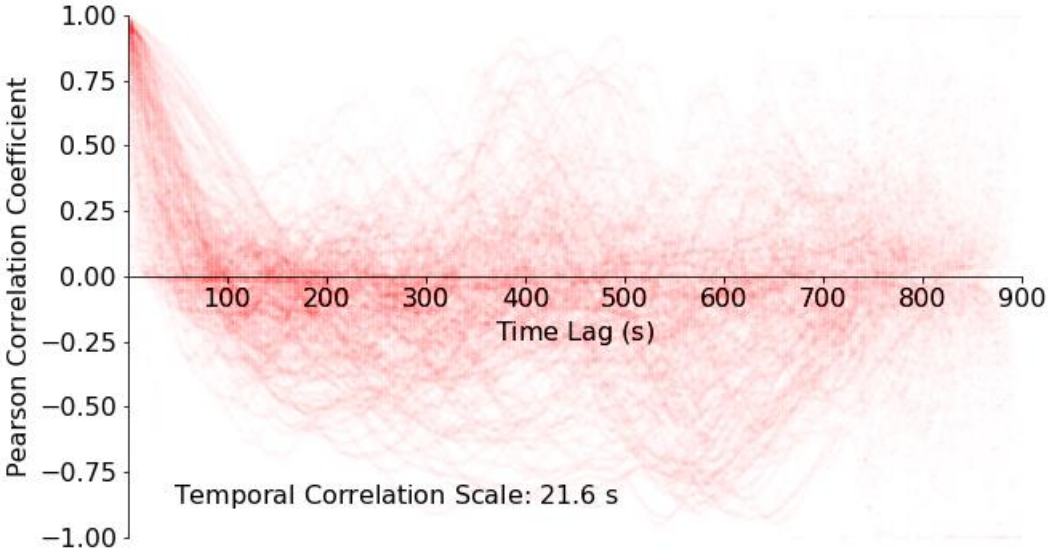


Figure B-6. Temporal Correlations for FFT 07 Ensemble #3

D. FFT 07 Ensemble #4

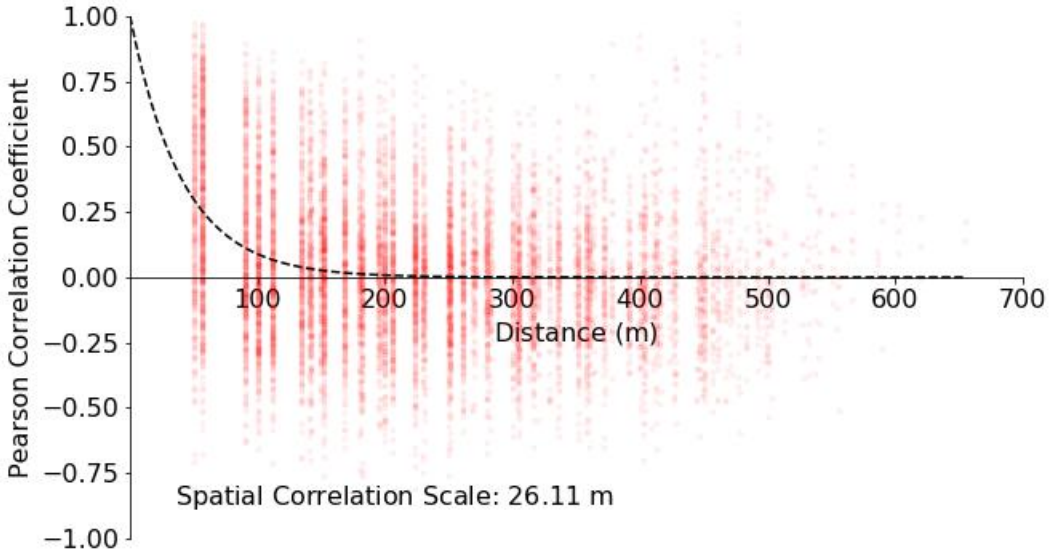


Figure B-7. Spatial Correlations for FFT 07 Ensemble #4

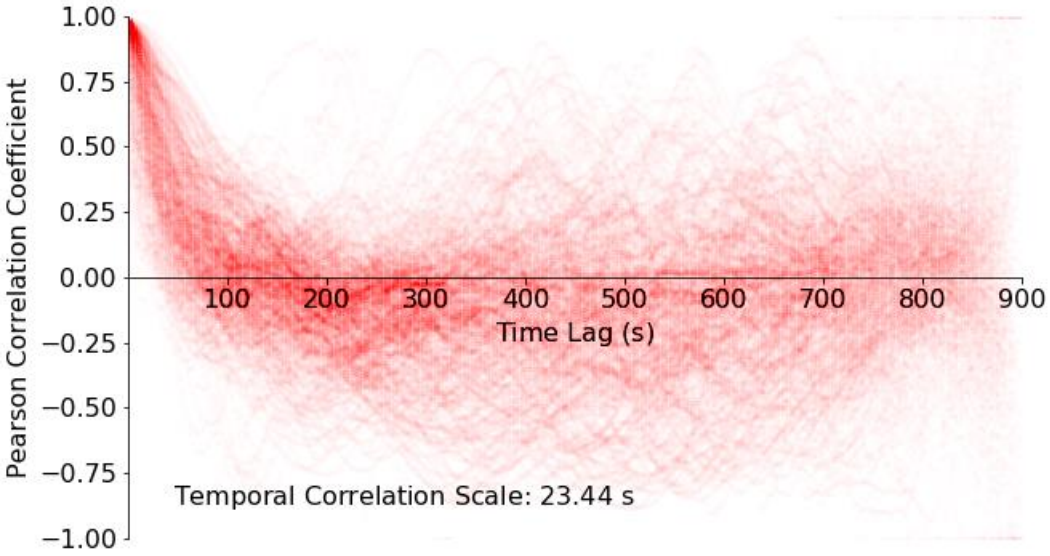


Figure B-8. Temporal Correlations for FFT 07 Ensemble #4

E. FFT 07 Ensemble #5

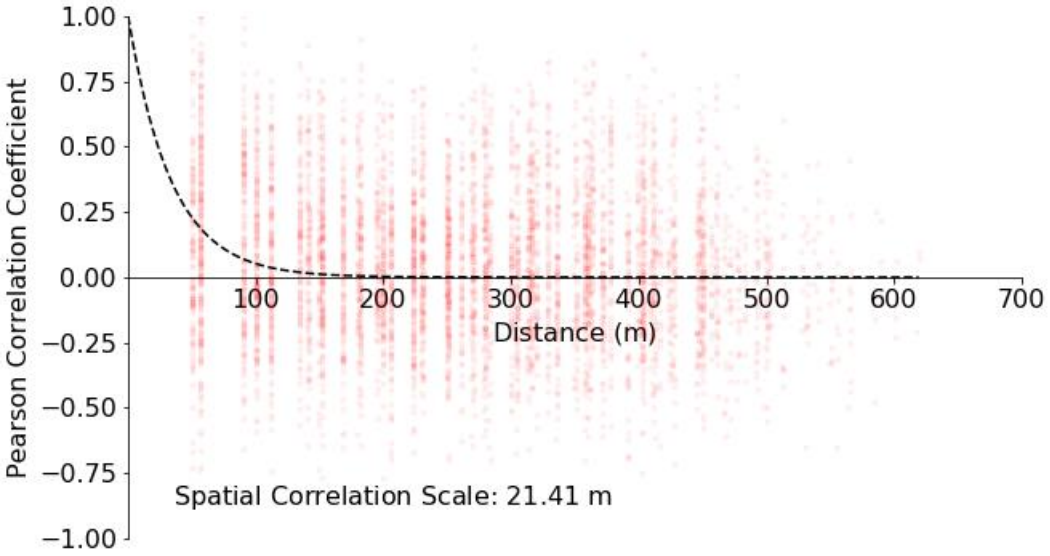


Figure B-9. Spatial Correlations for FFT 07 Ensemble #5

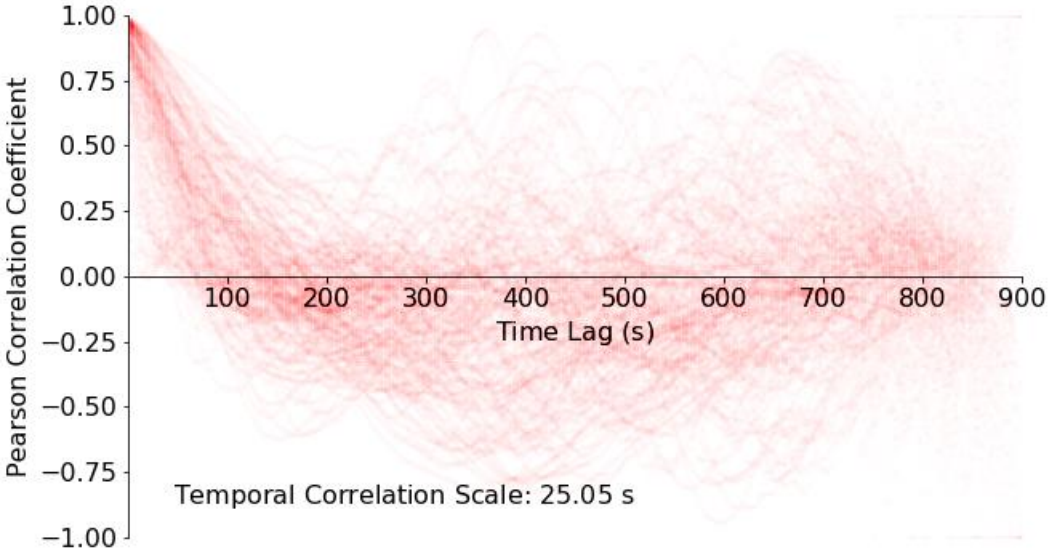


Figure B-10. Temporal Correlations for FFT 07 Ensemble #5

This page is intentionally blank.

Appendix C.
Results for Other Ensembles

This appendix includes the intermittency and upcrossing statistics for Fusion Field Trial 07 (FFT 07) Ensembles 2, 4, and 5. They are presented here for completeness, and similar trends to those discussed in section 8.A are visible throughout. The results for Ensemble 2 were not included in the main body of the paper because the results for Ensemble 2 and Ensemble 3 are somewhat redundant, and Ensemble 3 was the more coherent and consistent of the two ensembles. Results for Ensembles 4 and 5 are relegated to this appendix because the inclusion of more release locations in each trial of Ensembles 4 and 5 caused a more uniform and widespread plume, which in turn decreased the range of intermittency and concentration fluctuations. Furthermore, Ensembles 4 and 5 included fewer trials, because they were limited to only those trials that used three and four release locations, respectively.

A. Metrics for FFT 07 Ensemble 2

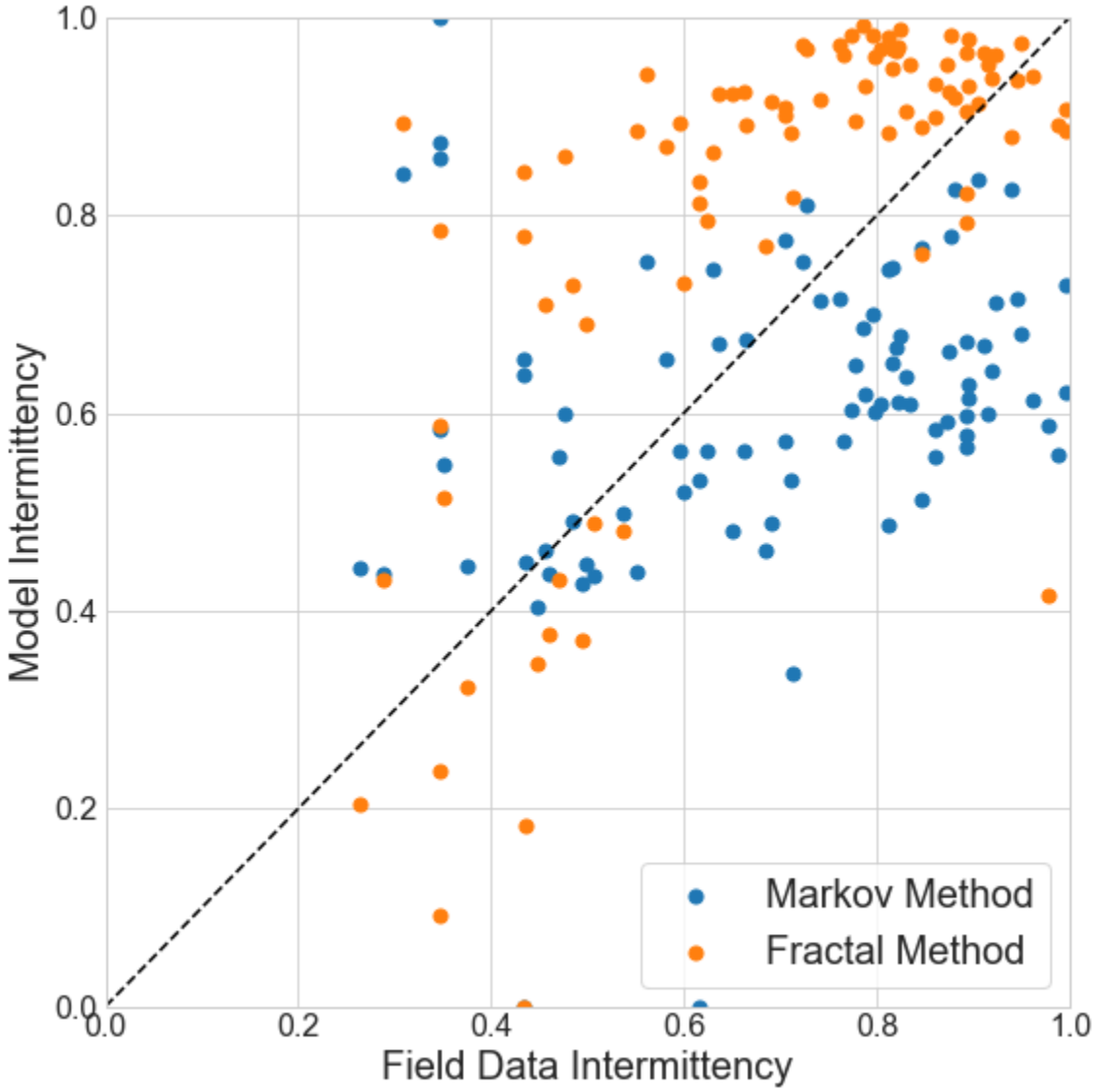


Figure C-1. Intermittency Comparison for Ensemble 2

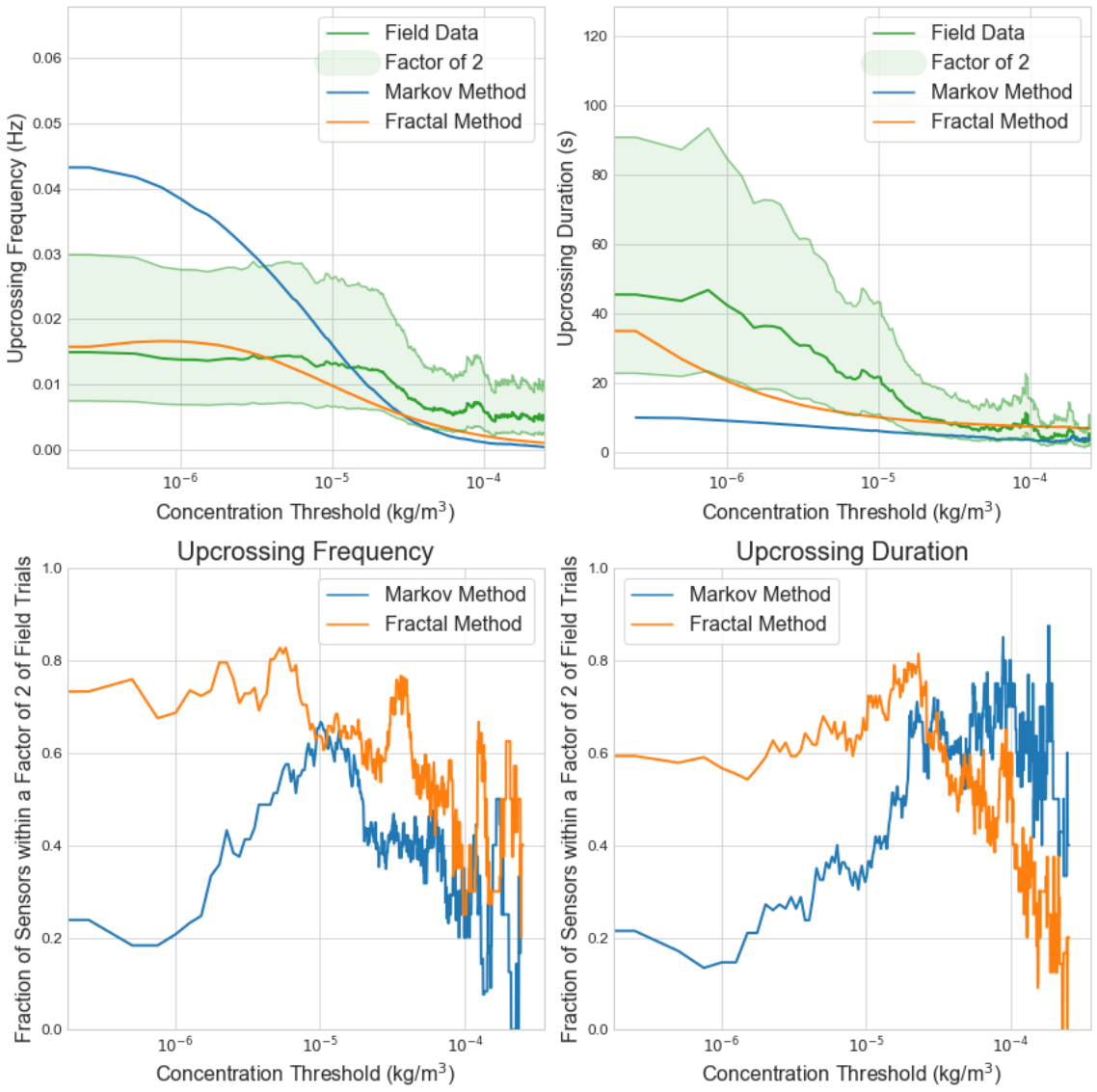


Figure C-2. Upcrossing Frequency and Duration Summary Statistics for Ensemble 2

B. FFT 07 Ensemble 4

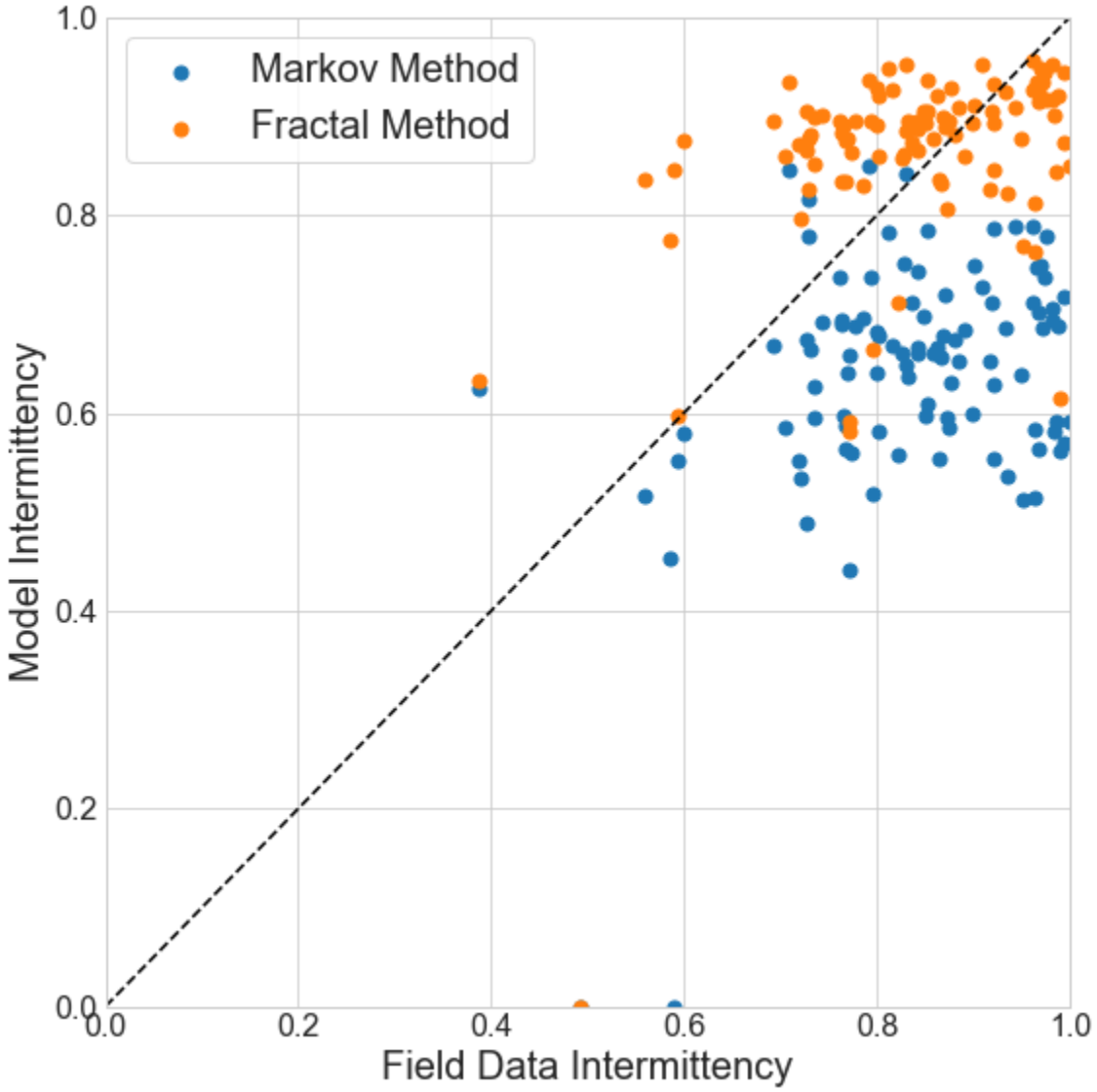


Figure C-3. Intermittency Comparison for Ensemble 4

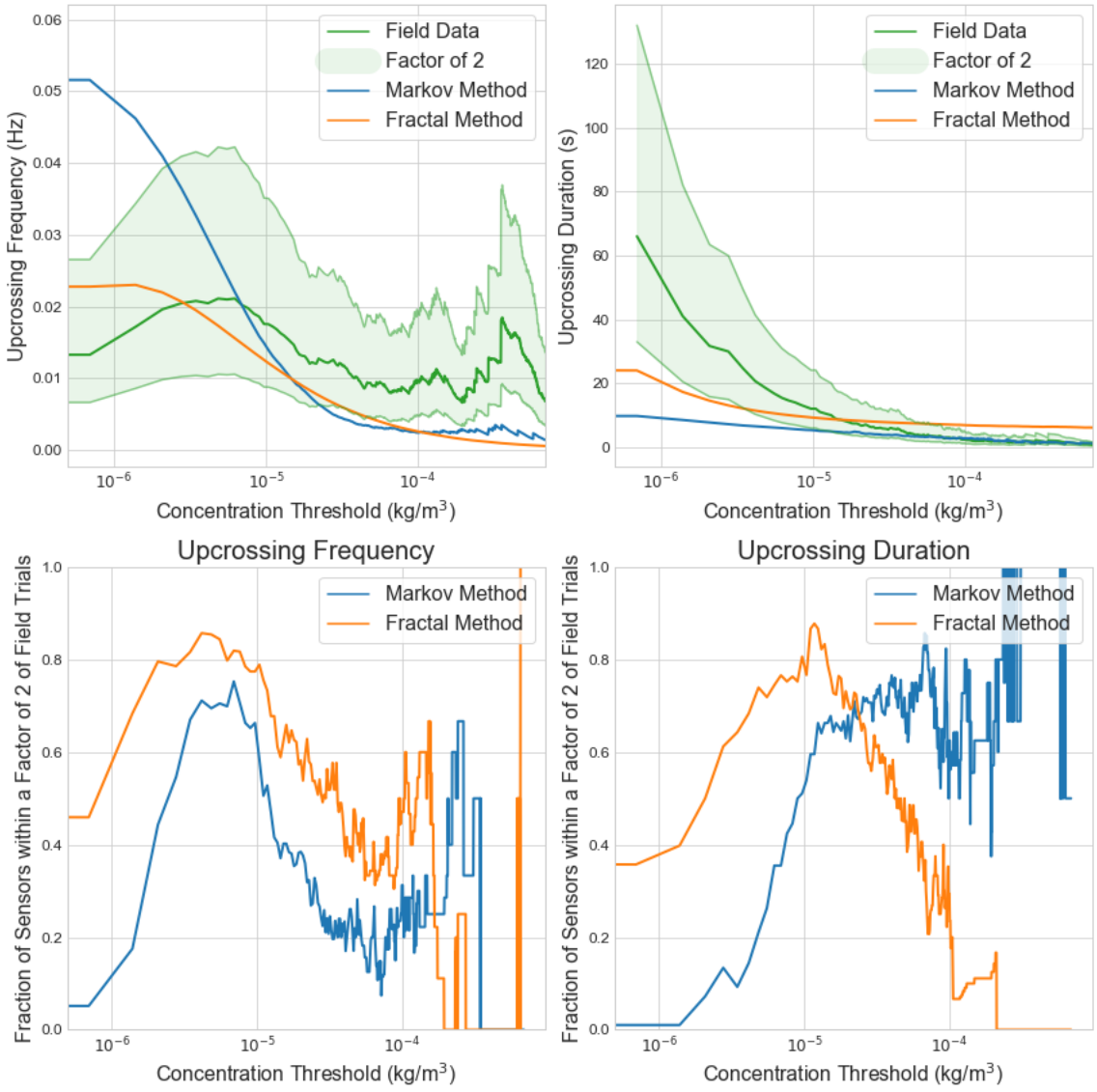


Figure C-4. Upcrossing Frequency and Duration Summary Statistics for Ensemble 4

C. FFT 07 Ensemble 5

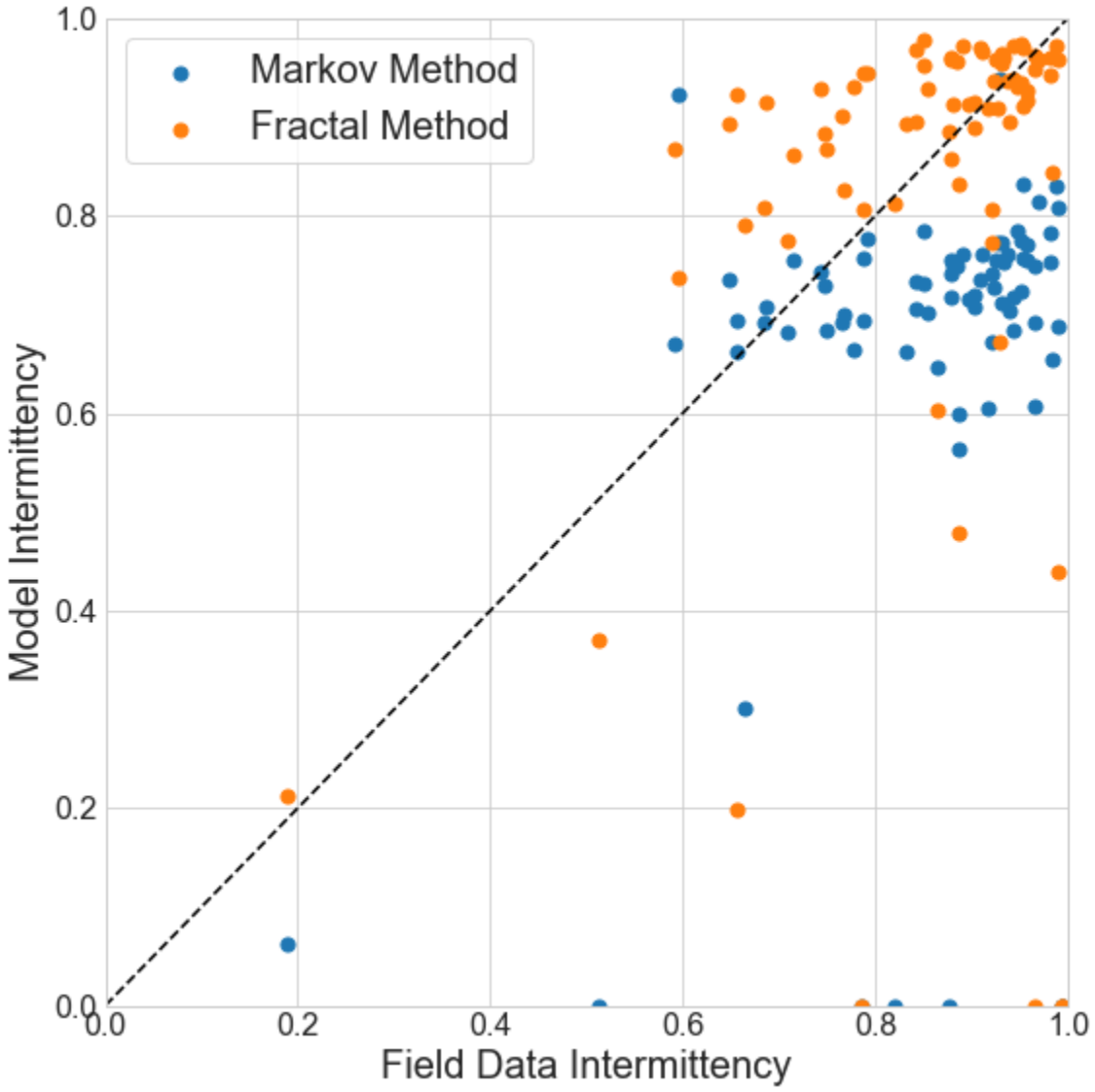


Figure C-5. Intermittency Comparison for Ensemble 5

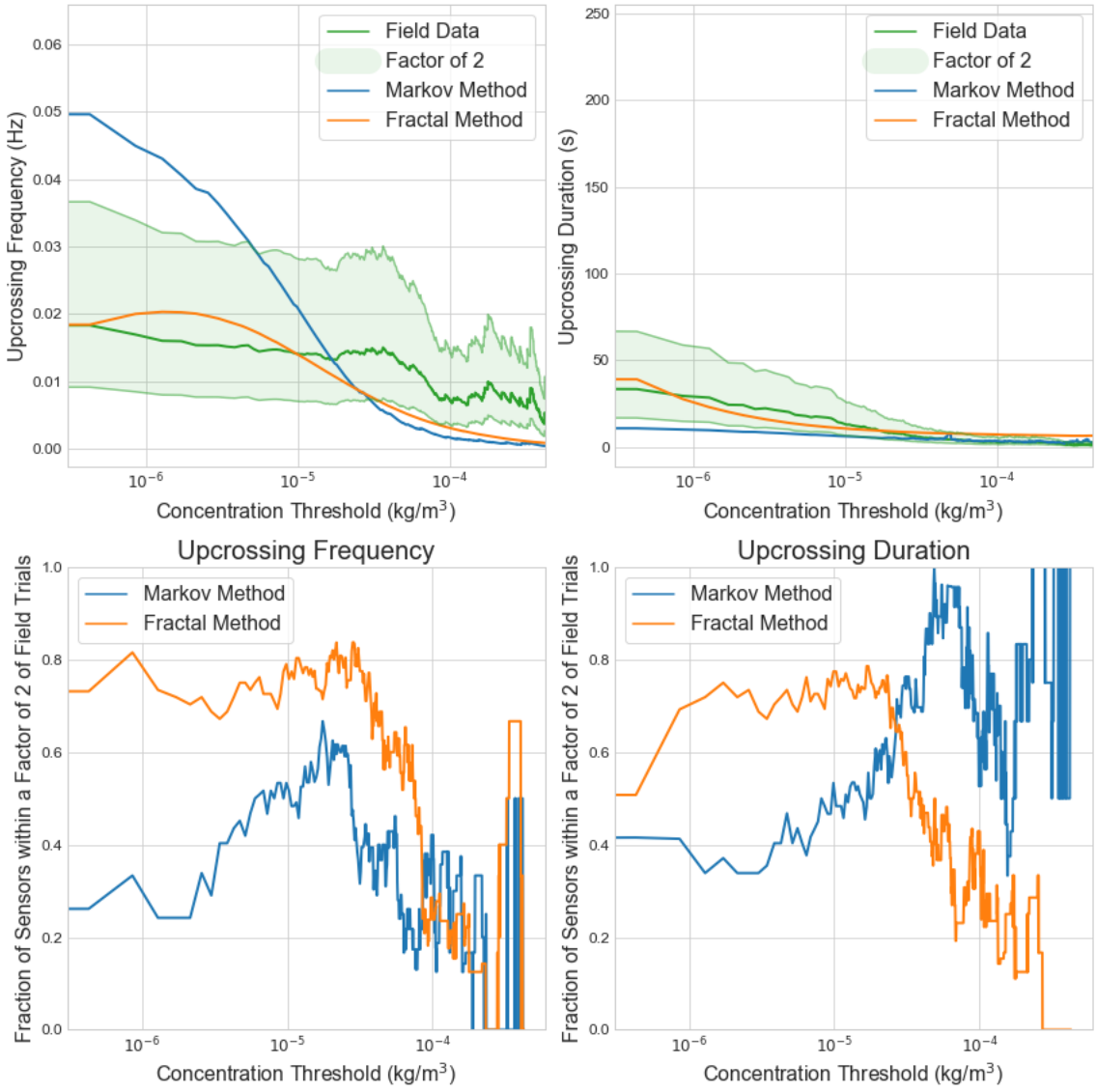


Figure C-6. Upcrossing Frequency and Duration Summary Statistics for Ensemble 5

This page is intentionally blank.

Appendix D.
Markov Method Implementation

The Markov method was implemented in Python 3.6.0 and has dependencies on numpy 1.11.3, scipy 1.1.0, and matplotlib 2.0.0 libraries. The python code used in this implementation is provided in this appendix.

1. markov_method.py

```
import numpy as np
import scipy as sp
from Transcendental_Solver import find_parameters
from lognorm_solver import lognorm_parameters
import warnings


def main():
    #example of usage at a single geographic point

    #numpy and pandas get ornery about taking the log of 0,
    #but the output is accounted for appropriately
    with warnings.catch_warnings():
        warnings.filterwarnings("ignore", category=RuntimeWarning)

        #read in ensemble mean and variance
        mean, var = read_ensemble(path)

        #markov method for input with 30 second timesteps, output with 1 second timesteps
        realizations = combo_method_transient(mean, var, timestep=30, dt=1)

        #realizations will be an array with dimensions realizations x time
        save_and_plot_concentrations(realizations)


def read_ensemble(path):
    pass

def save_and_plot_concentrations(realizations):
    pass


def combo_method_transient(mean, var, timestep=1., tcorr=22., runs=1000, dt=None,
↪   dtsteps=10, pdf='lognormal'):
    '''
    <mean> (array-like): concentration means (time series)
    <var> (array-like): concentration variances (time series)
    <timestep> (float -or- array-like): length of a timestep -or- length of each timestep
    <tcorr> (float -or- array-like): time correlation scale -or- time correlation scale
↪   at each timestep
    <runs> (int): number of individual realizations to produce
    <dt> (float): timestep to use for steady-state markov method
    <dtsteps> (int): number of steps to break each timestep into for steady-state markov
↪   method
    NOTE: <dt> and <dtsteps> cannot be used at the same time. If specified, dt takes
↪   precendence over <dtsteps>
    <pdf> ("lognormal" or "gamma"): specifies which concentration fluctuation probability
↪   density function
                                    to use for steady-state markov method
    '''
    #time and tcorr lists, should be same length as mean and var
    try: len(timestep)
    except TypeError: timestep = [timestep] * mean.shape[0]
    try: len(tcorr)
```

```
    except TypeError: tcorr = [tcorr] * mean.shape[0]
    if dt is not None:
        assert all((round(t%dt,5)==0.) or (round(t%dt,5)==round(dt,5)) for t in
        ↪  timestep), \
               'dt must divide evenly into time.'
        #dt = [dt] * mean.shape[0]
        #dtsteps=[int(round(t/dt,0)) for t, dt in zip(time, dt)]
    #else:
        #try: len(dtsteps)
        #except TypeError: dtsteps = [dtsteps] * mean.shape[0]
        #dt = [t/dtsteps for t, dtsteps in zip(time, dtsteps)]

    assert (len(mean)==len(var)) and (len(mean)==len(timestep)) and
    ↪  (len(mean)==len(tcorr)), \
        'The mean, variance, time, and correlation time need to be the same length.'


    #Creates a steady-state time series for each time step has a length of
    #dtstep * number of runs (therefore dtsteps timesteps for each single instantiation).
    #After the first timestep, each realization is seeded with the last value of the
    #previous timestep in order to maintain appropriate temporal correlations
    C = []
    for tstep in range(mean.shape[0]):
        if tstep==0: seed=None
        else: seed = raw_seed

        if dt is not None:
            Ci, raw_seed = combo_method_steady(mean[tstep], var[tstep], timestep[tstep],
            ↪  tcorr=tcorr[tstep], dt=dt, runs=runs, seed=seed, pdf=pdf,
            ↪  return_seed=True)
        else:
            Ci, raw_seed = combo_method_steady(mean[tstep], var[tstep], timestep[tstep],
            ↪  tcorr=tcorr[tstep], dtsteps=dtsteps, runs=runs, seed=seed, pdf=pdf,
            ↪  return_seed=True)
        C.append(Ci)

    C = np.concatenate(C, axis=1)

    return C


def combo_method_steady(mean, var, time, tcorr=22., runs=1000, seed=None, dt=None,
↪  dtsteps=100, pdf='lognormal', return_seed=False):
    '''
    <mean> (float): concentration means (time series)
    <var> (float): concentration variances (time series)
    <time> (float): length of the final timeseries
    <tcorr> (float): time correlation scale
    <runs> (int): number of individual realizations to produce
    <seed> (float -or- array-like): specifies the starting concentration for the runs
    <dt> (float): timestep to use
    <dtsteps> (int): number of steps to break time into
    NOTE: <dt> and <dtsteps> cannot be used at the same time. If specified, dt takes
↪  precendence over <dtsteps>
    <pdf> ("lognormal" or "gamma"): specifies which concentration fluctuation probability
↪  density function to use
```

```
    return_seed (boolean): whether or not to also return the last timestep of
↪   concentration unshifted and unclipped (intended for interoperability with
↪   combo_method_transient)
    '''
    # Timestep length (seconds) and count
    #NOTE: dt takes precedence over dtsteps if both are specified
    if dt is None:
        dt = time/float(dtsteps)
    tsteps = int(round(time / dt))

    if seed is not None:
        try:
            assert len(seed)==runs, "The seed provided does not correlate to the number
            ↪  of realizations requested."
        except TypeError:
            seed = np.full(runs, seed)
    else:
        seed = np.full(runs, mean)

    #Concentration mean and variance
    c = mean
    v = var
    T = float(tcorr) # Correlation time

    if pdf=='gamma':
        # Normalized mean square concentration
        nmsc = v/(c*c) + 1
        #k and s dependence on nmsc roughly linear for nmsc>2, asymptotic as nmsc
        ↪  approaches 1
        #so if you're getting non-finite values, you'll need to increase nmsc_bound
        nmsc_lower_bound = 1.2
        # nmsc_upper_bound = 20.
        if nmsc < nmsc_lower_bound: nmsc = nmsc_lower_bound
        # elif nmsc > nmsc_upper_bound: nmsc = nmsc_upper_bound

        # Solve system of transcendental equations for clipped-gamma parameters
        k, s, l, gamma = find_parameters(nmsc)

    elif (pdf=='lognorm') or (pdf=='lognormal'):
        #solve system of transcendental equations for lognormal parameters
        c50, sig, cbase, gamma = lognorm_parameters(c, v)
        #adjust mean upward to account for intermittancy
        c /= gamma

    else:
        raise ValueError("pdf must be 'gamma' or 'lognormal'")

    #Concentration time series instantiation
    C = np.empty([len(seed), tsteps+1])
    #first value is either seeded based on last timestep, or set to mean
    #this value will be removed after markov process is completed
    C[:,0] = seed

    for i in range(1, tsteps+1):
        a = -(C[:,i-1]-c)/T
```

```
        if pdf=='gamma':
            pc = k * np.power(np.abs(C[:,i-1]/c)*k,k-1) * np.exp(-np.abs(C[:,i-1]/c)*k) /
            ↪  (c*sp.special.gamma(k))
            #parameter to shift concentration semi-randomly (based on concentration pdf)
            b21 = 2./(T*pc*sp.special.gamma(k))
            b2  = b21*c*((1./k) * sp.special.gammaincc(k+1,np.abs(C[:,i-1]/c)*k) *
            ↪  sp.special.gamma(k+1) - sp.special.gammaincc(k,np.abs(C[:,i-1]/c)*k) *
            ↪  sp.special.gamma(k))
            b = np.sqrt(b2)
            #if b goes infinite, use L'Hopitals rule
            if not np.isfinite(b).any():
                b = np.sqrt(2*(c - np.abs(C[:,i-1])) / (T*((k-1)*(s/np.abs(C[:,i-1])) -
                ↪  (1/s))))
                #if b still goes infinite, use inst_conc >> mean_conc approximation
                if not np.isfinite(b).any():
                    b = np.sqrt(2*np.abs(C[:,i-1])*c / (T*k))

        elif (pdf=='lognorm') or (pdf=='lognormal'):
            term1 = np.log(np.abs(C[:,i-1])/c50) / (np.sqrt(2)*sig)
            b2 = (c*np.sqrt(2*np.pi)*sig*np.abs(C[:,i-1])/T)*np.exp(term1**2) *
            ↪  (sp.special.erf(term1)-sp.special.erf(term1-(sig**2/(np.sqrt(2)*sig))))
            b = np.sqrt(b2)
        else:
            raise ValueError("pdf must be 'gamma' or 'lognormal'")

        r = np.random.normal(size=len(seed))

        C[:,i] = C[:,i-1] + a*dt + b*np.sqrt(dt)*r
        if np.isnan(C[:,i]).any():
            raise RuntimeError('Time series has become non-finite (either NaN or Inf).')

    #remove initial seed
    C = C[:,1:]

    if return_seed: seed = C[:,-1].copy()

    if pdf=='gamma':
        #shift and clip to create intermittency
        if np.mean(C) != 0.:
            C = (C/np.mean(C) - 1).clip(min=0.)

        #renormalize
        if np.mean(C) != 0.:
            C = C/np.mean(C) * c

    elif (pdf=='lognorm') or (pdf=='lognormal'):
        #shift and clip to create intermittency
        if np.mean(C) != 0.:
            C = (C - cbase).clip(min=0)

    if return_seed: return C, seed
    return C


if __name__ == '__main__':
    main()
```

2. lognorm_solver.py

```
"""
@author: stsmith
"""

from scipy.optimize import fsolve
import scipy as sp
import numpy as np
import matplotlib.pyplot as plt
import warnings


def eqn_system(params, ip2, gamma):
    '''
    the equations themselves, which are run through scipy's solver
    '''
    ip2 = float(ip2)
    gamma = float(gamma)
    phi, sig = params
    if phi <= 0:
        return np.nan, np.nan

    term1 = 1-sp.special.erf(np.log(phi)/(np.sqrt(2)*sig))
    term2 = 1-sp.special.erf((np.log(phi)-(sig**2))/(np.sqrt(2)*sig))
    term3 = 1-sp.special.erf((np.log(phi)-2*(sig**2))/(np.sqrt(2)*sig))

    residualA = 0.5*term1 - gamma

    bigphi = (0.5*np.exp(0.5*(sig**2))*term2) - (0.5*phi*term1)
    phi2bar = (0.5*np.exp(2*(sig**2))*term3) - (phi*np.exp(0.5*(sig**2))*term2) +
    ↪  (0.5*(phi**2)*term1)
    residualB = gamma*phi2bar/(bigphi**2) - 1 - ip2

    return residualA, residualB


def lognorm_parameters(mean, var, param_test=False):
    '''
    takes mean and variance for concentration and computes
    relevant lognormal parameters for non-intermittent concentration series
    '''
    #inputs to system of eqns
    i2 = float(var)/(float(mean)**2)
    if i2 < 0.887: i2 = 0.887 #internal cutoff based on phi function minimum
    #if i2 < 3.56: i2 = 3.56 #internal cutoff based on phi(i2)=1
    ip2 = (2*i2)/(2+i2)
    gamma = (1+ip2)/(1+i2)

    #initial guesses based on empirical continuity of functions
    if i2 <= 0.3:
        phi_guess, sig_guess = 0.9, 0.1
    elif i2 <= 1:
       phi_guess, sig_guess = 0.7, 0.4
    elif i2 <= 10:
        phi_guess, sig_guess = 1., 1.
```

```
    elif i2 <= 30:
        phi_guess, sig_guess = 2., 1.
    elif i2 <= 100:
        phi_guess, sig_guess = 10, 1.1
    elif i2 <= 1e3:
        phi_guess, sig_guess = 20, 1.2
    #the below were found by fitting to initially calculated values
    elif (i2 > 1e3) and (i2<=1e4):
        a,b,c = 0.019587784159339417, 6.9849176149662497, 2.274841402556921
        phi_guess, sig_guess = c*(np.log(a*i2))**2+b, 1.3
    elif (i2 > 1e4) and (i2<=8e4):
        a,b,c = 8.8735303782044926e-05, 107.28967892260884, 18.105801201005235
        phi_guess, sig_guess = c*(np.log(a*i2))**2+b, 1.3
    else:
        a,b,c = 8.8735303782044926e-05, 107.28967892260884, 18.105801201005235
        phi_guess, sig_guess = c*(np.log(a*i2))**2+b, 1.4

    #solve system
    phi, sig = fsolve(eqn_system, (phi_guess, sig_guess), args=(ip2,gamma), maxfev=1000)

    #calculate pre-shifted parameters and shift itself
    c50 = float(mean)/(gamma*np.sqrt(ip2+1))
    cbase = phi*c50

    #return median, sigma, shift, and intermittency factor
    if param_test==True: return phi, sig
    return c50, sig, cbase, gamma

if __name__ == '__main__':
    '''
    call this script to test the lognorm_parameters() function
    only recommended for values of var/(mean**2) on the order of 1e5 or less
    '''
    #normalized mean square concentration values
    testvar = np.linspace(1e-1, 1e5, num=10001)
    testmean = np.ones_like(testvar).astype(float)

    #run for each value
    params = []
    for mean, var in zip(testmean, testvar):
        with warnings.catch_warnings():
            warnings.filterwarnings("ignore", category=RuntimeWarning)
            phi, sig = lognorm_parameters(mean, var, True)
        params.append([phi, sig])
    params = np.array(params)

    #prep figures
    fig = plt.figure(figsize=(8,12))
    ax0 = fig.add_subplot(211)
    ax1 = fig.add_subplot(212)

    #plot phi
    l0,=ax0.plot(testvar/(testmean**2), params[:,0], 'b', label='phi')
    plt.sca(ax0)
    plt.xscale('log')
    plt.yscale('log')
```

```
    #plot sig
    l1,=ax1.plot(testvar/(testmean**2), params[:,1], 'g', label='sig')
    plt.sca(ax1)
    plt.xscale('log')
    # plt.yscale('log')

    #add niceties and show
    plt.legend([l0,l1], ['phi','sig'], fontsize=20)
    plt.tight_layout()
    plt.show()
```

3. gamma_solver.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 25 09:50:03 2017

@author: sweinric
"""

from __future__ import division
from scipy.optimize import fsolve
import scipy as sp
import numpy as np
import math
import clipped_gamma as cg
import matplotlib.pyplot as plt
import warnings


def transcendentalSystem(params,mu,gamma):
    '''
    the equations themselves, which are run through scipy's solver
    '''
    k,r=params
    residualA = sp.special.gammaincc(k,r)-gamma
    residualB =
    ↪  (((gamma*r)+(1+k-r)*((gamma*(k-r))+(math.exp(-r)*(r**k)/sp.special.gamma(k)))) /
    ↪  (((gamma*(k-r))+(math.exp(-r)*(r**k)/sp.special.gamma(k)))**2))-mu

    return residualA,residualB


def find_parameters(mu):
    '''
    computes relevant pdf parameters for non-intermittent concentration series
    (shifted-clipped gamma pdf)
    '''
    #compute intermittency parameter (gamma)
    gamma=min(1.,3./mu)

    #trivial case
    if gamma==1:
        r = 0
        k = 1/(mu - 1)
```

```
    #run solver for all other input values
    else:
        try: guess = cg.param_solve(0,0, mu, gamma)
        except: guess = [1,1,1]
        k_guess = guess[0]
        r_guess = guess[2] / guess[1]

        warnings.filterwarnings('error')
        try:
            k,r = fsolve(transcendentalSystem,(k_guess,
            ↪  r_guess),args=(mu,gamma),maxfev=1000)
        except:
            upper, lower = find_edges(mu)
            k0,s0,l0,g0 = find_parameters(upper)
            k1,s1,l1,g1 = find_parameters(lower)

            k = np.interp(mu, [upper,lower], [k0,k1])
            s = np.interp(mu, [upper,lower], [s0,s1])
            l = np.interp(mu, [upper,lower], [l0,l1])
            gamma = np.interp(mu, [upper,lower], [g0,g1])
            return k, s, l, gamma
        warnings.resetwarnings()

    #calculate s and lambda based on k and r (lambda/s)
    s=1/((gamma*(k-r))+(math.exp(-r)*(r**k)/sp.special.gamma(k)))
    l=r*s

    return k, s, l, gamma


def find_edges(initial_mu, initial_step=0.001):
    '''
    finds bounds of troublesome function areas which break the solver.
    these bounds are used to linearly interpolate values for each parameter.
    '''
    warnings.filterwarnings('error')

    mu = initial_mu
    step = initial_step
    while True:
        gamma = min(1.,3./mu)
        try: guess = cg.param_solve(0,0, mu, gamma)
        except: guess = [1,1,1]
        k_guess = guess[0]
        r_guess = guess[2] / guess[1]

        try: k,r = fsolve(transcendentalSystem,(k_guess,
        ↪  r_guess),args=(mu,gamma),maxfev=1000)
        except:
            mu-=step
            step*=1.1
            continue
        break
    lower_bound = mu
```

```
    mu = initial_mu
    step = initial_step
    while True:
        gamma = min(1.,3./mu)
        try: guess = cg.param_solve(0,0, mu, gamma)
        except: guess = [1,1,1]
        k_guess = guess[0]
        r_guess = guess[2] / guess[1]

        try: k,r = fsolve(transcendentalSystem,(k_guess,
        ↪  r_guess),args=(mu,gamma),maxfev=1000)
        except:
            mu+=step
            step*=1.1
            continue
        break
    upper_bound = mu

    warnings.resetwarnings()
    return lower_bound, upper_bound


if __name__ == '__main__':
    '''
    call this script to test the find_parameters() function
    outputs a plot of k, s, and lambda for 1000 inputs with values
    between 1.2 and 8 (relevant for FUSION07 ensembles)
    '''
    #normalized mean square concentration values
    tester = np.linspace(1.2, 8, num=1001)
    dt = tester[1]-tester[0]

    #run for each value
    params = []
    flag = 0
    for value in tester:
        k, s, l, gamma = find_parameters(value)

        # except:
            # if flag==0:
                # print(value-dt)
                # flag = 1
            # continue
        # if flag==1:
            # print('{}\n'.format(value))
            # flag = 0

        params.append([k, s, l, gamma])
    params = np.array(params)

    #prep figures
    fig = plt.figure(figsize=(8,16))
    ax0 = fig.add_subplot(311)
    ax1 = fig.add_subplot(312)
    ax2 = fig.add_subplot(313)
```

```
    #plot k
    l0,=ax0.plot(tester, params[:,0], 'b', label='k')
    plt.sca(ax0)
    plt.xscale('log')
    plt.yscale('log')

    #plot s
    l1,=ax1.plot(tester, params[:,1], 'g', label='s')
    plt.sca(ax1)
    plt.xscale('log')
    plt.yscale('log')

    #plot lambda
    l2,=ax2.plot(tester, params[:,2], 'r', label='l')
    plt.sca(ax2)
    plt.xscale('log')

    #make the plot pretty and show it
    plt.sca(ax0)
    plt.legend([l0,l1,l2], ['k','s','lambda'], fontsize=20)
    plt.tight_layout()
    plt.show()
```

4. clipped_gamma.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 13 09:59:00 2017

@author: jwillert
"""

# This function is described in a multitude of papers.
# See either of the Gunatilaka papers.

# This function includes the nonlinear solver required
# in order to compute the parameters for the clipped
# gamma distribution.  Computation of the nonlinear
# residual and Jacobian are included in this file
# as well.

import scipy.special as special
import numpy as np
import math

def param_solve(c,v,nmsc,gamma):

    [k,w] = newton(gamma,nmsc,c)

    [s,l] = get_s_lambda(k,w,gamma)

    return k, s, l, gamma

def get_s_lambda(k,w,gamma):

    sinv = (-w + k)*gamma + (1/special.gamma(k+1))*pow(w,k)*np.exp(-w)
```

```
    s = 1.0/sinv
    l = w*s

    return [s,l]

def residual(k,w,gamma,nmsc):

    # Here, w = lambda / s

    # Equation 1 residual
    #print k, s, l
    sinv = (-w + k)*gamma + (1/special.gamma(k))*pow(w,k)*np.exp(-w)

    # Equation 2 residual
    num = (w*gamma + (-w + k + 1)*sinv)
    r1 = num/pow(sinv,2.0) - nmsc

    # Equation 3 residual
    #r2 = special.gammaincc(k,w)/special.gamma(k+1) - gamma
    r2 = special.gammaincc(k,w) - gamma
    #print gamma

    return [r1, r2]

def compute_jacobian(k,w,gamma,nmsc,delta,res):

    J = np.zeros([2,2])

    c0 = residual(k+delta,w,gamma,nmsc)
    c1 = residual(k,w+delta,gamma,nmsc)
    #print "res\n",res,"c0\n",c0, "c1\n",c1

    J[0][0] = (c0[0]-res[0])/delta
    J[1][0] = (c0[1]-res[1])/delta
    J[0][1] = (c1[0]-res[0])/delta
    J[1][1] = (c1[1]-res[1])/delta

    return J

def newton(gamma,nmsc,mean):

    if gamma == 1:

        w = 0
        k = 1/(nmsc - 1)

        x_new = [k, w]

    else:

        x_cur = np.matrix.transpose(np.matrix([1.00,0.00]))
        r_0 = residual(x_cur.item(0),x_cur.item(1),gamma,nmsc)
        r_0_norm = np.linalg.norm(r_0)
        delta = 0.0001
        r_new_norm = r_0_norm
```

```
for i in range(100):
    #print i, r_new_norm,"\n"
    #print x_cur
    r_cur = residual(x_cur.item(0),x_cur.item(1),gamma,nmsc)

    J = compute_jacobian(x_cur.item(0),x_cur.item(1),gamma,nmsc,delta,r_cur)
    r = np.matrix.transpose(np.matrix(r_cur))

    #print "J\n", J

    try: Jinv = np.linalg.inv(J)
    except:
        if r_new_norm < 2e-13: break
        else: raise ValueError('Not really converging.')

    #print "RES\n", r, "\njac\n", J, "\nJinv\n", Jinv, "\nxcur\n", x_cur

    r_cur_norm = np.linalg.norm(r_cur)

    r_new_norm = r_cur_norm + 1.0

    step = 2.0

    #print "step\n", Jinv*r
    while r_new_norm > 0.99*r_cur_norm:

        #print "steplen\n", step
        step = step/2

        x_new = x_cur - step*Jinv*r

        #(stuart) I added this to prevent a possible infinite loop case
        if step < 1e-15:
            break

        if math.isnan(x_new.item(0)) or math.isnan(x_new.item(1)):
            continue

        if x_new.item(0) < 0 or x_new.item(1) < 0 :
            continue

        try: r_new = residual(x_new.item(0),x_new.item(1),gamma,nmsc)
        except OverflowError: continue

        if math.isnan(r_new[0]) or math.isnan(r_new[1]):
            continue

        #print x_cur, "\n", x_new, "\n", r_new
        r_new_norm = np.linalg.norm(r_new)

        #print step
        if step < 0.001:
            break

    #print r_new, r_new_norm, x_new
```

```
        #print r_new_norm, r_new, x_new

        x_cur = x_new
        r_cur = r_new

        #NOTE: This isn't necessarily the tolerance. I added a couple
        #of lines above which gives a tolerance of 2e-13 if Jinv is no longer
        #calculable. Which means the below is more of a tolerance goal than a rule.
        if r_new_norm < 1e-15:
            break

        delta = np.min([0.000001, 0.01*r_new_norm/r_0_norm])

    #print J
    [k, w] = x_new
    k = k.max()
    w = w.max()
    x_new = [k, w]
return x_new
```

Appendix E.
Fractal Method Implementation

The fractal method was implemented in Python 2.7.14 and has dependencies on numpy 1.11.3, scipy 0.19.1, pandas 0.20.3, and numba 0.35.0 libraries. The python code used in this implementation is provided in this appendix.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jun 01 11:50:49 2017

@author: sweinric
"""
from __future__ import division
import numpy as np
import pandas as pd
import math
import scipy.interpolate as spi
import time
import os
import re
from numba import guvectorize, float64

#Define simulation parameters
suffix = '_lognormal_test'
numRefine = 4 #4
numRealizations = 100 #100
firstInterpResolution = 30 #30
ensembleNum=1

#Assign initial spatial and temporal grid spacing
if ensembleNum==1:
    initialSpatialResolution=26.66 # m ensemble#1
    initialTemporalResolution=22.60 # s ensemble#1
elif ensembleNum==2:
    initialSpatialResolution=25.37 # m ensemble#2
    initialTemporalResolution=24.30 # s ensemble#2
elif ensembleNum==3:
    initialSpatialResolution=29.66 # m ensemble#3
    initialTemporalResolution=23.79 # s ensemble#3
elif ensembleNum==4:
    initialSpatialResolution=26.11 # m ensemble#4
    initialTemporalResolution=22.96 # s ensemble#4
elif ensembleNum==5:
    initialSpatialResolution=21.41 # m ensemble#5
    initialTemporalResolution=24.90 # s ensemble#5

#Path to ensemble mean and variance data
ensemblePath = 'C:\\Users\\sweinric\\Documents\\HPAC_Variance\\Ensemble\\'
ensembleFilename = 'ensemble_' + str(ensembleNum) + '.csv'
fieldFilename = re.sub('\.csv$','',ensembleFilename)

startTime = time.time()

#Assign fractal dimension
fractalDimension = 1.3
alpha = 3-fractalDimension

ensembleMeasureFilename=
↪  '\\\\corp-proj\\archive\\3250\\325027_HPAC_Variance\\4_data\\FUSION_07\\measures\\ensemble_'
↪  + str(ensembleNum) + '_1s\\upc_frequency.csv'
ensembleMeasure = pd.read_csv(ensembleMeasureFilename)
concentrationLevels = ensembleMeasure['conc'].tolist()
```

```
outputDirectory = 'C:\\Users\\sweinric\\Documents\\HPAC_Variance\\3D_Plots\\' +
↪  fieldFilename + '_' + str(numRefine) + '_Refinements_' + str(numRealizations) +
↪  '_Realizations_Multifractal_vectorized' + suffix + '\\'
if not os.path.exists(outputDirectory):
    os.makedirs(outputDirectory)

################################################################################
#Function to relate the mean and variance field to gaussian parameters
@guvectorize([float64(float64[:],float64[:],float64[:],float64[:,:])],'(n),(n),(p)->(n,p)')
def parameterSolver(meanC,sdC,dummy, result):

    for i in range(meanC.shape[0]):
        if (meanC[i] > 0) and (sdC[i] > 0):
            squareMean = meanC[i]**2
            variance = sdC[i]**2
            meanG=np.log(squareMean/np.sqrt(variance+squareMean))
            sdG=np.sqrt(np.log((variance/squareMean)+1))
        else:
            meanG=0
            sdG=0
        result[i,0]=meanG
        result[i,1]=sdG
        result[i,2]=meanC[i]
        result[i,3]=sdC[i]

################################################################################
#Function the pulse shape, i.e. triangular pulses
def pulseFunc(dist,spacing):
    if dist<spacing:
        pulse = 1-(dist/spacing)
    else:
        pulse=0
    return pulse

################################################################################
#Function obtain field of gaussian parameters at the finest grid refinement.
#Interpolation is split into an initial course interpolation because direct triangulation
#of irregular data onto a fine grid is computationally expensive. The first interpolation
#interpolates the irregular data onto a course regular grid which is then interpolated
#again onto the finest grid refinement
def getEnsembleData(ensembleFilename):

    global xRegularGrid
    global yRegularGrid
    global tRegularGrid
    global pointData
    global meanData
    global interpGridXvals
    global interpGridYvals
    global outputAxisT
    global meanGridInterpData
    global nativeTimeResolution
    global sensorIDs
    global refLat
    global refLon
```

```
ensembleData = pd.read_csv(ensembleFilename,header=0)
timeList = [sum(x * int(t) for x, t in zip([3600, 60, 1], timeCode.split(":"))) for
↪  timeCode in ensembleData['time']]
ensembleData['time'] = timeList


lonValues = ensembleData['long']
latValues = ensembleData['lat']
refLon = sum(lonValues)/len(lonValues)
refLat = sum(latValues)/len(latValues)
lenLon = (math.pi/180)*6371000*math.cos(math.radians(refLat))
xValues = [(value-refLon)*lenLon for value in lonValues]
yValues = [(value-refLat)*111132 for value in latValues]

ensembleData['long']=xValues
ensembleData['lat']=yValues
sensorIDs = ensembleData['sensor'].unique()

meanData = ensembleData['mean'].as_matrix()
sdData = np.sqrt(ensembleData['var'].as_matrix())
pointData = ensembleData[['time','long','lat']].as_matrix()

samplerLocs = ensembleData[['long','lat']].drop_duplicates().reset_index()
tValues = ensembleData['time']
uniqueTimeList=sorted(tValues.unique())
nativeTimeResolution = uniqueTimeList[1]-uniqueTimeList[0]

xDomain =
↪  [min(xValues)-initialSpatialResolution,max(xValues)+initialSpatialResolution]
yDomain =
↪  [min(yValues)-initialSpatialResolution,max(yValues)+initialSpatialResolution]
tDomain = [min(tValues)-nativeTimeResolution,max(tValues)+nativeTimeResolution]

interpGridXvals = np.linspace(xDomain[0],xDomain[1],num=firstInterpResolution)
interpGridYvals = np.linspace(yDomain[0],yDomain[1],num=firstInterpResolution)
interpGridTvals =
↪  np.arange(tDomain[0],tDomain[1]+nativeTimeResolution,nativeTimeResolution)

initialAxisX,initialXSpacing = np.linspace(xDomain[0], xDomain[1],
↪  round((xDomain[1]-xDomain[0])/initialSpatialResolution), retstep=True)
initialAxisY,initialYSpacing = np.linspace(yDomain[0], yDomain[1],
↪  round((yDomain[1]-yDomain[0])/initialSpatialResolution), retstep=True)
initialAxisT,initialTSpacing = np.linspace(tDomain[0], tDomain[1],
↪  round((tDomain[1]-tDomain[0])/initialTemporalResolution), retstep=True)

outputAxisX,outputXResolution =
↪  np.linspace(xDomain[0],xDomain[1],(2+(len(initialAxisX)-1)*(2**(numRefine+1)))/2,
↪  retstep=True)
outputAxisY,outputYResolution =
↪  np.linspace(yDomain[0],yDomain[1],(2+(len(initialAxisY)-1)*(2**(numRefine+1)))/2,
↪  retstep=True)
outputAxisT,outputTResolution =
↪  np.linspace(tDomain[0],tDomain[1],(2+(len(initialAxisT)-1)*(2**(numRefine+1)))/2,
↪  retstep=True)
```

```
tRegularGrid,xRegularGrid,yRegularGrid =
↪  np.meshgrid(interpGridTvals,interpGridXvals,interpGridYvals)

outputGridT,outputGridX,outputGridY =
↪  np.meshgrid(outputAxisT,outputAxisX,outputAxisY)
outputCoordinates =
↪  np.array(zip(outputGridX.flatten(),outputGridT.flatten(),outputGridY.flatten()))

startTime = time.time()

if os.path.isfile(outputDirectory+'firstInterpFieldMean.npy') and
↪  os.path.isfile(outputDirectory+'firstInterpFieldSD.npy'):
    meanGridInterpData = np.load(outputDirectory+'firstInterpFieldMean.npy')
    sdGridInterpData = np.load(outputDirectory+'firstInterpFieldSD.npy')
else:
    meanGridInterpData =
    ↪  spi.griddata(pointData,meanData,(tRegularGrid,xRegularGrid,yRegularGrid),
    ↪  method='linear', fill_value=0)
    sdGridInterpData =
    ↪  spi.griddata(pointData,sdData,(tRegularGrid,xRegularGrid,yRegularGrid),
    ↪  method='linear', fill_value=0)
    np.save(outputDirectory+'firstInterpFieldMean.npy',meanGridInterpData)
    np.save(outputDirectory+'firstInterpFieldSD.npy',sdGridInterpData)

endTime = time.time()
print 'Time to convert to a regular grid: ' + str(endTime-startTime)

resolution = (outputXResolution,outputYResolution,outputTResolution)

if os.path.isfile('C:\\Users\\sweinric\\Documents\\HPAC_Variance\\Field_Data\\' +
↪  fieldFilename + '_refinements_' + str(numRefine) + suffix + '.npy'):
    fieldData = np.load('C:\\Users\\sweinric\\Documents\\HPAC_Variance\\Field_Data\\'
    ↪  + fieldFilename + '_refinements_' + str(numRefine) + suffix + '.npy')
else:
    startTime = time.time()
    if os.path.isfile(outputDirectory+'secondInterpFieldMean.npy') and
    ↪  os.path.isfile(outputDirectory+'secondInterpFieldSD.npy'):
        meanFieldData = np.load(outputDirectory+'secondInterpFieldMean.npy')
        sdFieldData = np.load(outputDirectory+'secondInterpFieldSD.npy')
    else:
        meanFieldData =
        ↪  spi.interpn((interpGridXvals,interpGridTvals,interpGridYvals),
        ↪  meanGridInterpData, outputCoordinates, fill_value=0)
        sdFieldData = spi.interpn((interpGridXvals,interpGridTvals,interpGridYvals),
        ↪  sdGridInterpData, outputCoordinates, fill_value=0)

        fieldData = np.reshape(meanFieldData,
        ↪  (len(outputAxisX),len(outputAxisT),len(outputAxisY)), order='C')

        np.save(outputDirectory+'secondInterpFieldMean.npy',meanFieldData)
        np.save(outputDirectory+'secondInterpFieldSD.npy',sdFieldData)

    fieldData = parameterSolver(meanFieldData,sdFieldData,[0,0,0,0])
    fieldData = np.reshape(fieldData,
    ↪  (len(outputAxisX),len(outputAxisT),len(outputAxisY),4), order='C')
    fieldData = np.moveaxis(fieldData,0,2)
```

```
        endTime = time.time()
        print 'Solve parameters: ' + str(endTime-startTime)
        np.save('C:\\Users\\sweinric\\Documents\\HPAC_Variance\\Field_Data\\' +
        ↪  fieldFilename + '_refinements_' + str(numRefine) + suffix + '.npy',
        ↪  fieldData)

    meanCdata = fieldData[:,:,:,2]
    sdCdata = fieldData[:,:,:,3]
    return initialAxisX, initialAxisY, initialAxisT, outputAxisX, outputAxisY,
    ↪  outputAxisT, resolution, fieldData, meanCdata, sdCdata, samplerLocs,
    ↪  set(timeList)
################################################################################

startTime=time.time()
initialAxisX, initialAxisY, initialAxisT, outputAxisX, outputAxisY, outputAxisT,
↪  resolution, fieldData, concGridData, sdGridData, samplerLocs, timeList =
↪  getEnsembleData(ensemblePath + ensembleFilename)

outputGridT,outputGridX,outputGridY =
↪  np.meshgrid(outputAxisT[1:-1],outputAxisX[1:-1],outputAxisY[1:-1])

meanGdata =  np.where(fieldData[:,:,:,0]==0,-np.inf,fieldData[:,:,:,0])
sdGdata = fieldData[:,:,:,1]
meanCdata = fieldData[:,:,:,2]
sdCdata = fieldData[:,:,:,3]
fieldGenerationEnd = time.time()

sumConcField =
↪  np.zeros((len(outputAxisT[1:-1]),len(outputAxisY[1:-1]),len(outputAxisX[1:-1])))
initialSDArray =
↪  np.sqrt(((sdGdata[1:-1,1:-1,1:-1])**2)/((((2/3)**3)*(1-(2**(-2*(numRefine+1)*alpha)))
↪  / (1-(2**(-2*alpha))))))

singleRlznIntrmtcy=list()
singleRlznUpFreq=list()
singleRlznUpDur=list()

ensembleTotalTime=0
ensembleTotalPeakTime=0
ensemblePeakCount=0

#Iterate over realizations to be simulated
for realization in range(numRealizations):
    concFluc =
    ↪  np.zeros((len(outputAxisT[1:-1]),len(outputAxisY[1:-1]),len(outputAxisX[1:-1])))
    pulseGenerationStart = time.time()
    #Iterate over refinement levels in a given realization
    for refineLevel in range(numRefine):
        print 'Working on refinement #' + str(refineLevel)
        refineLevelStart=time.time()

        xAxis,xSpacing = np.linspace(initialAxisX[0], initialAxisX[-1],
        ↪  (2+(len(initialAxisX)-1)*(2**(refineLevel+1)))/2, retstep=True)
        yAxis,ySpacing = np.linspace(initialAxisY[0], initialAxisY[-1],
        ↪  (2+(len(initialAxisY)-1)*(2**(refineLevel+1)))/2, retstep=True)
```

```
tAxis,tSpacing = np.linspace(initialAxisT[0], initialAxisT[-1],
↪  (2+(len(initialAxisT)-1)*(2**(refineLevel+1)))/2, retstep=True)

#Obtain p-factors used in the multifractal p-model implementation
if refineLevel>0:
    p=0.8
    stdDev = initialSDArray
    xPlist=list()
    for xParent in range(len(prevXaxis)):
        yPlist=list()
        for yParent in range(len(prevYaxis)):
            tPlist=list()
            for tParent in range(len(prevTaxis)):
                daughter=totalParray[tParent,yParent,xParent] *
                ↪  np.random.choice([p,p,p,p,1-p,1-p,1-p,1-p],(2,2,2),replace=False)
                tPlist.append(daughter)
            tParray = np.concatenate(tPlist)
            yPlist.append(tParray)
        yParray = np.concatenate(yPlist,axis=1)
        xPlist.append(yParray)
    totalParray = np.concatenate(xPlist,axis=2)
    totalParray = np.sqrt(totalParray[:-1,:-1,:-1])
else:
    totalParray = 1+np.zeros((len(tAxis),len(yAxis),len(xAxis)))
    stdDev = initialSDArray*(2**(0.5-refineLevel*alpha))

prevXaxis=xAxis
prevYaxis=yAxis
prevTaxis=tAxis

xAxis = xAxis[1:-1]
yAxis = yAxis[1:-1]
tAxis = tAxis[1:-1]

#Create array of random pulse amplitudes
pulseRandArray = np.multiply(np.random.normal(loc=0.0, scale=1,
↪  size=(len(tAxis),len(yAxis),len(xAxis))),totalParray[1:-1,1:-1,1:-1])
↪  #multiply this random array elementwise by multifractal p-model

#Calcualte random pulse offsets from gridpoints
pulseOffsetXList=np.random.uniform(-(xSpacing/2),(xSpacing/2),size=len(xAxis))
pulseOffsetYList=np.random.uniform(-(ySpacing/2),(ySpacing/2),size=len(yAxis))
pulseOffsetTList=np.random.uniform(-(tSpacing/2),(tSpacing/2),size=len(tAxis))
pulseCenterXList=pulseOffsetXList + xAxis
pulseCenterYList=pulseOffsetYList + yAxis
pulseCenterTList=pulseOffsetTList + tAxis

#Calcualte the pulse shape scaling at the gridpoint locations
interpXfunc = np.array([[pulseFunc(abs(outputGridpoint - pulseCenter),xSpacing)
↪  for pulseCenter in pulseCenterXList] for outputGridpoint in
↪  outputAxisX[1:-1]])
interpYfunc = np.array([[pulseFunc(abs(outputGridpoint - pulseCenter),ySpacing)
↪  for pulseCenter in pulseCenterYList] for outputGridpoint in
↪  outputAxisY[1:-1]])
```

```
    interpTfunc = np.array([[pulseFunc(abs(outputGridpoint - pulseCenter),tSpacing)
    ↪  for pulseCenter in pulseCenterTList] for outputGridpoint in
    ↪  outputAxisT[1:-1]])
    randomPulse =
    ↪  interpTfunc.dot(interpYfunc.dot(pulseRandArray.dot(interpXfunc.transpose())))

    #Multiply the standard deviation by the pulse scaling at the gridpoint locations
    concFlucLayer = np.multiply(stdDev,randomPulse)
    concFluc = concFluc + concFlucLayer
    refineLevelEnd = time.time()
    print 'Time to process refinement #' + str(refineLevel) + ': ' +
    ↪  str(refineLevelEnd-refineLevelStart)

#Add fluctuations to Gaussian mean field
concField=concFluc+meanGdata[1:-1,1:-1,1:-1]

#Exponentiate fluctuating gaussian field to obtain lognormal distribution
concField = np.exp(concField)
pulseGenerationEnd = time.time()
print 'Time to process a single realization: ' +
↪  str(pulseGenerationEnd-pulseGenerationStart)
print 'Total time: ' + str(pulseGenerationEnd-startTime)

#Create interpolation function for the simulated fluctuating concentration field
flucFieldFunc = spi.RegularGridInterpolator((outputAxisT[1:-1], outputAxisY[1:-1],
↪  outputAxisX[1:-1]), concField)

minLat=outputAxisY[1]
maxLat=outputAxisY[-2]
minLon=outputAxisX[1]
maxLon=outputAxisX[-2]
coreTime = [elem for elem in timeList if (elem >= outputAxisT[1]) and (elem <=
↪  outputAxisT[-2])]
coreSamplerLocs = np.array([samplerLocs.ix[index,['lat','long']].tolist() for index
↪  in range(samplerLocs.shape[0]) if (samplerLocs.ix[index,'lat'] >= minLat) and
↪  (samplerLocs.ix[index,'lat'] <= maxLat) and (samplerLocs.ix[index,'long'] >=
↪  minLon) and (samplerLocs.ix[index,'long'] <= maxLon)])

timeGrid,latGrid = np.meshgrid(coreTime,coreSamplerLocs[:,0])
timeGrid,lonGrid = np.meshgrid(coreTime,coreSamplerLocs[:,1])

coreInterpPoints = zip(timeGrid.flatten(),latGrid.flatten(),lonGrid.flatten())

#interpolate the fluctuating concentration field at the sampler locations
flucSamplerData = flucFieldFunc(coreInterpPoints)
flucSamplerData =
↪  np.reshape(flucSamplerData,(len(coreTime),samplerLocs['lat'].shape[0]))

np.save(outputDirectory + 'Fluc_Grid_Data_Realization_' + str(realization) + '.npy',
↪  concField)
np.save(outputDirectory + 'Fluc_Sampler_Data_Realization_' + str(realization) +
↪  '.npy', flucSamplerData)
```

Appendix F.
Illustrations

Figures

Tables

Appendix G.
References

Barad, Morton L. *Project Prairie Grass, A Field Program in Diffusion: Vol. I.* AFCRC-TR-58-235(I) GRP-59-VOL-1. Hanscom AFB, MA: Air Force Cambridge Research Labs, 1958.

Barad, Morton L. *Project Prairie Grass, A Field Program in Diffusion: Vol. II.* AFCRC-TR-58-235(II) GRP-59-VOL-2. Hanscom AFB, MA: Air Force Cambridge Research Labs, 1958.

Biltoft, Christopher A. *Customer Report for Mock Urban Setting Test.* DPG Document No. WDTC-FR-01-121. Dugway, UT: West Desert Test Center, U.S. Army Dugway Proving Ground, 2001.

Biltoft, Christopher A. *Concentration Fluctuation Modeling of Chemical Hazards (Assess Vulnerability).* Dugway, UT: U.S. Army Dugway Proving Ground, 1993.

Biltoft, Christopher A., Shayes D. Turley, T. B. Watson, G. H. Crescenti, and R. G. Carter. *Over-Land Atmospheric Dispersion (OLAD) Test Summary and Analysis*. WDTC-FR-99-016. Dugway, UT: U.S. Army Dugway Proving Ground, 1999.

Capability Development Document for the Joint Biological Tactical Detection System Increment 1. Washington, DC: Joint Requirements Office for Chemical, Biological, Radiological, and Nuclear Defense, 1 May 2012. DRAFT.

Capability Production Document for Joint Chemical Agent Detector Increment: I. Washington, DC: Joint Requirements Office for Chemical, Biological, Radiological, and Nuclear Defense, 4 June 2008.

Du, Shuming, David J. Wilson, and Eugene Yee. "A Stochastic Time Series Model for Threshold Crossing Statistics of Concentration Fluctuations in Non-Intermittent Plumes." *Boundary-Layer Meteorology* 92, no. 2 (1999): 229–241.

Finn, D., K.L. Clawson, R.M. Eckman et al. "Project Sagebrush Phase 1." NOAA Technical Memorandum OAR ARL-268. College Park, MD: National Oceanic Atmospheric Administration, 2015.

Gunatilaka, Ajith, Alex Skvortsov, and Ralph Gailis. "High Fidelity Simulation of Hazardous Plume Concentration Time Series Based on Models of Turbulent Dispersion." Paper presented at the 15th International Conference on Information Fusion (FUSION), Singapore, July 9-12, 2012.

Hanna, Steven R. "Concentration Fluctuations in a Smoke Plume." *Atmospheric Environment (1967)* 18, no. 6 (1984): 1091-1106.

Hilderman, T.L. and D.J. Wilson. "Simulating Concentration Fluctuation Time Series with Intermittent Zero Periods and Level Dependent Derivatives." *Boundary-Layer Meteorology* 91, no. 3 (1999): 451-482.

Kristensen, L., J. C. Weil, and J. C. Wyngaard. "Recurrence of High Concentration Values in a Diffusing, Fluctuating Scalar Field." *Boundary Layer Studies and Applications*. Dordrecht, Netherlands (1989): 263-276.

Lawrence, Alison E., Forrest R. Smith, John Alex Vig, and Charles E. Snyder. *User Manual for the Chemical and Biological Attack Consequence Estimator Version 1.1*. IDA Document D-8505. Alexandria, VA: Institute for Defense Analyses, November 2017.

Lawrence, Alison E., Scott L. Weinrich, Robert L. Cubeta, and John A. Vig. *Interpretation of Joint Biological Tactical Detection System (JBTDS) Requirements*. IDA Paper P-4870. Alexandria, VA: Institute for Defense Analyses, 2013. SECRET//NOFORN (only unclassified data used herein).

Lawrence, Alison E., Scott L. Weinrich, Jeffrey H. Grotte, Douglas P. Schultz, Nafis J. Upshur, W.M. Christensen, Christina M. Patterson, Eleanor L. Schwartz, Megan L. Prophet, and John N. Bombardt. *Joint Biological Tactical Detection System Analysis of Materiel Alternatives: Phase II*. IDA Paper P-4624. Alexandria, VA: Institute for Defense Analyses, March 2011. SECRET//NOFORN (only unclassified data used herein).

Lewellen, W.S. and R.I. Sykes. "Analysis of Concentration Fluctuations from Lidar Observations of Atmospheric Plumes." *Journal of Climate and Applied Meteorology* 25, no. 8 (1986).

Lovejoy, S. and B.B. Mandelbrot. "Fractal Properties of Rain, and a Fractal Model." *Tellus* 37A, no. 3 (1985).

Lukacs, Eugene. "A Characterization of the Gamma Distribution." *The Annals of Mathematical Statistics* 26, no. 2 (1955): 319-324.

Platt, Nathan, Dennis DeRiggi, Steve Warner, Paul Bieringer, George Bieberbach, Andrzej Wyszgrodzki, and Jeffrey Weil. "Method for Comparison of Large Eddy Simulation Generated Wind Fluctuations with Short-Range Observations." *International Journal of Environment and Pollution* 48, no. 1-4 (2012).

PubChem. "Propylene." Accessed August 31, 2018, https://pubchem.ncbi.nlm.nih.gov/compound/Propene.

Roberts, Philip J.W. and Donald R. Webster. "Turbulent Diffusion." Chap. 1 in *Environmental Fluid Dynamics: Theories and Applications,* edited by Hayley H. Shen, Alexander H.D. Cheng, Keh-Han Wang, Michelle H. Teng, and Clark C. Liu. Reston, VA: ASCE Publications, 2002.

Serfozo, Richard. *Basics of Applied Stochastic Processes*. Berlin, Germany: Springer Science & Business Media, 2009.

Sreenivason, K.R. and C. Meneveau. "The Fractal Facets of Turbulence." *Journal of Fluid Mechanics* 173 (1986).

Storwald, D. P. "Detailed Test Plan for the Fusing Sensor Information from Observing Networks (Fusion) Field Trial (FFT-07)," Document No. WDTC-TP-07-078. Dugway, UT: U.S. Army Dugway Proving Ground, Meteorology Division, West Desert Test Center.

Sykes, R.I. and R.S. Gabruk. "Fractal Representation of Turbulent Dispersing Plumes." *Journal of Applied Meteorology* 33, no. 6 (1994).

Sykes, R.I., R.S. Gabruk, and D.S. Henn. "The Small-scale Structure of Dispersing Clouds in the Atmosphere." A.R.A.P. Report No. 710. San Diego, CA: Titan Corporation, July 1994.

Sykes, R. Ian, Stephen F. Parker, Douglas S. Henn, and Biswanath Chowdhury. "SCIPUFF Version 2.7 Technical Documentation." Sage Management, December 2011.

Tirabassi, T. "Solutions of the Advection-Diffusion Equation." *Transactions on Ecology and the Environment* 21 (1997).

Townsend, Albert A. *The Structure of Turbulent Shear Flow*. Cambridge, UK: Cambridge University Press, 1980.

Wilson, David J. *Concentration Fluctuations and Averaging Time in Vapor Clouds.* New York, NY: Center for Chemical Process Safety of the American Institute of Chemical Engineers, 1995.

Yee, Eugene, P. R. Kosteniuk, G. M. Chandler, C. A. Biltoft, and J. F. Bowers. "Recurrence Statistics of Concentration Fluctuations in Plumes within a Near-Neutral Atmospheric Surface Layer." *Boundary-Layer Meteorology* 66, no. 1-2 (1993): 127-153.

Yee, Eugene and R. Chan. "A Simple Model for the Probability Density Function of Concentration Fluctuations in Atmospheric Plumes." *Atmospheric Environment* 31, no. 7 (1997): 991–1002.

Yee, Eugene, R. Chan, P.R. Kosteniuk, G.M. Chandler, C.A. Biltoft, and J.F. Bowers. "Experimental Measurements of Concentration Fluctuations and Scales in a Dispersing Plume in Atmospheric Surface Layer Obtained Using a Very Fast Response Concentration Detector." *Journal of Applied Meteorology and Climatology* 33, no. 8 (August 1994).

Yee, Eugene, R. Chan, P.R. Kosteniuk, G.M. Chandler, C.A. Biltoft, and J.F. Bowers. "Measurements of Level-Crossing Statistics of Concentration Fluctuations in Plumes Dispersing in the Atmospheric Surface Layer." *Boundary-Layer Meteorology* 73, no. 1-2 (1995): 53-90.

This page is intentionally blank.

Appendix H.
Abbreviations

AT&D	atmospheric transport and dispersion
CBRN	chemical, biological, radiological, and nuclear
CFD	computational fluid dynamics
DPID	digital photoionization detector
DTRA	Defense Threat Reduction Agency
FFT 07	Fusion Field Trial 2007
HPAC	Hazard Prediction and Assessment Capability
JEM	Joint Effects Model
LES	large eddy simulation
m	meter
PDF	probability density function
QC	quality control
s	second
SCIPUFF	Second-order Closure Integrated Puff
UTC	Coordinated Universal Time
UVIC	ultraviolet ion collector

This page is intentionally blank.

Proposed Distribution Lists for Final Version of

P-10487 H 19-000063/1

Implementation and Testing of Two Methods for Incorporating Concentration Fluctuations into the Hazard Prediction and Assessment Capability

Last Updated: 21-Feb-2019

Distribution	Paper	CDs/DVDs	PDFs via E-mail
Executive Office			
Dr. David S. Chu, President	1	0	0
Mr. Philip L. Major, Vice President, Programs	1	0	0
SFRD			
ADM John C. Harvey, Jr. USN (Ret), Division Director	0	0	1
Dr. Katherine M. Sixt, Assistant Director, SFRD	0	0	1
Dr. Alison E. Lawrence, Research Staff Member	0	0	1
Mr. Stuart Smith, Research Associate	0	0	1
Mr. Robert L. Cubeta, Research Associate	0	0	1
Dr. James F. Heagy, Research Staff Member	0	0	1
Dr. Ivo K. Dimitrov, Research Staff Member	0	0	1
Dr. Nathan Platt, Research Staff Member	0	0	1
Dr. Emily D. Heuring, Research Staff Member	0	0	1
Dr. Breeana G. Anderson, Research Staff Member	0	0	1
Mr. Scott L. Weinrich, Research Associate	0	0	1
Ms. Cherise J. Barnes, Project Assistant	0	0	1
Ms. Catherine E. Shuster, Research Associate	0	0	1
Dr. Jeffry T. Urban, Research Staff Member	0	0	1
Other Divisions			
IDA Library	0	0	1
Internal Totals:	**2**	**0**	**14**

Distribution	Paper	CDs/DVDs	PDFs via E-mail
Sponsor's Office			
Richard Fry richard.n.fry.civ@mail.mil	0	0	1
John Hannan, PhD john.r.hannan2.civ@mail.mil	0	0	1
US Government			
Aaron Hyre aaron.m.hyre.civ@mail.mil	0	0	1
Mike Retford michael.j.retford.civ@mail.mil	0	0	1
Mr. Jeffrey A. Steel jeffrey.a.steel.civ@mail.mil	0	0	1
Mr. Brandon Fischer brandon.j.fischer4.civ@mail.mil	0	0	1
Mr. Ryan Cahall ryan.cahall@censeoinsight.com	0	0	1
Mr. Artemas Herzog art.herzog@censeoinsight.com	0	0	1
Thomas Mazolla thomas.a.mazzola.ctr@mail.mil	0	0	1
Alberto Pareja-Lecaros alberto.lecaros@cubic.com	0	0	1
Defense Technical Information Center DTIC-OCA/Acquisition Team 8725 John J. Kingman Rd. Ft. Belvoir, VA 22060-6218 (703) 767-8040 jrike@dtic.mil	0	0	1
DTRIAC	0	0	1
External Totals:	**0**	**0**	**12**

Date Distribution Completed _____