Deliverable D1.12

Modelling of extreme wind conditions at higher atmospheres

Part 2:

Constrained stochastic simulation of spatial wind gusts

Agreement n.:	308974
Duration	November 2012 – October 2017
Co-ordinator:	DTU Wind

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7-ENERGY-2012-1-2STAGE under grant agreement No. 308974 (INNWIND.EU).

Document information

Document Name:	**Constrained stochastic simulation of spatial wind gusts**
Document Number:	**Deliverable D 1.1.2 – Part 2**
Author:	**René Bos and Wim Bierbooms (TU Delft)**
Document Type	Report
Dissemination level	PP
Review:	**Jens Tambke (ForWind / University of Oldenburg)**
Date:	**February 2015**
WP:	**WP1: Conceptual Design**
Task:	**Task 1.1: External Conditions**
Approval:	Approved by Task Leader

TABLE OF CONTENTS

1 INTRODUCTION 5

2 CONSTRAINED STOCHASTIC SIMULATION 6
  2.1 The 1D solution 6
  2.2 The 3D solution 7
  2.3 Setting constraints on volumes 9
  2.4 Setting multiple constraints 11

3 STATISTICS OF SPATIAL GUSTS 15
  3.1 Rice's formula 15
  3.2 Level excursions in $\mathbb{R}\mathbf{3}$ 16
  3.3 A special note on low-pass filters 18
  3.4 Gust probability 18

4 VALIDATION 21
  4.1 Data source 21
  4.2 Gust shapes 23
  4.3 Probability of occurrence 26

5 APPLICATION 31
  5.1 The method in a nutshell 31
  5.2 A step-by-step guide 32

6 CONCLUSION 35

A APPENDIX: EXAMPLE MATLAB CODES 36
  A.1 A local velocity maximum (*example 1*) 36
  A.2 A local velocity maximum in 3D space (*example 2*) 37
  A.3 A local maximum averaged over a cubic subvolume (*example 3*) 39
  A.4 One-dimensional velocity jump over a distance (*example 4*) 41
  A.5 A cluster of three single-point velocity maxima (*example 5*) 42
  A.6 Velocity jump between two planes (*example 6*) 44
  A.7 Return levels for wind speeds in a domain using the IEC NTM *(example 7)* 46

B MATLAB FUNCTION TO GENERATE IEC-COMPATIBLE GUSTS 50

1 INTRODUCTION

The design of very large wind turbines demands a different way of approaching extreme gust loads. At present, IEC standards require a design to withstand a 50-year extreme load arising from stochastic turbulence. However, especially during conceptual design, this 50-year return level has to be extrapolated from relatively short time series, leading to considerable uncertainty. In addition, designs need to handle a Mexican hat wavelet with a perturbation velocity that is uniform over the rotor plane. Clearly, such a gust shape becomes very unrealistic as rotor diameters increase.

More realistic gust shapes can be obtained through *constrained stochastic simulation*[1]. This method allows a designer to generate extremes in a time series, while adhering to the statistics of turbulence. However, the method has not yet been fully extended to a 3D domain. Moreover, quantifying the statistics of extremes in a 3D domain has been an issue.

This report shows how extreme gusts can be generated in a 3D domain, and how these gusts can be connected to a certain probability of occurrence. First, the method of generating extremes is explained in chapter 2 through various examples. Chapter 3 then deals with the statistics of these extremes. Furthermore, a comparison to real-life measurements is included in chapter 4. A general guide to the method, targeted at designers, can be found in chapter 5. Finally, chapter 6 presents the conclusions.

[1] Bierbooms, W. A. A. M. (2005). “Constrained Stochastic Simulation -- Generation of Time Series around Some Specific Event in a Normal Process.” *Extremes* 8 (3): 207–224. doi: 10.1007/s10687-006-7968-7.

2 CONSTRAINED STOCHASTIC SIMULATION

Constrained stochastic simulation is a method to embed events of certain properties in a field of stochastic turbulence. This way, a designer can easily simulate situations where a structure may experience extreme loading, for example by an N-year extreme gust. The basic principles are best explained through the 1D solution, e.g. one velocity component in one direction. Although it is already well-documented in literature[2], it is repeated here to introduce the reader to the subject and to provide clear use of notation. The remaining part of this chapter will deal with the full 3D solution and its applications.

2.1 The 1D solution

A one-dimensional, single-component process, $u(x)$, can be expressed as a Fourier series by

$$u(x) \approx \sum_{\kappa} \sqrt{\frac{2\pi E(\kappa)}{L_x}} n(\kappa) \mathrm{e}^{\mathrm{i}\kappa x}, \tag{2.1}$$

where $E(\kappa)$ is the spectral density function, L_x the domain size in x-direction, $n(\kappa) \sim \mathrm{N}(0,1)$ are zero-centered, normally distributed Fourier coefficients, and κ is the wave number. This summation can be reformulated as a dot product, $u(x) \approx \boldsymbol{\psi}(x) \cdot \mathbf{n}$, where $\boldsymbol{\psi}$ is the DFT (Discrete Fourier Transform) matrix:

$$u(x) \approx \sqrt{\frac{2\pi}{L_x}} \begin{bmatrix} \sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1 x} & \sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2 x} & \cdots & \sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N x} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix}. \tag{2.2}$$

Now, define a vector of constraints:

$$\mathbf{b} = \begin{bmatrix} u(x_0) \\ \dot{u}(x_0) \\ \ddot{u}(x_0) \end{bmatrix}, \tag{2.3}$$

to specify an event at x_0. For example, a local velocity maximum corresponds to $u(x_0) > 0$, $\dot{u}(x_0) = 0$, and $\ddot{u}(x_0) < 0$. The DFT matrix can then be expanded to include first and second order derivatives, yielding a linear system, $\mathbf{An} = \mathbf{b}$:

$$\sqrt{\frac{2\pi}{L_x}} \begin{bmatrix} \sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1 x_0} & \sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2 x_0} & \cdots & \sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N x_0} \\ \mathrm{i}\kappa_1\sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1 x_0} & \mathrm{i}\kappa_2\sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2 x_0} & \cdots & \mathrm{i}\kappa_N\sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N x_0} \\ -\kappa_1^2\sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1 x_0} & -\kappa_2^2\sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2 x_0} & \cdots & -\kappa_N^2\sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N x_0} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix} = \begin{bmatrix} u(x_0) \\ \dot{u}(x_0) \\ \ddot{u}(x_0) \end{bmatrix}.$$

In the case of extreme gusts, the third constraint can often be omitted, considering that it is unlikely for $u(x_0)$ to be a local minimum when the amplitudes are high positive (or vice versa in the case of negative amplitudes). The constrained vector, $\mathbf{n}_c$, which satisfies the above is given by the *Woodbury matrix identity*:

$$\mathbf{n}_c = \mathbf{n} + \boldsymbol{\Sigma}\mathbf{A}^*(\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^*)^{-1}(\mathbf{b} - \mathbf{An}), \tag{2.4}$$

where the asterisk, $\mathbf{A}^*$, denotes the conjugate transpose of $\mathbf{A}$ and $\boldsymbol{\Sigma} = \mathrm{E}[\mathbf{nn}^{\mathrm{T}}]$ is the covariance matrix of $\mathbf{n}$. Since $\mathbf{n}$ has unit variance, the $\boldsymbol{\Sigma}$ is an identity matrix and equation (2.4) reduces to

$$\mathbf{n}_c = \mathbf{n} + \mathbf{A}^*(\mathbf{AA}^*)^{-1}(\mathbf{b} - \mathbf{An}). \tag{2.5}$$

A Fourier transform of $\mathbf{n}_c$ then straightforwardly yields the constrained time series.

---

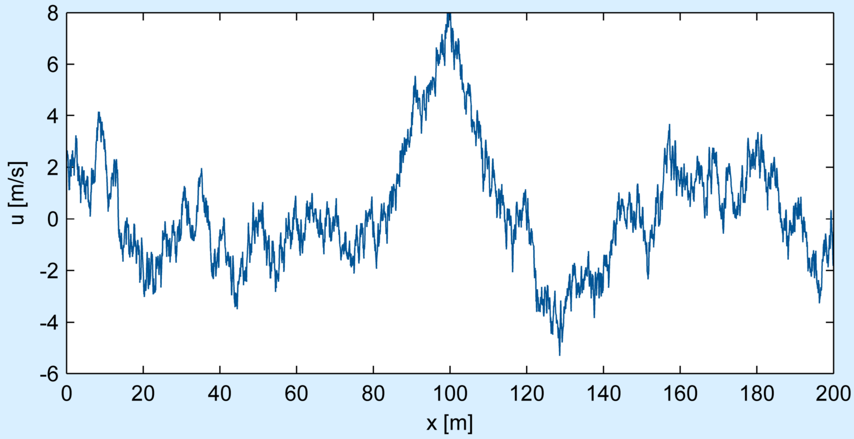
[2] Bierbooms, W. A. A. M. (2005). "Constrained Stochastic Simulation -- Generation of Time Series around Some Specific Event in a Normal Process." *Extremes* 8 (3): 207–224. doi: 10.1007/s10687-006-7968-7.

Example 1 *A local velocity maximum*

Consider a turbulent wind field $u(x)$, where $0 \leq x \leq 200$ m, with a longitudinal standard deviation of $\sigma_1 = 1.5$ m/s and an isotropic von Kármán energy spectrum in accordance with IEC 61400-1 annex B.1. Say one wants to place an 8 m/s velocity peak at $x = 100$ m. In that case, the linear system $\mathbf{An} = \mathbf{b}$, omitting any second derivatives, becomes

$$\sqrt{\frac{2\pi}{L_x}}\begin{bmatrix} \sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1\cdot 100} & \sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2\cdot 100} & \cdots & \sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N\cdot 100} \\ \mathrm{i}\kappa_1\sqrt{E(\kappa_1)}\mathrm{e}^{\mathrm{i}\kappa_1\cdot 100} & \mathrm{i}\kappa_2\sqrt{E(\kappa_2)}\mathrm{e}^{\mathrm{i}\kappa_2\cdot 100} & \cdots & \mathrm{i}\kappa_N\sqrt{E(\kappa_N)}\mathrm{e}^{\mathrm{i}\kappa_N\cdot 100} \end{bmatrix}\begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \end{bmatrix}.$$

Here, the variance is included in the matrix $\mathbf{A}$ to make $\mathbf{n}$ a vector of standard-normal distributed coefficients. An inverse Fourier transform of the constrained Fourier coefficients then results in the time series shown below.



2.2 The 3D solution

In three dimensions, the Fourier series is given by

$$\mathbf{u}(\mathbf{x}) \approx \sum_{\boldsymbol{\kappa}} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot\mathbf{x}}\mathbf{C}(\boldsymbol{\kappa})\mathbf{n}(\boldsymbol{\kappa}), \tag{2.6}$$

where $\mathbf{u} = [u, v, w]^{\mathrm{T}}$ is a velocity vector, $\mathbf{x} = [x, y, z]^{\mathrm{T}}$ a position vector, $\boldsymbol{\kappa} = [\kappa_x, \kappa_y, \kappa_z]^{\mathrm{T}}$ the wave number vector, $\mathbf{C}(\boldsymbol{\kappa})$ a correlation tensor, and $\mathbf{n}(\boldsymbol{\kappa}) \sim \mathrm{N}(0, \mathbf{I}_3)$ a vector of standard-normal distributed coefficients[3]. For multiple dimensions, the linear system, $\mathbf{An} = \mathbf{b}$, has to be expanded. This can be achieved by rewriting the multiple summation to a single sum, e.g.

$$\begin{array}{ccccccc} a_{1,1} & + & a_{1,2} & + & \cdots & + & a_{1,N_y} \\ + & & + & & & & + \\ a_{2,1} & + & a_{2,2} & + & \cdots & + & a_{2,N_y} \\ + & & + & & & & + \\ \vdots & & \vdots & & \ddots & & \vdots \\ + & & + & & & & + \\ a_{N_x,1} & + & a_{N_x,2} & + & \cdots & + & a_{N_x,N_y} \end{array}$$

$$\downarrow$$

$$a_{1,1} + a_{1,2} + \cdots + a_{1,N_y} + a_{2,1} + a_{2,2} + \cdots + a_{2,N_y} + \cdots + a_{N_x,1} + a_{N_x,2} + \cdots + a_{N_x,N_y}$$

[3] Alternatively, the correlation tensor can be included in the covariance matrix such that $\mathbf{n} \sim \mathrm{N}(0, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} = \mathbf{C}^*\mathbf{C}$. However, this may lead to quite bulky matrices when working with large domain sizes. For three dimensions, $\boldsymbol{\Sigma} = \mathrm{E}[\mathbf{nn}^{\mathrm{T}}]$ is a $N_xN_yN_z \times N_xN_yN_z$ matrix.

Example 2 *A local velocity maximum in 3D space*

Consider a turbulent wind field $\mathbf{u}(\mathbf{x})$, where $0 \leq x \leq 200$ m, $0 \leq y \leq 200$ m, $0 \leq z \leq 200$ m, with an isotropic standard deviation of $\sigma_{\text{iso}} = 1.5$ m/s and a von Kármán isotropic energy spectrum in accordance with IEC 61400-1 annex B.1. The isotropic covariance matrix is given by

$$\mathbf{C}_{\text{iso}} = \sigma_{\text{iso}} \sqrt{\frac{2\pi^2 \ell^3 E(\kappa)}{L_x L_y L_z \kappa^4}} \begin{bmatrix} 0 & \kappa_z & -\kappa_y \\ -\kappa_z & 0 & \kappa_x \\ \kappa_y & -\kappa_x & 0 \end{bmatrix},$$

where ℓ is the turbulence length scale and

$$\kappa = \sqrt{\kappa_x^2 + \kappa_y^2 + \kappa_z^2}.$$

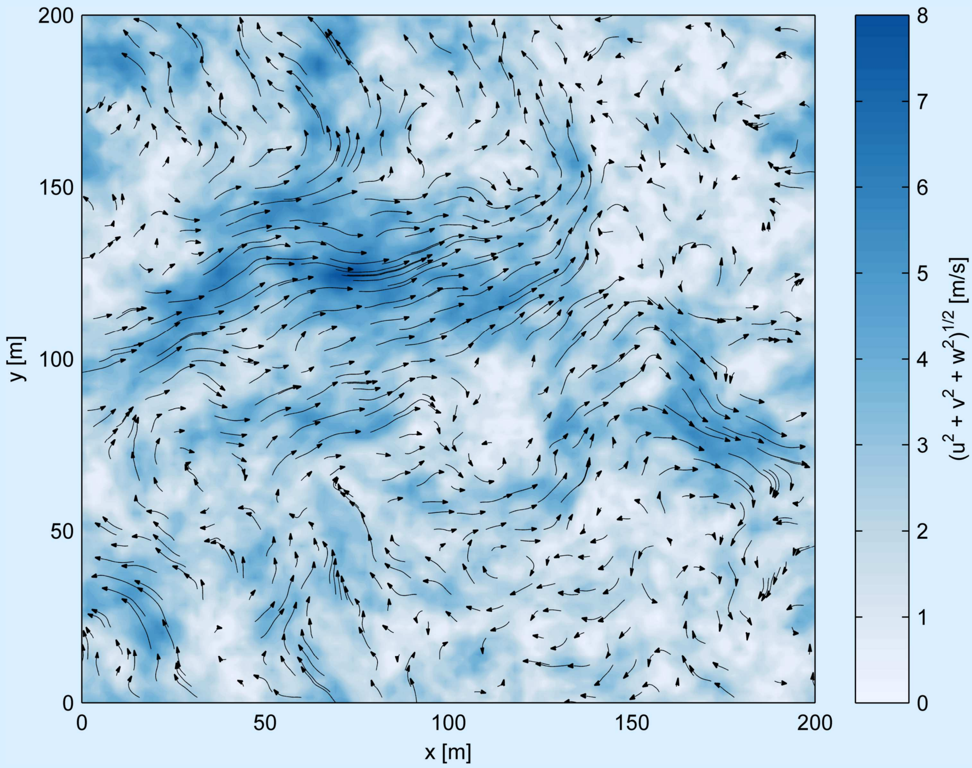
Say one wants to place an 8 m/s velocity peak, in horizontal direction, at $\mathbf{x} = [75, 125, 100]^{\text{T}}$ m. In that case, the linear system $\mathbf{An} = \mathbf{b}$, again omitting any second derivatives, becomes

$$\begin{bmatrix} \mathbf{C}_{1,1,1} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1} \cdot [75,125,100]^{\text{T}}} & \mathbf{C}_{2,1,1} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1} \cdot [75,125,100]^{\text{T}}} & \cdots \\ \mathrm{i}\kappa_x \mathbf{C}_{1,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1} \cdot [75,125,100]^{\text{T}}} & \mathrm{i}\kappa_x \mathbf{C}_{2,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1} \cdot [75,125,100]^{\text{T}}} & \cdots \\ \mathrm{i}\kappa_y \mathbf{C}_{1,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1} \cdot [75,125,100]^{\text{T}}} & \mathrm{i}\kappa_y \mathbf{C}_{2,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1} \cdot [75,125,100]^{\text{T}}} & \cdots \\ \mathrm{i}\kappa_z \mathbf{C}_{1,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1} \cdot [75,125,100]^{\text{T}}} & \mathrm{i}\kappa_z \mathbf{C}_{2,1,1}^{(u)} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1} \cdot [75,125,100]^{\text{T}}} & \cdots \end{bmatrix} \begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \end{bmatrix} = \begin{bmatrix} 8 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where the matrix $\mathbf{C}^{(u)}$ corresponds to the first row, i.e.

$$\mathbf{C}_{\text{iso}}^{(u)} = \sigma_{\text{iso}} \sqrt{\frac{2\pi^2 \ell^3 E(\kappa)}{L_x L_y L_z \kappa^4}} [0 \quad \kappa_z \quad -\kappa_y].$$

An inverse 3D Fourier transform of the constrained Fourier coefficients then results in the velocity field shown below. A slice is made through the field at $z = 100$ m.



The sum with respect to the wave number vector, $\boldsymbol{\kappa}$, then denotes the triple sum:

$$\sum_{\boldsymbol{\kappa}} \mathbf{n}(\boldsymbol{\kappa}) = \sum_{\kappa_x}\sum_{\kappa_y}\sum_{\kappa_z} \mathbf{n}(\kappa_x, \kappa_y, \kappa_z) = \sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_z} \mathbf{n}(\boldsymbol{\kappa}_{i,j,k}).$$

Using this notation, the DFT matrix multiplication $\mathbf{u}(\mathbf{x}) \approx \boldsymbol{\Psi}(x)\mathbf{n}$ now becomes

$$\mathbf{u}(\mathbf{x}) \approx \begin{bmatrix} \mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}} & \mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}} & \cdots & \mathbf{C}_{N_x,N_y,N_z}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{N_x,N_y,N_z}\cdot\mathbf{x}} \end{bmatrix} \begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \\ \mathbf{n}_{N_x,N_y,N_z} \end{bmatrix}. \tag{2.7}$$

Since a purely horizontal gust requires

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial y} = \frac{\partial u}{\partial z} = 0, \tag{2.8}$$

the linear system, omitting the second derivatives, can be set up as follows:

$$\begin{bmatrix} \mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots & \mathbf{C}_{N_x,N_y,N_z}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{N_x,N_y,N_z}\cdot\mathbf{x}_0} \\ \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots & \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{N_x,N_y,N_z}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{N_x,N_y,N_z}\cdot\mathbf{x}_0} \\ \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots & \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{N_x,N_y,N_z}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{N_x,N_y,N_z}\cdot\mathbf{x}_0} \\ \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots & \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{N_x,N_y,N_z}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{N_x,N_y,N_z}\cdot\mathbf{x}_0} \end{bmatrix} \begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \\ \mathbf{n}_{N_x,N_y,N_z} \end{bmatrix} = \begin{bmatrix} u(\mathbf{x}_0) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where $\mathbf{C}^{(u)}_{i,j,k}$ denotes the u-component, or first row, of the correlation tensor[4], e.g.

$$\mathbf{C}^{(u)} = [C_{uu} \quad C_{uv} \quad C_{uw}].$$

Again, this can be solved for the constrained vector $\mathbf{n}_c$.

2.3 Setting constraints on volumes

Although velocity amplitude may be a good measure of the damage inflicted by a gust acting on a single point, for spatial structures this velocity needs to act on a significant area. In that case, it might be convenient to switch from single-point amplitudes to velocities averaged over space.

For an unsteady velocity field, the mean velocity of a fluid parcel of volume $V = \lambda_x\lambda_y\lambda_z$, at a position $\mathbf{x}$, can be expressed as

$$\bar{\mathbf{u}}(\mathbf{x}) = \frac{1}{\lambda_x\lambda_y\lambda_z}\int_V \mathbf{u}(\mathbf{x}+\mathbf{r})\mathrm{d}\mathbf{r}, \tag{2.9}$$

where

$$\int \mathrm{d}\mathbf{r} = \iiint \mathrm{d}r_x\mathrm{d}r_y\mathrm{d}r_z.$$

Now, let the velocity field by represented by a Fourier-Stieltjes integral

$$\mathbf{u}(\mathbf{x}) = \int_{\boldsymbol{\kappa}} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot\mathbf{x}}\mathrm{d}\mathbf{Z}(\boldsymbol{\kappa}). \tag{2.10}$$

[4] Therefore, the leftmost matrix, $\mathbf{A}$, has 3+1+1+1 rows, which equals the number of rows on the right-hand side.

Example 3 *A local maximum averaged over a cubic subvolume*

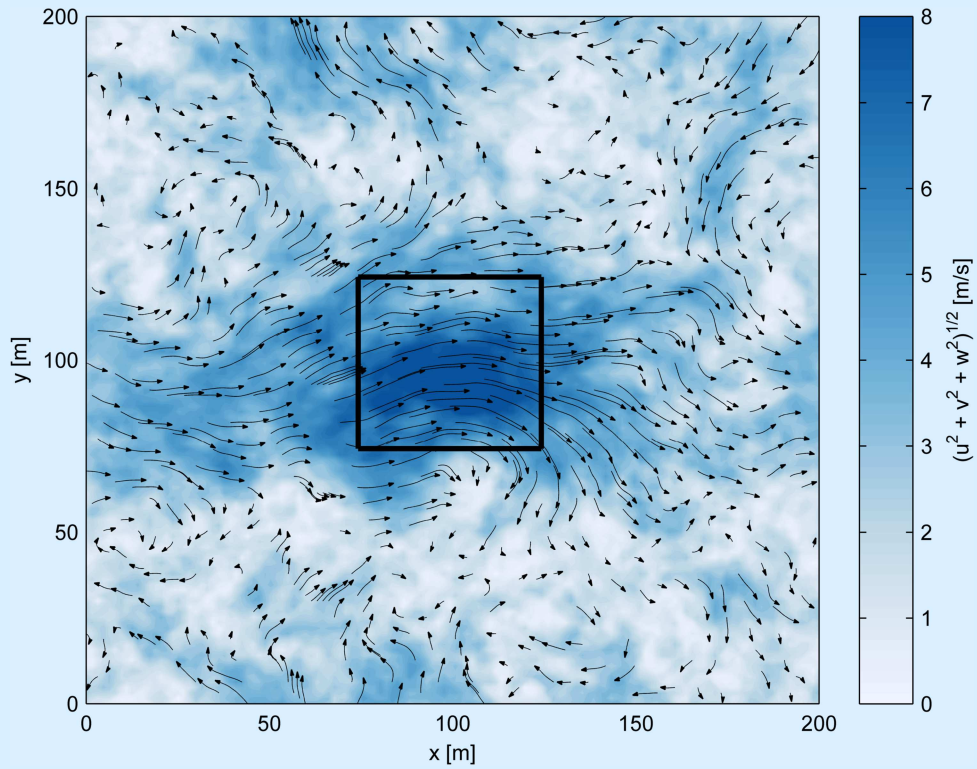
Consider a turbulent wind field $\mathbf{u}(\mathbf{x})$, where $0 \leq x \leq 200$ m, $0 \leq y \leq 200$ m, $0 \leq z \leq 200$ m, with an isotropic standard deviation of $\sigma_{\text{iso}} = 1.5$ m/s and a von Kármán isotropic energy spectrum in accordance with IEC 61400-1 annex B.1. Given is a cubic subvolume at $\mathbf{x} = [100, 100, 100]^{\mathrm{T}}$ m with dimensions $\lambda_x = 50$ m, $\lambda_y = 50$ m, and $\lambda_z = 50$ m. For the average horizontal velocity inside the subvolume to be, say, 5 m/s, define a filter using equation (2.11) or (2.12):

$$G(\boldsymbol{\kappa}) = \operatorname{sinc}\left(\frac{50\kappa_x}{2}\right)\operatorname{sinc}\left(\frac{50\kappa_y}{2}\right)\operatorname{sinc}\left(\frac{50\kappa_z}{2}\right).$$

From there on, the procedure is equal to the previous example, only with the inclusion of the filter:

$$\begin{bmatrix} G(\boldsymbol{\kappa})\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,100,100]^{\mathrm{T}}} & G(\boldsymbol{\kappa})\mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,100,100]^{\mathrm{T}}} & \cdots \end{bmatrix} \begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

A slice at $z = 100$ m. then yields the velocity field shown below.



Then it holds that

$$\begin{aligned} \bar{\mathbf{u}}(\mathbf{x}) &= \frac{1}{\lambda_x\lambda_y\lambda_z}\int_V \left[\int_{\boldsymbol{\kappa}} \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot(\mathbf{x}+\mathbf{r})}\mathrm{d}\mathbf{Z}(\boldsymbol{\kappa})\right]\mathrm{d}\mathbf{r}\,, \\ &= \frac{1}{\lambda_x\lambda_y\lambda_z}\int_{\boldsymbol{\kappa}} \left[\int_V \mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot(\mathbf{x}+\mathbf{r})}\mathrm{d}\mathbf{r}\right]\mathrm{d}\mathbf{Z}(\boldsymbol{\kappa})\,, \end{aligned}$$

$$= \frac{1}{\lambda_x\lambda_y\lambda_z}\int_{\boldsymbol{\kappa}}\left[\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot\mathbf{x}}\prod_{i=1}^{3}\left(\frac{\mathrm{e}^{\mathrm{i}\kappa_i r_{i,2}}}{\mathrm{i}\kappa_i}-\frac{\mathrm{e}^{-\mathrm{i}\kappa_i r_{i,1}}}{\mathrm{i}\kappa_i}\right)\right]\mathrm{d}\mathbf{Z}(\boldsymbol{\kappa}),$$

$$= \int_{\boldsymbol{\kappa}}\left[\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}\cdot\mathbf{x}}\prod_{i=1}^{3}\operatorname{sinc}\left(\frac{\kappa_i\lambda_i}{2}\right)\right]\mathrm{d}\mathbf{Z}(\boldsymbol{\kappa}),$$

where integration has been carried out for $r_i = -\frac{1}{2}\lambda_i$ to $+\frac{1}{2}\lambda_i$. The above implies that a mean velocity can be easily obtained by applying a low pass filter:

$$G(\boldsymbol{\kappa}, V) = \prod_{i=1}^{3}\operatorname{sinc}\left(\frac{\kappa_i\lambda_i}{2}\right), \tag{2.11}$$

or, when omitting the product operator:

$$G(\boldsymbol{\kappa}, V) = \operatorname{sinc}\left(\frac{\kappa_x\lambda_x}{2}\right)\operatorname{sinc}\left(\frac{\kappa_y\lambda_y}{2}\right)\operatorname{sinc}\left(\frac{\kappa_z\lambda_z}{2}\right). \tag{2.12}$$

Since the low-pass filter is not dependent on $\mathbf{x}$, it can straightforwardly be inserted in the DFT matrix. It adds three parameters, λ_x, λ_y, and λ_z, which grants control over a cube-shaped volume[5]:

$$\begin{bmatrix} G_{1,1,1}\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & G_{2,1,1}\mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots \\ \mathrm{i}\kappa_x G_{1,1,1}\mathbf{C}_{1,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_x G_{2,1,1}\mathbf{C}_{2,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots \\ \mathrm{i}\kappa_y G_{1,1,1}\mathbf{C}_{1,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_y G_{2,1,1}\mathbf{C}_{2,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots \\ \mathrm{i}\kappa_z G_{1,1,1}\mathbf{C}_{1,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_0} & \mathrm{i}\kappa_z G_{2,1,1}\mathbf{C}_{2,1,1}^{(u)}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_0} & \cdots \end{bmatrix}\begin{bmatrix}\mathbf{n}_{1,1,1}\\ \mathbf{n}_{2,1,1}\\ \vdots\end{bmatrix} = \begin{bmatrix}\bar{u}(\mathbf{x}_0)\\0\\0\\0\\0\\0\end{bmatrix}.$$

The solution for a single-point velocity amplitude is then easily retrieved by setting $\lambda_x\lambda_y\lambda_z = 0$, such that $G(\boldsymbol{\kappa}, 0) = 1$.

2.4 Setting multiple constraints

Adding more constraints to the system is very straightforward by adding more rows to the linear system, essentially treating it as a block matrix:

$$\begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\\ \vdots\\ \mathbf{A}_k\end{bmatrix}\mathbf{n} = \begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \vdots\\ \mathbf{b}_k\end{bmatrix}, \tag{2.13}$$

This way, it is possible to generate multiple gusts in one domain, or specify a velocity jump over a certain distance.

For a one-dimensional problem with k maxima, a linear system with first derivatives can be set up according to

$$\begin{bmatrix} \mathrm{e}^{\mathrm{i}\kappa_1 x_1} & \mathrm{e}^{\mathrm{i}\kappa_2 x_1} & \cdots & \mathrm{e}^{\mathrm{i}\kappa_N x_1} \\ \mathrm{i}\kappa_1\mathrm{e}^{\mathrm{i}\kappa_1 x_1} & \mathrm{i}\kappa_2\mathrm{e}^{\mathrm{i}\kappa_2 x_1} & \cdots & \mathrm{i}\kappa_N\mathrm{e}^{\mathrm{i}\kappa_N x_1} \\ \mathrm{e}^{\mathrm{i}\kappa_1 x_2} & \mathrm{e}^{\mathrm{i}\kappa_2 x_2} & \cdots & \mathrm{e}^{\mathrm{i}\kappa_N x_2} \\ \mathrm{i}\kappa_1\mathrm{e}^{\mathrm{i}\kappa_1 x_2} & \mathrm{i}\kappa_2\mathrm{e}^{\mathrm{i}\kappa_2 x_2} & \cdots & \mathrm{i}\kappa_N\mathrm{e}^{\mathrm{i}\kappa_N x_2} \\ \vdots & \vdots & & \vdots \\ \mathrm{e}^{\mathrm{i}\kappa_1 x_k} & \mathrm{e}^{\mathrm{i}\kappa_2 x_k} & \cdots & \mathrm{e}^{\mathrm{i}\kappa_N x_k} \\ \mathrm{i}\kappa_1\mathrm{e}^{\mathrm{i}\kappa_1 x_k} & \mathrm{i}\kappa_2\mathrm{e}^{\mathrm{i}\kappa_2 x_k} & \cdots & \mathrm{i}\kappa_N\mathrm{e}^{\mathrm{i}\kappa_N x_k} \end{bmatrix}\begin{bmatrix}n_1\\n_2\\ \vdots\\ n_N\end{bmatrix} = \begin{bmatrix}u(x_1)\\ \dot{u}(x_1)\\ u(x_2)\\ \dot{u}(x_2)\\ \vdots\\ u(x_k)\\ \dot{u}(x_k)\end{bmatrix}.$$

The linear system for a three-dimensional problem with k horizontal maxima looks a little more complex, but works in the same way:

[5] Of course, different volume shapes can be addressed by using a different expression for the low-pass filter.

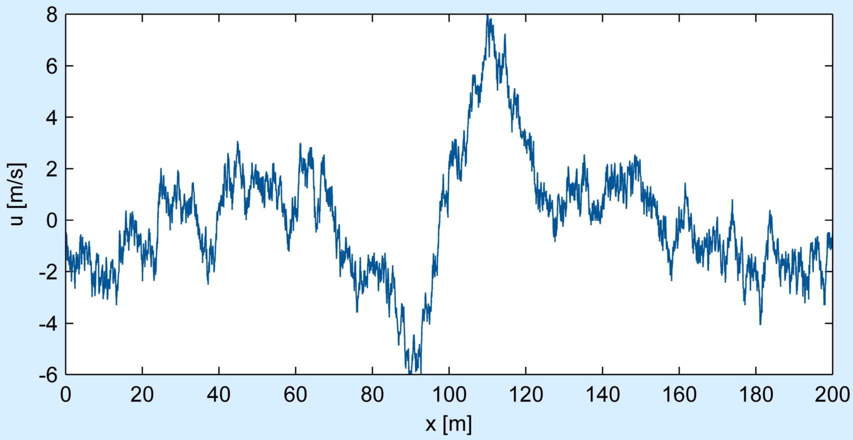
$$
\begin{bmatrix}
G_{1,1,1}\mathbf{C}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_1} & G_{2,1,1}\mathbf{C}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_1} & \cdots \\
i\kappa_x G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_1} & i\kappa_x G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_1} & \cdots \\
i\kappa_y G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_1} & i\kappa_y G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_1} & \cdots \\
i\kappa_z G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_1} & i\kappa_z G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_1} & \cdots \\
G_{1,1,1}\mathbf{C}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_2} & G_{2,1,1}\mathbf{C}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_2} & \cdots \\
i\kappa_x G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_2} & i\kappa_x G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_2} & \cdots \\
i\kappa_y G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_2} & i\kappa_y G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_2} & \cdots \\
i\kappa_z G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_2} & i\kappa_z G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_2} & \cdots \\
\vdots & \vdots & \\
G_{1,1,1}\mathbf{C}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_k} & G_{2,1,1}\mathbf{C}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_k} & \cdots \\
i\kappa_x G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_k} & i\kappa_x G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_k} & \cdots \\
i\kappa_y G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_k} & i\kappa_y G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_k} & \cdots \\
i\kappa_z G_{1,1,1}\mathbf{C}^{(u)}_{1,1,1}e^{i\boldsymbol{\kappa}_{1,1,1}\cdot\mathbf{x}_k} & i\kappa_z G_{2,1,1}\mathbf{C}^{(u)}_{2,1,1}e^{i\boldsymbol{\kappa}_{2,1,1}\cdot\mathbf{x}_k} & \cdots
\end{bmatrix}
\begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \end{bmatrix}
=
\begin{bmatrix} \bar{u}(\mathbf{x}_1) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \bar{u}(\mathbf{x}_2) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \bar{u}(\mathbf{x}_k) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.
$$

Example 4 *One-dimensional velocity jump over a distance*

Given the conditions from example 1. Placing a velocity jump from $u(x_1 = 90\text{ m}) = -6$ m/s to $u(x_2 = 110\text{ m}) = 8$ m/s requires a linear system in the shape of

$$
\begin{bmatrix}
e^{i\kappa_1\cdot 90} & e^{i\kappa_2\cdot 90} & \cdots & e^{i\kappa_N\cdot 90} \\
i\kappa_1 e^{i\kappa_1\cdot 90} & i\kappa_2 e^{i\kappa_2\cdot 90} & \cdots & i\kappa_N e^{i\kappa_N\cdot 90} \\
e^{i\kappa_1\cdot 110} & e^{i\kappa_2\cdot 110} & \cdots & e^{i\kappa_N\cdot 110} \\
i\kappa_1 e^{i\kappa_1\cdot 110} & i\kappa_2 e^{i\kappa_2\cdot 110} & \cdots & i\kappa_N e^{i\kappa_N\cdot 110}
\end{bmatrix}
\begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_N \end{bmatrix}
=
\begin{bmatrix} -6 \\ 0 \\ 8 \\ 0 \end{bmatrix}.
$$

Again, an inverse Fourier transform leads to the following time series:



Example 5 *A cluster of three single-point velocity maxima*

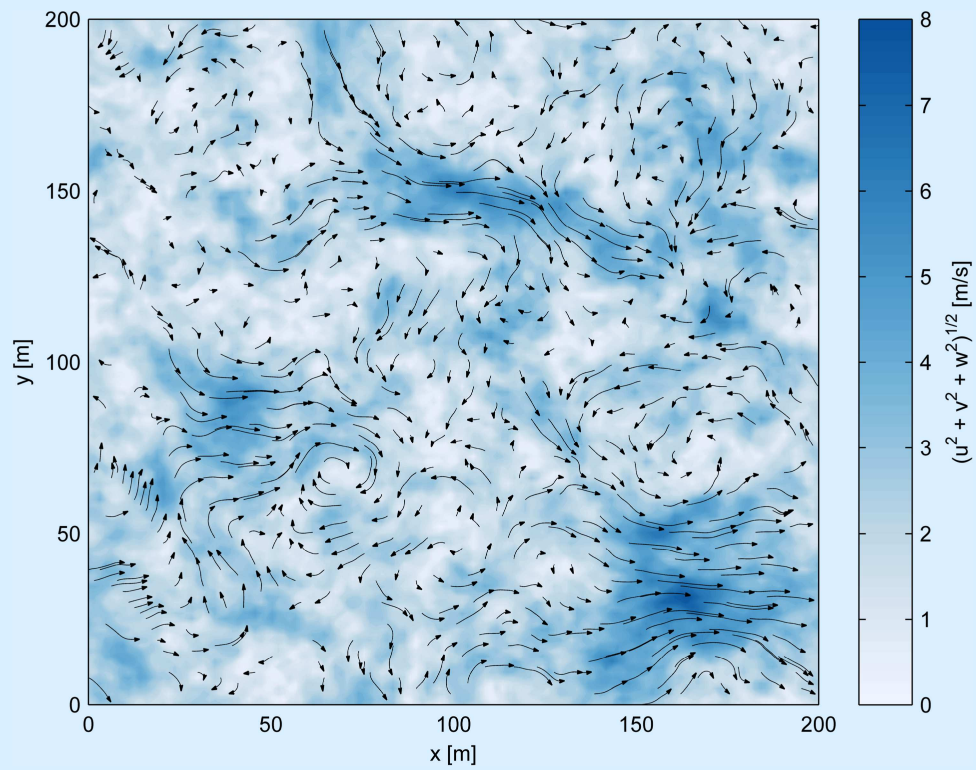
Given is a cubic domain with sides 200x200x200 m. Inside are three individual velocity maxima:

$$\mathbf{u}\left(\begin{bmatrix}40\\90\\100\end{bmatrix}\right)=\begin{bmatrix}6\\0\\0\end{bmatrix}\ \mathrm{m/s},\qquad \mathbf{u}\left(\begin{bmatrix}100\\150\\100\end{bmatrix}\right)=\begin{bmatrix}7\\0\\0\end{bmatrix}\ \mathrm{m/s},\qquad \mathbf{u}\left(\begin{bmatrix}160\\30\\100\end{bmatrix}\right)=\begin{bmatrix}8\\0\\0\end{bmatrix}\ \mathrm{m/s}.$$

In that case, the linear system $\mathbf{An}=\mathbf{b}$ becomes

$$\begin{bmatrix}
\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_x\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_y\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_z\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[40,90,100]^{\mathrm{T}}} & \cdots\\
\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_x\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_y\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_z\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[100,150,100]^{\mathrm{T}}} & \cdots\\
\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_x\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_y\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \cdots\\
\mathrm{i}\kappa_z\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[160,30,100]^{\mathrm{T}}} & \cdots
\end{bmatrix}
\begin{bmatrix}\mathbf{n}_{1,1,1}\\ \mathbf{n}_{2,1,1}\\ \vdots\end{bmatrix}
=\begin{bmatrix}6\\0\\0\\0\\0\\0\\7\\0\\0\\0\\0\\0\\8\\0\\0\\0\\0\\0\end{bmatrix},$$

yielding the field shown below:



Example 6 *Velocity jump between two planes*

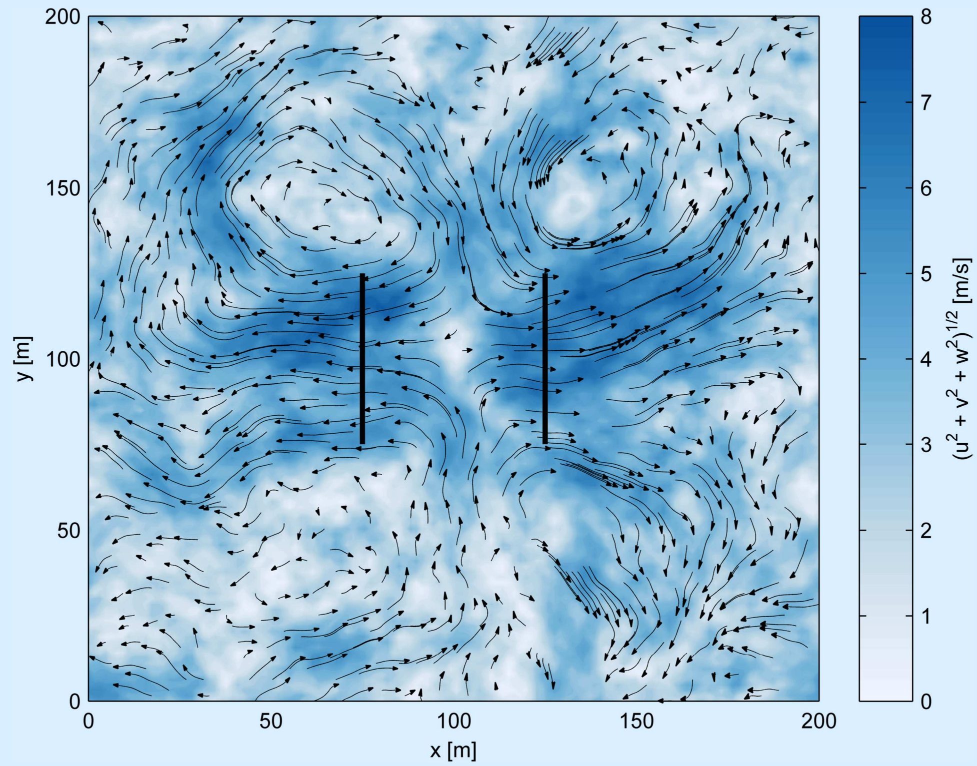
Given are two planes located at $\mathbf{x}_1 = [75, 100, 100]^{\mathrm{T}}$ and $\mathbf{x}_2 = [125, 100, 100]^{\mathrm{T}}$, with each a width and height of $\lambda_y = 50$ m and $\lambda_z = 50$ m, respectively. Now, specify a velocity jump where the wind speed increases from $\bar{\mathbf{u}}_1 = -5$ m/s to $\bar{\mathbf{u}}_1 = 5$ m/s, averaged over the planes. Then, define the following filter shape:

$$G(\boldsymbol{\kappa}) = \mathrm{sinc}\left(\frac{50\kappa_y}{2}\right)\mathrm{sinc}\left(\frac{50\kappa_z}{2}\right),$$

and set the linear system up as follows:

$$\begin{bmatrix} G(\boldsymbol{\kappa})\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[75,100,100]^{\mathrm{T}}} & G(\boldsymbol{\kappa})\mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[75,100,100]^{\mathrm{T}}} & \cdots \\ G(\boldsymbol{\kappa})\mathbf{C}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[125,100,100]^{\mathrm{T}}} & G(\boldsymbol{\kappa})\mathbf{C}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_x G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_y G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \cdots \\ \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{1,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{1,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \mathrm{i}\kappa_z G(\boldsymbol{\kappa})\mathbf{C}^{(u)}_{2,1,1}\mathrm{e}^{\mathrm{i}\boldsymbol{\kappa}_{2,1,1}\cdot[125,100,100]^{\mathrm{T}}} & \cdots \end{bmatrix} \begin{bmatrix} \mathbf{n}_{1,1,1} \\ \mathbf{n}_{2,1,1} \\ \vdots \end{bmatrix} = \begin{bmatrix} -5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

This yields a velocity field, of which a slice at $z = 100$ m is shown below.



3 STATISTICS OF SPATIAL GUSTS

Generating gusts in a spatial domain leads to an interesting problem when quantifying the statistics. First and foremost, it requires a three-dimensional version of Rice's formula. Under the assumption of Gaussian turbulence, the gust probability can be expressed by the second-order spectral moment, together with a relevant length scale.

3.1 Rice's formula

Let a one-dimensional gust be an event in stationary, homogeneous, Gaussian turbulence, where $u(x) \sim \mathrm{N}(0, \sigma^2)$, $x \in \mathbb{R}$. The expected number of times $u(x)$ takes the value A over a length L_x is then given by *Rice's formula*[6]:

$$\frac{\mathrm{E}[N_{u=A}]}{L_x} = \int_{-\infty}^{\infty} |\dot{u}| f(A, \dot{u}) \mathrm{d}\dot{u}, \tag{3.1}$$

where $\dot{u} = \mathrm{d}u/\mathrm{d}x$ and $f(u, \dot{u})$ is the joint probability density of u and its first derivative,

$$\begin{bmatrix} u \\ \dot{u} \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma^2 & 0 \\ 0 & r \end{bmatrix}\right), \tag{3.2}$$

where

$$r = \mathrm{E}\left[\left(\frac{\partial u}{dx}\right)^2\right], \tag{3.3}$$

is the variance of the first derivative of u. The number of excursions above a threshold A can be obtained simply by only taking upcrossings into account ($\dot{u} > 0$). This yields

$$\frac{\mathrm{E}[N_{u>A}]}{L_x} = \int_{0}^{\infty} \frac{\dot{u}}{2\pi\sigma\sqrt{r}} \mathrm{e}^{-\frac{\dot{u}^2}{2r} - \frac{A^2}{2\sigma^2}} \mathrm{d}\dot{u}, \tag{3.4}$$

which can be solved by making the substitution $v = \dot{u}^2/(2r)$, $\mathrm{d}v = \dot{u}/r \,\mathrm{d}\dot{u}$, leading to

$$\frac{\mathrm{E}[N_{u>A}]}{L_x} = \frac{\sqrt{r}}{2\pi\sigma} \mathrm{e}^{-\frac{A^2}{2\sigma^2}}. \tag{3.5}$$

To illustrate, figure 1 shows a comparison between the level excursions counted from synthetic turbulence, using an isotropic von Kármán spectrum, and equation (3.5).

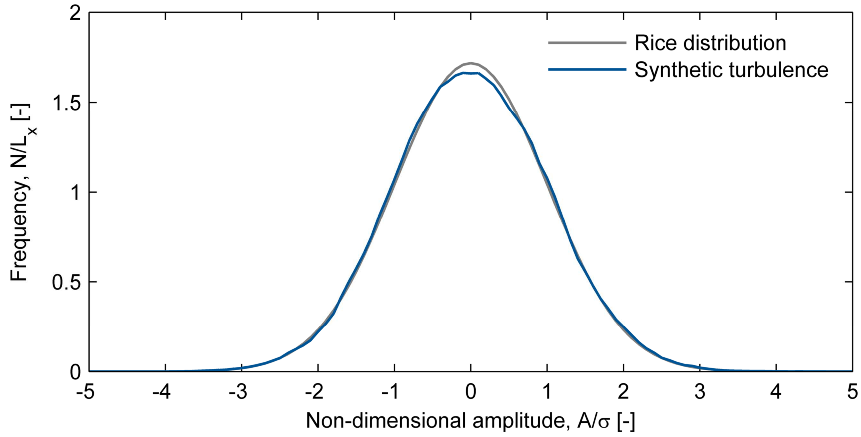


Figure 1: Level excursions counted in synthetic 1D isotropic turbulence, compared to the Rice distribution.

[6] Rice, S. O. (1944). "Mathematical Analysis of Random Noise." *Bell System Technology Journal* 23 (3): 282–332.

3.2 Level excursions in $\mathbb{R}^3$

For higher-order dimensions, however, the problem quickly becomes much more complex as the boundaries of an upcrossing are harder to define. This problem has been tackled by Adler and Taylor[7] through the use of the Euler characteristic, $\varphi(x)$, making it a matter of topology. The Euler characteristic counts the number of connected components in a field and subtracting the amount of holes. If, for an excursion set $u(\mathbf{x}) \geq A$, the amplitude threshold is high enough, the holes disappear and what is left are the number of regions where the A is exceeded (see figure 2):

$$\mathrm{P}\left(\sup_{\mathbf{x}\in B} u(\mathbf{x}) \geq A\right) \approx \mathrm{E}[\varphi(Z_A)]. \tag{3.6}$$

where the approximation is off by an error

$$\left|\mathrm{P}\left(\sup_{\mathbf{x}\in B} u(\mathbf{x}) \geq A\right) - \mathrm{E}[\varphi(Z_A)]\right| < O\left(e^{-\alpha\frac{A^2}{2\sigma^2}}\right). \tag{3.7}$$

for some $\alpha > 1$. A generalized expression for the expected Euler characteristic exists for N dimensions and is given by Adler and Taylor in the book *Random Fields and Geometry* (theorem 11.7.2), preceded by about 300 pages of background:

$$\mathrm{E}[\varphi(Z_A)] = \mathrm{e}^{-\frac{A^2}{2\sigma^2}} \sum_{k=0}^{N} \sum_{J\in O_k} \frac{|J|\sqrt{|\mathbf{R}_J|}}{(2\pi)^{(k+1)/2}\,\sigma^k} H_{k-1}\left(\frac{A}{\sigma}\right), \tag{3.8}$$

where

$$H_{\mathrm{n}}(x) = \begin{cases} n! \displaystyle\sum_{j=0}^{\lfloor n/2 \rfloor} \frac{(-1)^j x^{n-2j}}{j!\,(n-2j)!\,2j}, & n \geq 0, \\ \sqrt{2\pi}\Psi(x)e^{x^2/2}, & n = -1; \end{cases} \tag{3.9}$$

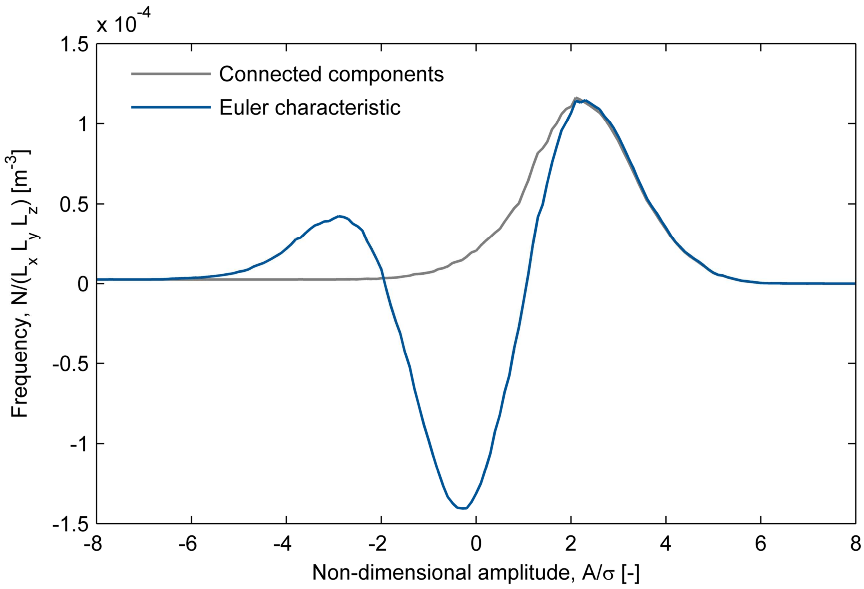


Figure 2: Connected components in the excursion set $\boldsymbol{u}(\mathbf{x}) \geq \boldsymbol{A}$, compared to the Euler characteristic. The field $\boldsymbol{u}(\mathbf{x})$ is the horizontal velocity component of isotropic turbulence, smoothed with a 20x20x20 m low-pass box filter.

[7] Adler, R. J., and J. E. Taylor (2007). *Random Fields and Geometry*. New York, NY: Springer. doi:10.1007/978-0-387-48116-6.

is the nth Hermite polynomial for $x \in \mathbb{R}$ with $\Psi(x)$ being the tail of the Gaussian distribution[8], e.g.

$$\Psi(x) = 1 - \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-v^2/2} \mathrm{d}v. \tag{3.10}$$

Moreover, $|J|$ denotes the Lebesgue measure of J and $|\mathbf{R}_J|$ is the determinant of the matrix of second-order spectral moments:

$$\mathbf{R}_J = \int_{\boldsymbol{\kappa}} \kappa_i \kappa_j [G(\boldsymbol{\kappa}, V)]^2 \boldsymbol{\Phi}_{uu}(\boldsymbol{\kappa}) \mathrm{d}\boldsymbol{\kappa} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}. \tag{3.11}$$

Furthermore, $\sum_{J \in O_k}$ denotes summing over the $\binom{N}{k}$ k-dimensional faces of B which contain the origin. The variance of the filtered spectrum can be obtained through

$$\sigma^2 = \int_{\boldsymbol{\kappa}} [G(\boldsymbol{\kappa}, V)]^2 \boldsymbol{\Phi}_{uu}(\boldsymbol{\kappa}) \mathrm{d}\boldsymbol{\kappa}. \tag{3.12}$$

When B is a three-dimensional rectangle[9] with sides $L_x \times L_y \times L_z$, O_0 contains the single vertex at the origin; O_1 three ribs along the x-, y-, and z-axes; O_2 the xy-, xz-, and yz-planes; and O_3 the volume of the rectangle. Then, equation (3.8) can be written out as

$$\mathrm{E}[\varphi(Z_A)] = \Psi\left(\frac{A}{\sigma}\right) + \mathrm{e}^{-\frac{A^2}{2\sigma^2}} \left[c_1 + c_2 \frac{A}{\sigma} + c_3 \left(\frac{A^2}{\sigma^2} - 1\right)\right], \tag{3.13}$$

where

$$c_1 = \frac{L_x \sqrt{r_{xx}} + L_y \sqrt{r_{yy}} + L_z \sqrt{r_{zz}}}{2\pi\sigma},$$

$$c_2 = \frac{L_x L_y \sqrt{\begin{vmatrix} r_{xx} & r_{xy} \\ r_{yx} & r_{yy} \end{vmatrix}} + L_x L_z \sqrt{\begin{vmatrix} r_{xx} & r_{xz} \\ r_{zx} & r_{zz} \end{vmatrix}} + L_y L_z \sqrt{\begin{vmatrix} r_{yy} & r_{yz} \\ r_{zy} & r_{zz} \end{vmatrix}}}{(2\pi)^{\frac{3}{2}} \sigma^2},$$

$$c_3 = \frac{L_x L_y L_z}{(2\pi)^2 \sigma^3} \sqrt{\begin{vmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{vmatrix}}.$$

In most (if not all) cases, it is true that $L_x \gg L_y, L_z$. For large enough amplitudes ($\Psi(A/\sigma) \approx 0$), equation (3.13) can then be approximated as

$$\mathrm{E}[\varphi(Z_A)] \approx L_x L_y L_z \frac{\sqrt{|\mathbf{R}|}}{4\pi^2 \sigma^3} \mathrm{e}^{-\frac{A^2}{2\sigma^2}} \left(\frac{A^2}{\sigma^2} - 1\right), \tag{3.14}$$

where $\mathbf{R}$ is computed as in (3.11). The above is the same result mentioned in Adler (1976)[10] and Adler and Husofer (1976)[11], and is caused by neglecting the 0th- through 2nd-dimensional faces of the domain (i.e., the edges, vertices, and surfaces). The linear dependence on the domain size is now much easier to understand (twice the domain size equals twice as many excursions). However, if the gust scale is relatively large compared to the domain size, it is advisable to rely on

[8] The role of $\Psi(A/\sigma)$ becomes clear in the limit $A \to -\infty$, in which case the entire field is one local maximum and thus $\mathrm{E}[\varphi(Z_A)] \to 1$.

[9] If B is a one-dimensional domain of length L_x, equation (3.8) approaches (3.5):

$$\mathrm{E}[\varphi(Z_A)] = \Psi\left(\frac{A}{\sigma}\right) + \frac{\sqrt{r_{xx}}}{2\pi\sigma} \mathrm{e}^{-\frac{A^2}{2\sigma^2}}.$$

[10] Adler, R. J. (1976). "On Generalising the Notion of Upcrossings to Random Fields." *Advances in Applied Probability* 8 (4): 789. doi:10.2307/1425934.

[11] Adler, R. J., and A. M. Hasofer (1976). "Level Crossings for Random Fields." *The Annals of Probability* 4 (1): 1–12. doi:10.1214/aop/1176996176.

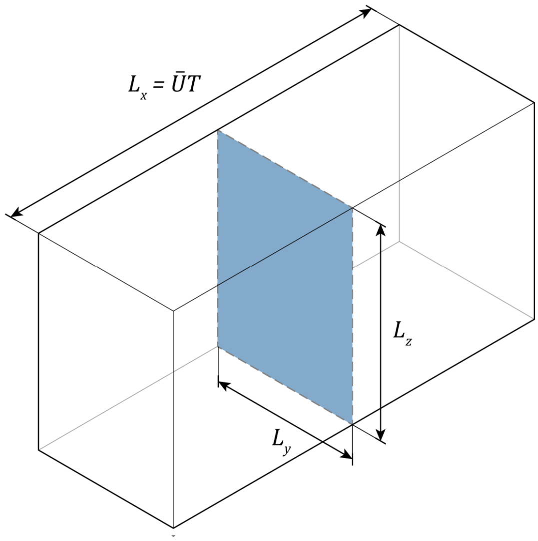


Figure 3: Wind advecting through an area with sides $\boldsymbol{L_y} \times \boldsymbol{L_z}$.

the full expression given by equation (3.13) to avoid any errors (the full expression does not contain any additional parameters).

3.3 A special note on low-pass filters

Low-pass filters were introduced in section 2.3 as a means to set constraints on volumes. However, they serve another important purpose when it comes to assessing the probability of gusts. When evaluating second order spectral moments for turbulent velocity fluctuations for which the spectrum scales according to $E(\kappa) \propto \kappa^{-5/3}$, the integral in equation (3.11) is not finite unless a low-pass filter is included. This is because the second-order fluctuations (i.e., $\partial u/\partial x$) display the *Richardson effect*.

For example, an interesting problem in geography is the *coastline paradox*, which shows that the length of a coastline is not well-defined, but instead is dependent on the ruler that is used to measure it. A short ruler will take into account more of the small-scale features of a coast's geometry than a long ruler, and would therefore return a longer length. In his 1967 paper, *How Long Is the Coast of Britain?*[12], Benoît Mandelbrot argues that, since geography is uninterested in the details, one should rather settle for a minimum feature size to define what features are relevant.

Since turbulent velocity fluctuations display a high degree of self-similarity across the inertial sub-range, the same problem exists for wind gusts. The use of low-pass filters in the constraint set, shown in section 2.3, and in equation (3.11) therefore act as a "ruler" that defines which scales are relevant. In meteorology, the instantaneous wind speed and peak gusts are usually recorded as the average over 2–3 seconds. The reason for that is that those few seconds correspond to a *wind run*—defined as the gust duration times the average velocity—of about 100 m. This time scale is chosen such that an average structure will likely experience the full gust load[13]. Of course, different scales can be thought of, depending on the structure of interest.

3.4 Gust probability

The occurrence probability of extreme wind gusts can be derived on basis of the *Poisson limit*. For high enough amplitudes, a binomial distribution with k successes can be approximated by

[12] Mandelbrot, B. B. (1967). "How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension", *Science* 156 (3775): 636–638. doi:10.1126/science.156.3775.636.
[13] Beljaars, A. C. M. (1987). "The Measurement of Gustiness at Routine Wind Stations". Technical Report WR 87-11. De Bilt, The Netherlands.

$$\mathrm{P}(N_{u>A} = k) \approx \frac{(\mathrm{E}[N_{u>A}])^k}{k!} \mathrm{e}^{-\mathrm{E}[N_{u>A}]}, \tag{3.15}$$

where $\mathrm{E}[N_{u>A}] \approx \mathrm{E}[\varphi(Z_A)]$ denotes the expected frequency of an event for which the horizontal velocity, u, is higher than a threshold A. Under the assumption of frozen turbulence, the domain length is given by a mean wind speed and a certain time period (i.e., $L_x = \bar{U}T$). Equation (3.13) or (3.14) can then be used to determine a return period:

$$T_A \approx \frac{T}{\mathrm{E}[\varphi(Z_A)]}. \tag{3.16}$$

In practice, the mean wind speed and variance are not constant, but instead given by a joint probability density function, $f(\bar{U}, \sigma)$. The expected number of gusts exceeding A then follows from

$$\mathrm{E}[N_{u>A}] \approx \int_0^\infty \int_0^\infty \mathrm{E}[\varphi(Z_A)]\, f(\bar{U}, \sigma)\, \mathrm{d}\sigma\, \mathrm{d}\bar{U}. \tag{3.17}$$

The above is visualized by example 7.

Example 7 *Return levels for wind speeds in a domain using the IEC NTM*

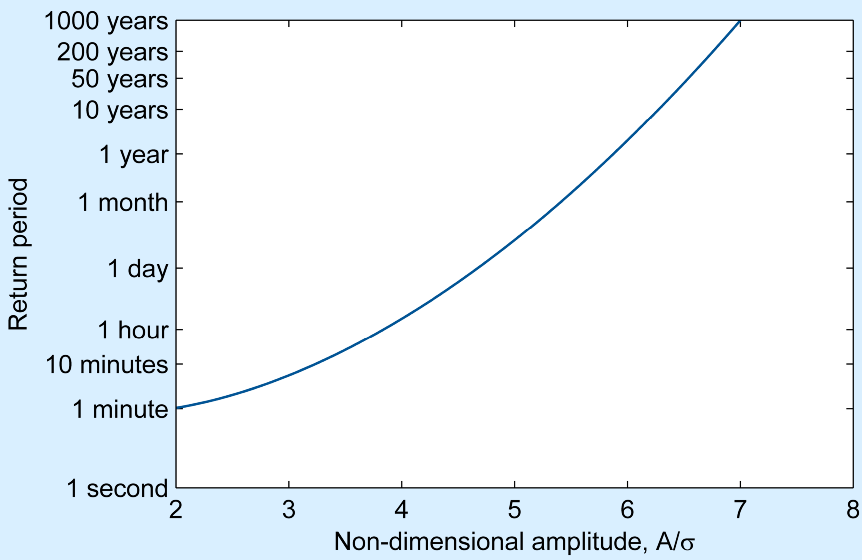
The normal turbulence model (NTM) in the IEC 61400-1 standards prescribes a wind speed and turbulence distribution by

$$f(V_{\mathrm{hub}}) = 1 - \mathrm{e}^{-\pi\left(\frac{V_{\mathrm{hub}}}{2V_{\mathrm{ave}}}\right)^2},$$
$$\sigma_1 = I_{\mathrm{ref}}(0.75V_{\mathrm{hub}} + 5.6),$$

with $V_{\mathrm{ave}} = 50$ m/s and $I_{\mathrm{ref}} = 0.16$ for a class 1A site. To evaluate the second-order spectral moments, the Mann model is used as defined in IEC 61400-1 Annex B. The expected number of wind gusts in the NTM over a sample length can then be found by calculating the expected Euler characteristic from equation (3.13):

$$\mathrm{E}[N_{u>A}] \approx \int_0^\infty \mathrm{E}[\varphi(Z_A)]\, f(V_{\mathrm{hub}})\, \mathrm{d}V_{\mathrm{hub}}.$$

Through a frontal surface of 100x100 m (e.g., see figure 3), the following return levels can be expected for Gaussian gusts, where the velocity is smoothed using a cubic kernel with sides $3\bar{U} \times 5 \times 5$ m (i.e., a longitudinal scale of 3 seconds).



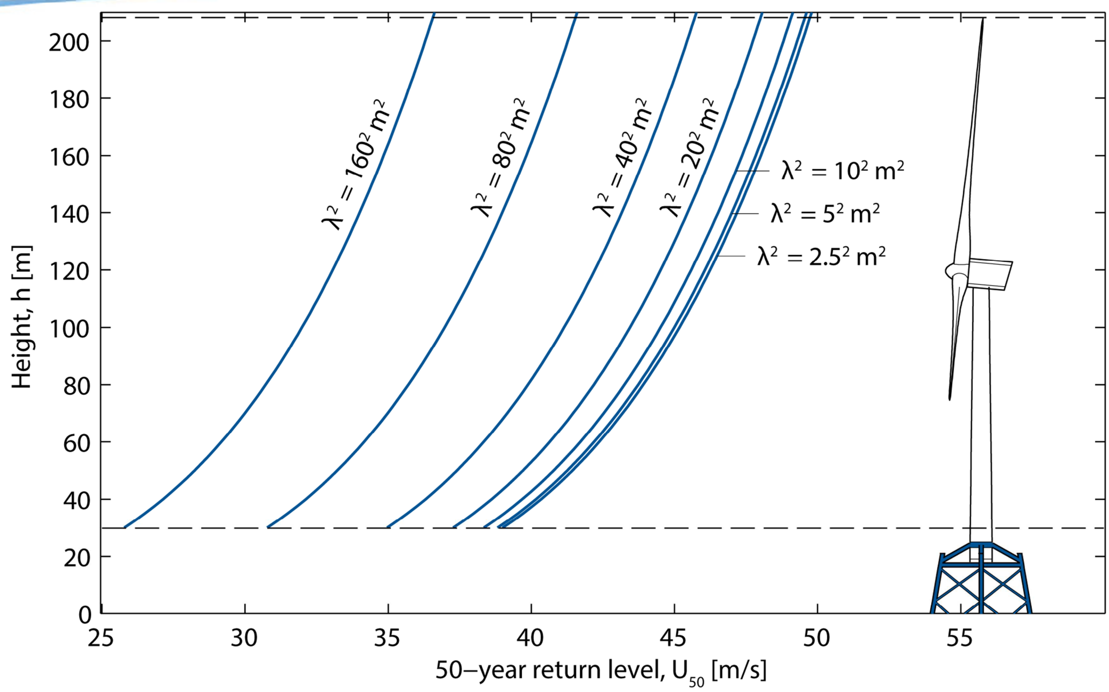


Figure 4: Absolute 50-year gust amplitudes, averaged over a rectangular box with size $3\bar{U} \times \lambda \times \lambda$ (i.e., a 3-second gust), through a square 178.3x178.3 m area (indicated with dashed lines). Turbulence properties are according to an IEC class 1A wind regime, using the Mann turbulence spectrum (see IEC 61400-1 Annex B). Added for scale is the DTU 10 MW reference turbine.

To conclude this chapter, the 50-year return levels will be presented for a class 1A wind regime, over a square frontal area with a size of 178.3x178.3 m (exactly fitting the rotor of the DTU 10 MW reference turbine). The methodology is similar to what is shown in example 7, but the expected level excursions are summed with respect to the absolute amplitude, $\bar{U}(z) + A$. The process is repeated for different heights along the wind shear profile (turbulence properties are fixed for a 119 m hub height) and for various filter sizes. The results, shown in figure 4, clearly show how the amplitude decreases with increasing area of effect. This underlines that coherent gusts (i.e., gusts with a uniform velocity over the rotor plane) are very unphysical events for large rotor diameters. Moreover, it also shows that velocity amplitudes taken from single-point measurements cannot be used to as input for coherent gust models.

4 VALIDATION

The previous chapters introduced low-pass filters as an important aspect of gust statistics. This chapter discusses what happens when gusts are extracted from high-frequency measurements, based on the filtered amplitude, and how this compares with theory.

4.1 Data source

The Offshore Windfarm Egmond aan Zee (OWEZ) is the first offshore wind farm ever to be constructed in the Netherlands, and is located about 10–18 km off the coast of Egmond aan Zee. The turbines were installed starting in the summer of 2006 and the park was officially opened in March 2007. Prior to construction, in 2005, a met mast was installed at 52° 36' 22.9" N, 4° 23' 22.7" E to gather wind and wave data from the site. The mast is a 116 m high triangular lattice tower placed on a monopile foundation. Measurement stations are located at 21, 70 and 116 m, and are each equipped with three cup anemometers, three wind vanes, and one sonic anemometer in the configuration shown in figure 5. Meteorological data spanning July 2005 to December 2010, subdivided in 10-minute intervals, is freely available online[14]. Moreover, raw high-frequency (4 Hz) wind data has been made available by the Energy Research Centre of the Netherlands (ECN) from May 2007 until December 2008.

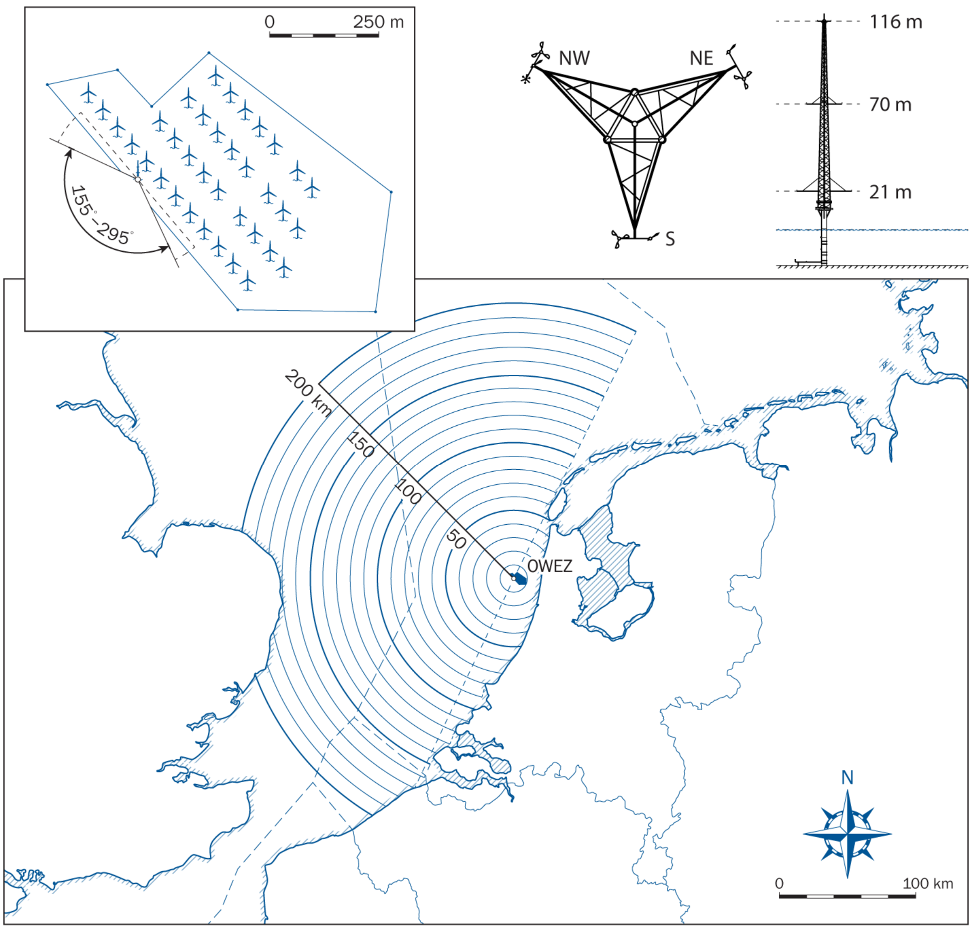
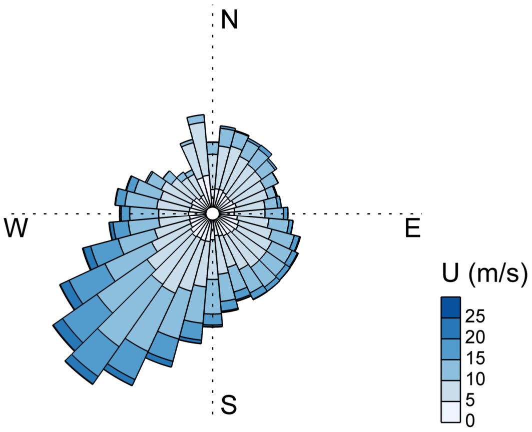
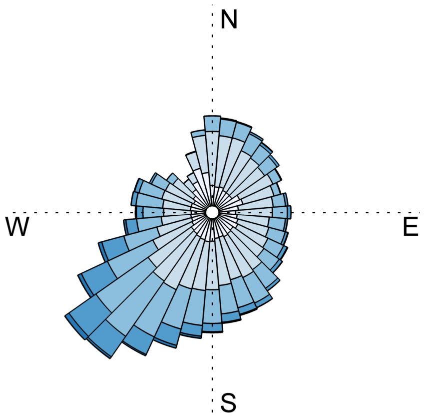


Figure 5: Location of the OWEZ wind farm in the Dutch North Sea. Added to the top-left is a detailed view of the park layout together with the directions of undistorted flow, as seen from the met mast.

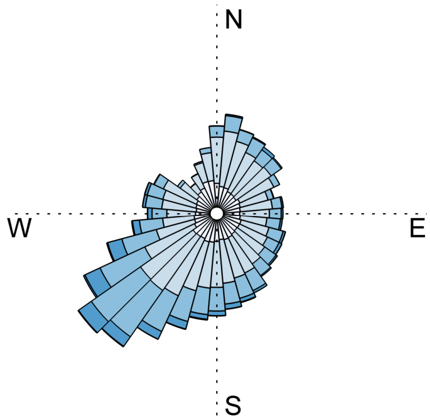
[14] NoordzeeWind (2007). "Reports & Data". URL: www.noordzeewind.nl/en/knowledge/reportsdata/ (Accessed 26 June 2013).



a) Wind rose at 116 m.



b) Wind rose at 70 m.



c) Wind rose at 21 m.

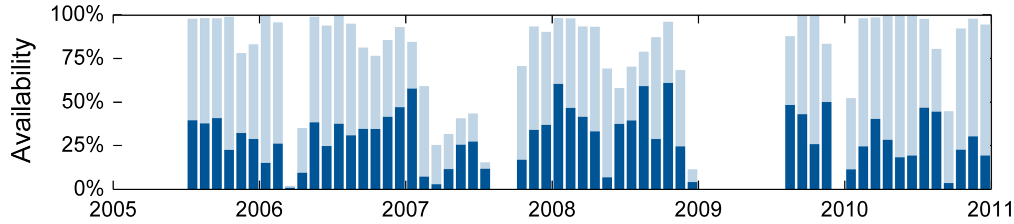
Figure 6: Wind roses for the three sensor heights at the OWEZ met mast.

Frequency plots of the mean wind speed and wind direction are shown in figure 6a through c. Approximate Weibull scale and shape parameters for the annual 10-minute mean wind speeds are respectively 9.1 and 2.4 for $h = 21$ m; 10.3 and 2.3 for $h = 70$ m; and 10.8 and 2.2 for $h = 116$ m. The prevailing wind direction is south west; typical for the Dutch coast.

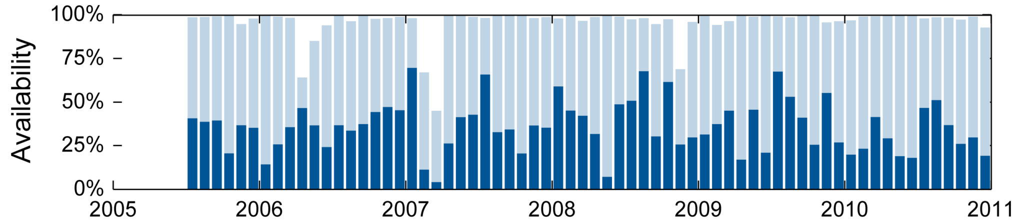
Since the park was already in operation during the time of the measurements, the met mast can encounter wake effects for wind directions from 315 to 135°[15]. However, measurements between 295 and 155° already show a significantly higher turbulence intensity[16]. Moreover, easterly winds will be affected by the roughness onshore. In that sense, excluding the data from $25 \leq \theta \leq 205°$, as shown in figure 5, will ensure offshore conditions with a fetch of at least 170 km. In addition, only the sonic anemometers placed on the northwest booms can deliver good high-frequency measurements. At this position, the tower wake invalidates the wind directions from roughly 90 to 150°. All these effects cause a large portion of the data to be too unreliable for a good site assessment.

These constraints mean that the measurements should be limited to $205 \leq \theta \leq 295°$, which amounts to 40% of the wind directions counted from 2006 to 2010 (see figure 6). A more detailed overview of the usable sonic anemometer records is given in figure 7a through c. Notable is the fact that the data set from $h = 116$ m is very incomplete during some periods, which means some seasonal bias can be expected.

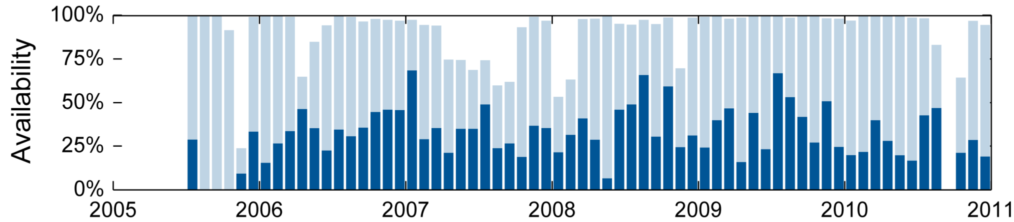
[15] Kouwenhoven, H. J. (2007). “Manual Data Files Meteo Mast NoordzeeWind”. Technical Report NZW-16-S-4-R03. NoordzeeWind, IJmuiden, The Netherlands.
[16] Wagenaar, J. W., and P. J. Eecen (2010). “3D Turbulence at the Offshore Wind Farm Egmond Aan Zee”. Technical Report ECN-E--10-075. Energy Research Centre of the Netherlands, Petten, The Netherlands.



a) Availability at 116 m (▭), compared to the availability in the range $205 \leq \theta \leq 295°$ (▬).



b) Availability at 70 m (▭), compared to the availability in the range $205 \leq \theta \leq 295°$ (▬).



c) Availability at 21 m (▭), compared to the availability in the range $205 \leq \theta \leq 295°$ (▬).

Figure 7: Availability of sonic data at the OWEZ met mast.

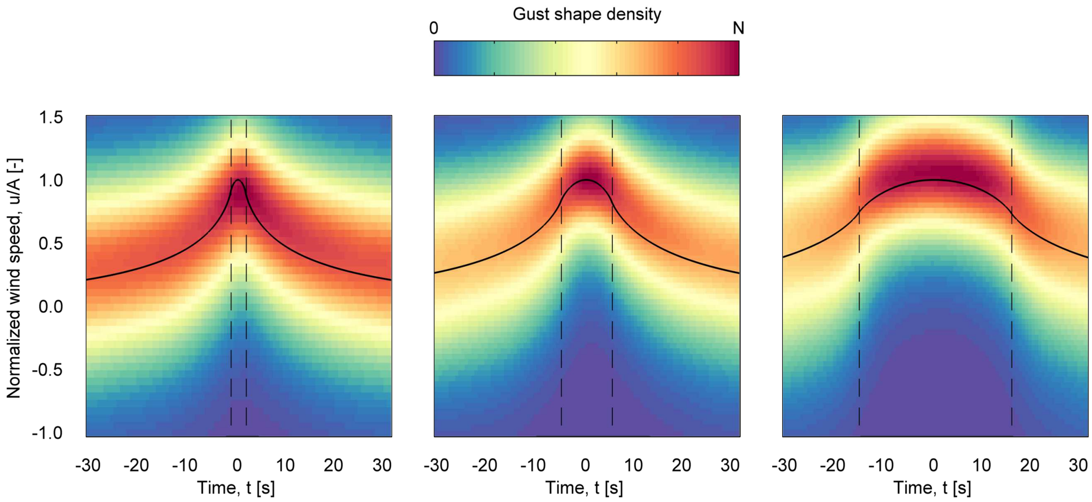
4.2 Gust shapes

The mean shape of extreme gusts can be shown to follow the turbulence autocorrelation function[17]. From the discussion in section 2.3, it is expected that the peaks from a filtered turbulence signal should follow the filtered autocorrelation function. To validate this, gusts exceeding an amplitude of 3σ (after filtering) have been extracted from the sonic anemometer. This process is repeated with different low-pass box filters having a window size of 3, 10, and 30 seconds. Gusts are centered and normalized with respect to the amplitude. A smooth autocorrelation function is obtained by fitting an exponential function to the unfiltered signal. At first, the gust shapes contained a dominant harmonic with a period of 2.5 seconds, coinciding with the tower's natural frequency[18]. Therefore, the wind signal was first filtered with a third-order Butterworth band-stop filter ranging from 0.35 to 0.45 Hz.

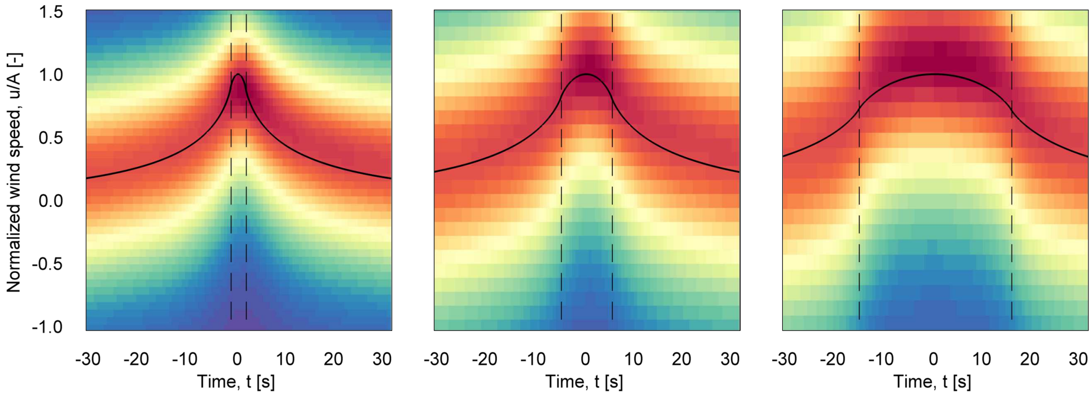
Figure 8a through g show that, for 7 different mean wind speed bins, the results match well with what is expected. For increasing wind speeds, the measured gusts tend to become more narrow (although the effect is most dominant in the absence of filters), which can be attributed to a higher rate of advection. The quality of the fit is somewhat lower for the 30 s window. This might be because turbulence spectra often do not agree well at low frequencies, depending on the stability of the atmospheric boundary layer.

---

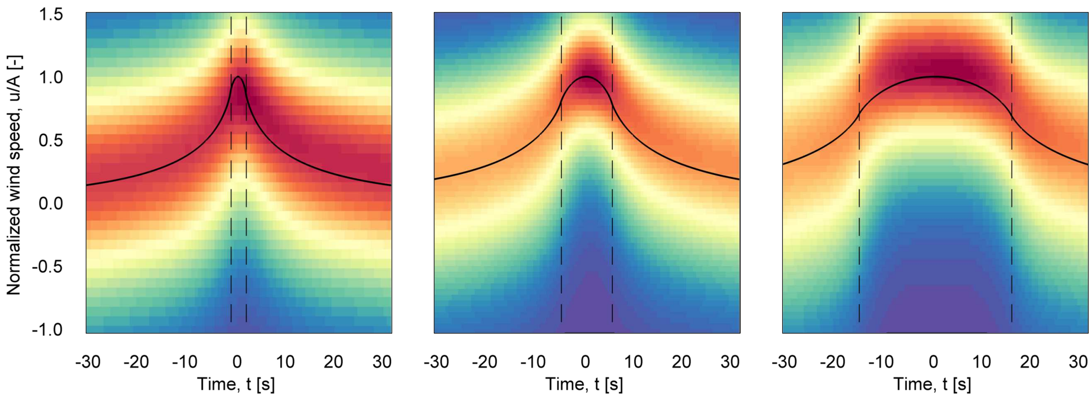
[17] Bierbooms, W. A. A. M., J. B. Dragt, and H. Cleijne (1999). "Verification of the Mean Shape of Extreme Gusts." *Wind Energy* 2 (3): 137–150. doi:10.1002/(SICI)1099-1824(199907/09)2:3<137::AID-WE24>3.0.CO;2-W.
[18] Eecen, P. J., and E. Branlard (2008). "The OWEZ Meteorological Mast". ECN-E--08-067. Petten, The Netherlands.



a) $0 < \bar{U} \leq 4$ m/s (N = 1194, 776, 434).

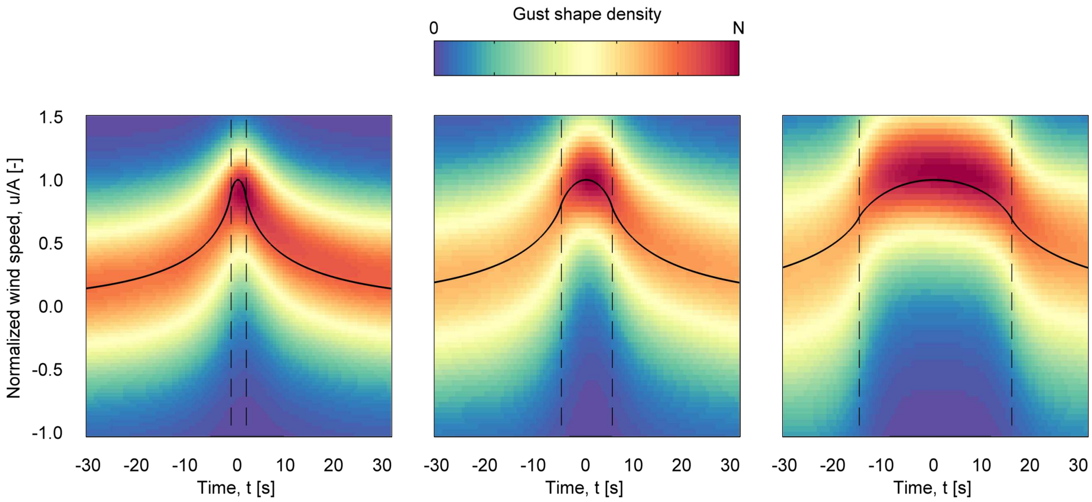


b) $4 < \bar{U} \leq 8$ m/s (N = 3101, 1861, 1060).

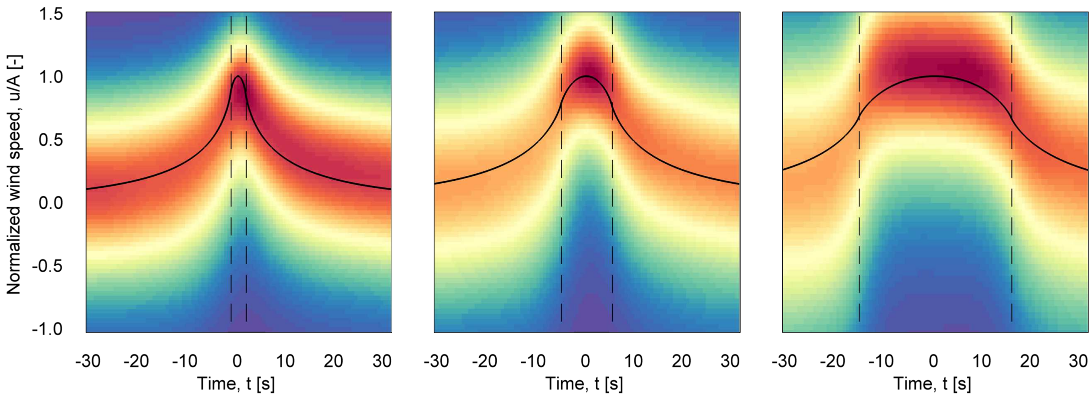


c) $8 < \bar{U} \leq 12$ m/s (N = 4272, 2488, 1060).

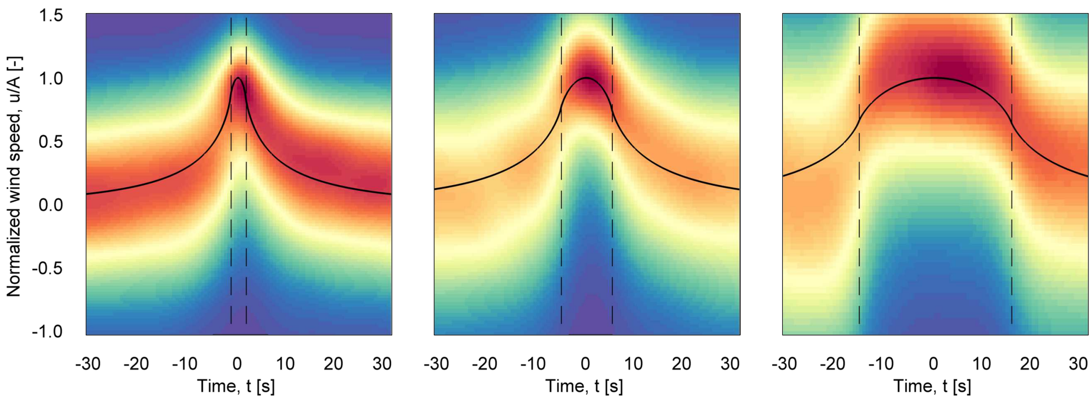
Figure 8: Mean gust shapes extracted from the OWEZ met mast compared to the filtered autocorrelation function (—). The figures shows the density of $\boldsymbol{N}$ gusts with an amplitude, $\boldsymbol{A} > \boldsymbol{3\sigma}$, averaged over a window of 3, 10, and 30 seconds (indicated by the black dashed lines). The gusts are centered and the wind speed is normalized with respect to the amplitude. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.



d) $12 < \overline{U} \leq 16$ m/s (N = 2792, 1701, 1033).

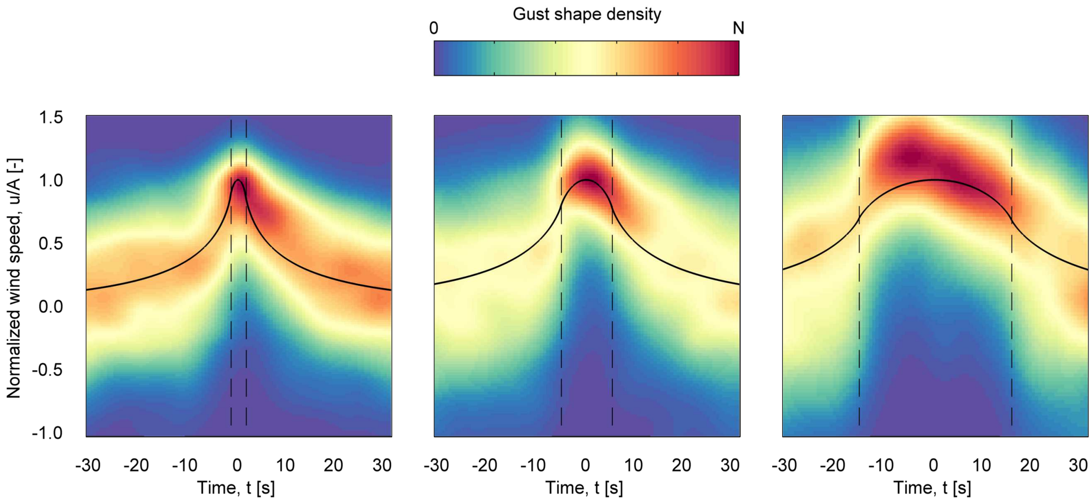


e) $16 < \overline{U} \leq 20$ m/s (N = 1790, 1168, 679).



f) $20 < \overline{U} \leq 24$ m/s (N = 443, 300, 171).

Figure 8 (cont.): Mean gust shapes extracted from the OWEZ met mast compared to the filtered autocorrelation function (—). The figures shows the density of $\boldsymbol{N}$ gusts with an amplitude, $\boldsymbol{A} > 3\boldsymbol{\sigma}$, averaged over a window of 3, 10, and 30 seconds (indicated by the black dashed lines). The gusts are centered and the wind speed is normalized with respect to the amplitude. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.



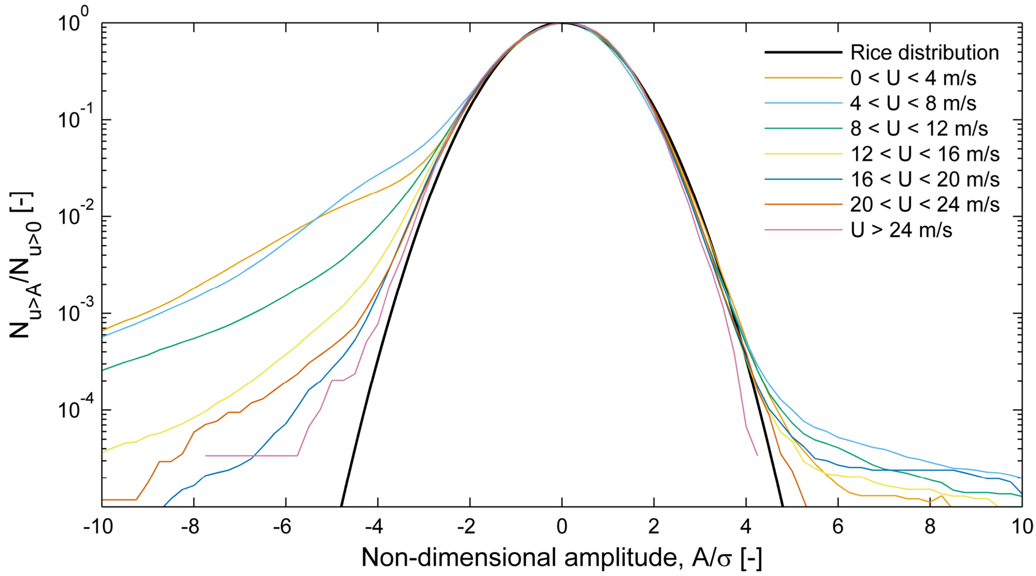
g) $\overline{U} \geq 24$ m/s (N = 60, 41, 23).

Figure 8 (cont.): Mean gust shapes extracted from the OWEZ met mast compared to the filtered autocorrelation function (**—**).. The figures shows the density of $\boldsymbol{N}$ gusts with an amplitude, $\boldsymbol{A} > \boldsymbol{3\sigma}$, averaged over a window of 3, 10, and 30 seconds (indicated by the black dashed lines). The gusts are centered and the wind speed is normalized with respect to the amplitude. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.

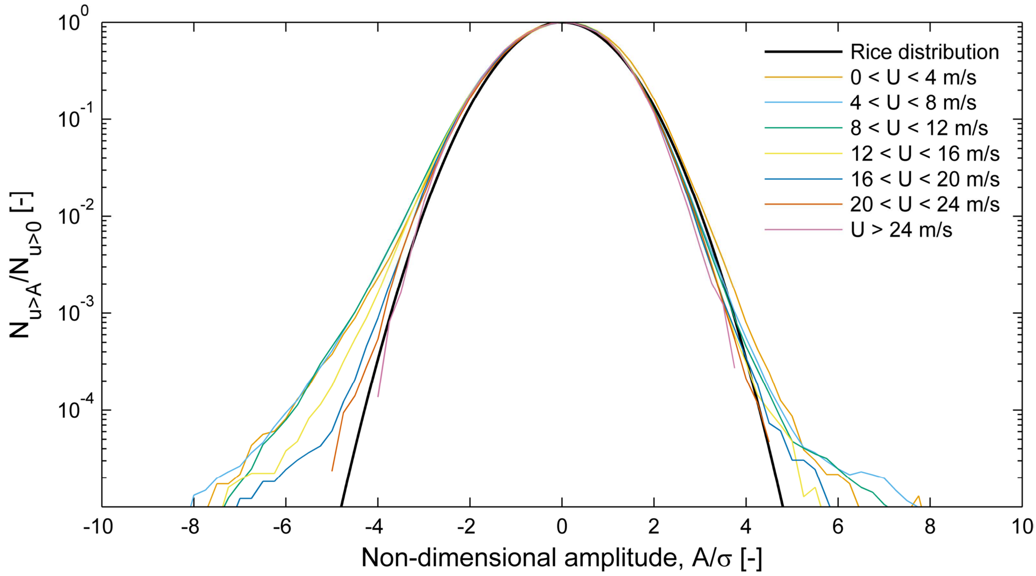
4.3 Probability of occurrence

Figure 9a through h shows the level excursions counted in 20 months of 4 Hz wind speed data measured at the OWEZ wind farm. Without filtering, the number of level excursions can deviate significantly from the Rice distribution for $A/\sigma < 2$ and $A/\sigma > 4$. However, the mean wind speed has a significant effect. At higher mean wind speeds (i.e., a higher rate of advection), an anemometer will sample the velocity field at larger *spatial* separations. Large-scale fluctuations are more Gaussian in nature, and will therefore agree better with the theoretical distribution, given by equation (3.5) (which is based on the assumption of a Gaussian process). The effect of a low-pass filter is similar to the effect of a higher wind speed, namely that the small-scale fluctuations are not taken into account. Though, it should be noted that at high amplitudes and high wind speeds, there may not be enough counts for a reliable trend.

From the data in figure 9, it seems that non-Gaussian behavior would not have any significant impact on extreme loads. First, one can argue that a wind turbine fulfills the same role as a low-pass filter, since the effect of small-scale fluctuations can easily cancel out over the length of a blade. And secondly, extreme loads for pitch-controlled turbines are likely to be found near and above the rated wind speed. Combined, these two aspects may well suppress the effects of non-Gaussianity. However, without actual load calculations, this of course remains mere speculation at this point.

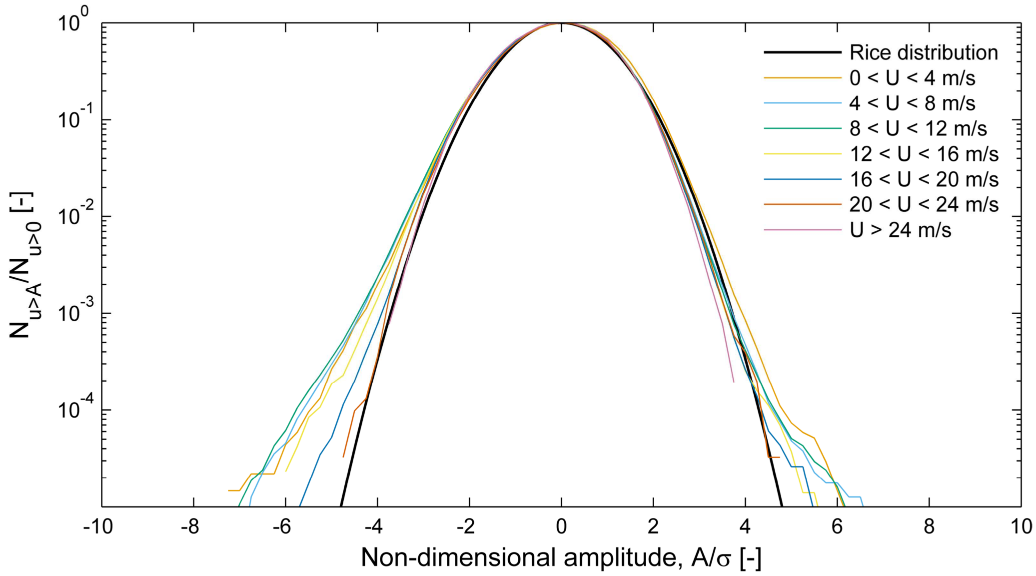


a) Level excursions for $\lambda = 0$ s.

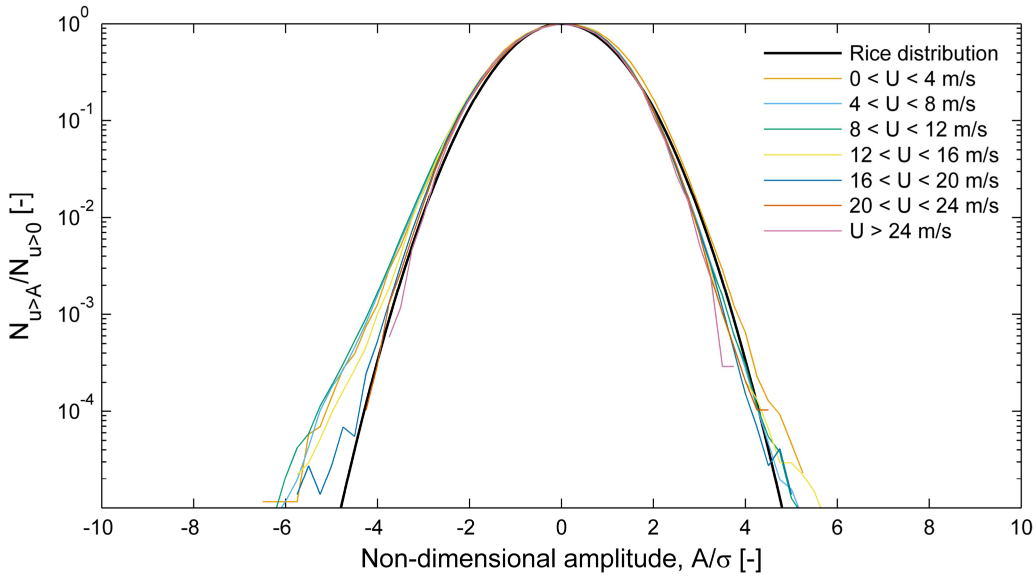


b) Level excursions for $\lambda = 3$ s.

Figure 9: Normalized frequency of level excursions where the velocity is averaged over a temporal window λ. Shown here are theoretical Rice distribution based on 10-minute statistics, compared to the actual counts in the OWEZ data, separated by mean wind speeds. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.

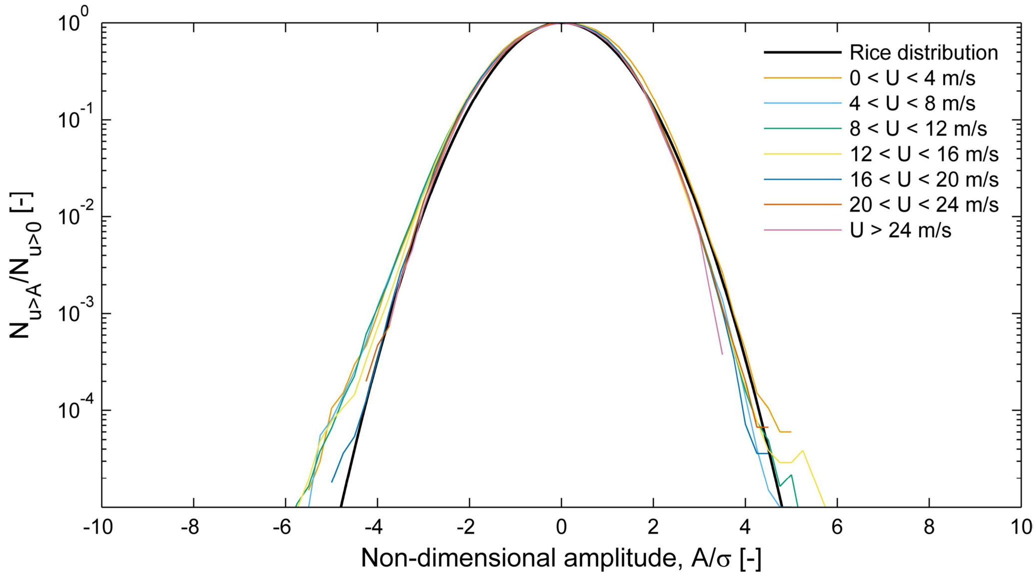


c) Level excursions for $\boldsymbol{\lambda} = 5$ s.

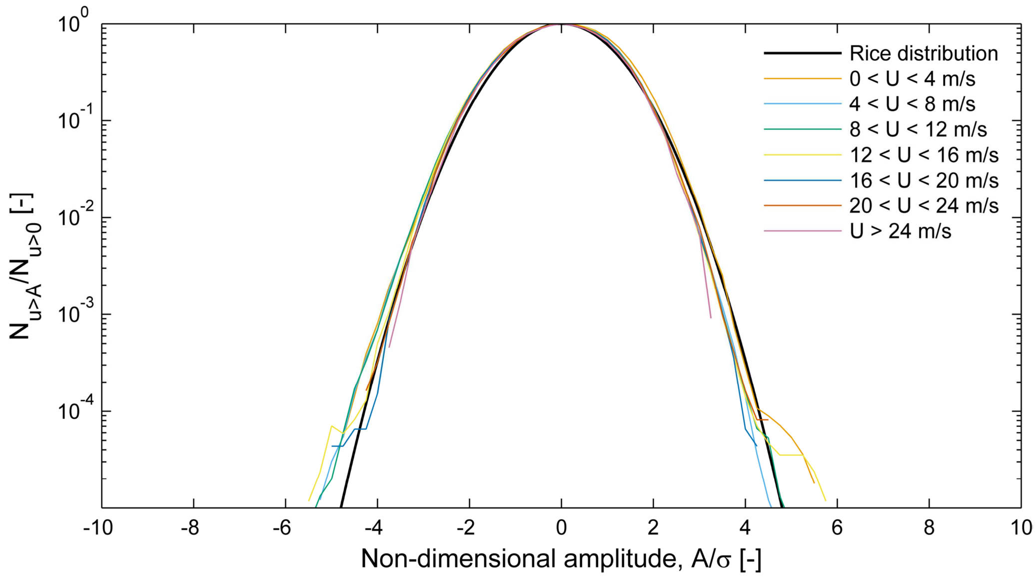


d) Level excursions for $\boldsymbol{\lambda} = 10$ s.

Figure 9 (cont.): Normalized frequency of level excursions where the velocity is averaged over a temporal window $\boldsymbol{\lambda}$. Shown here are theoretical Rice distribution based on 10-minute statistics, compared to the actual counts in the OWEZ data, separated by mean wind speeds. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.

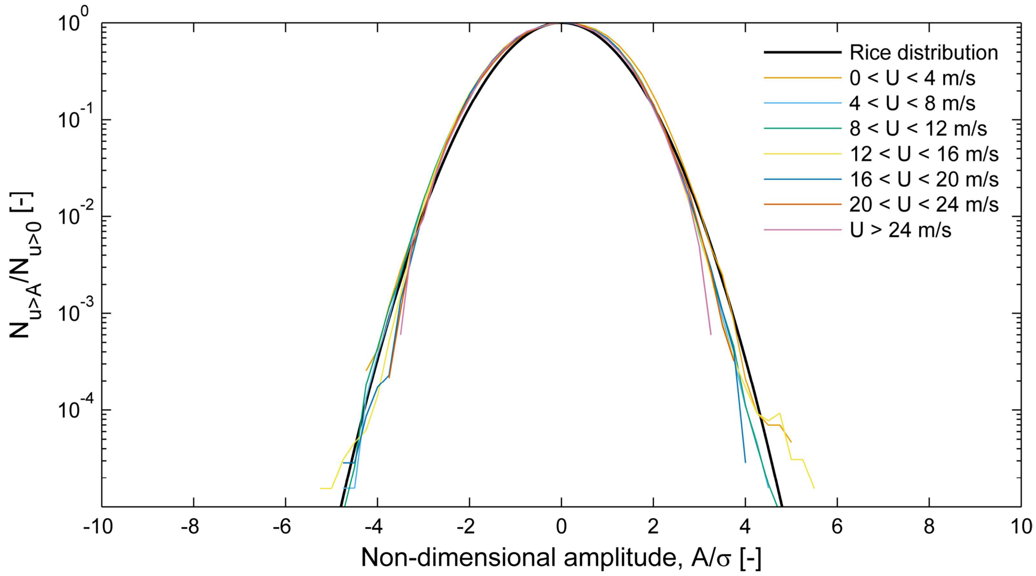


e) Level excursions for λ = 15 s.

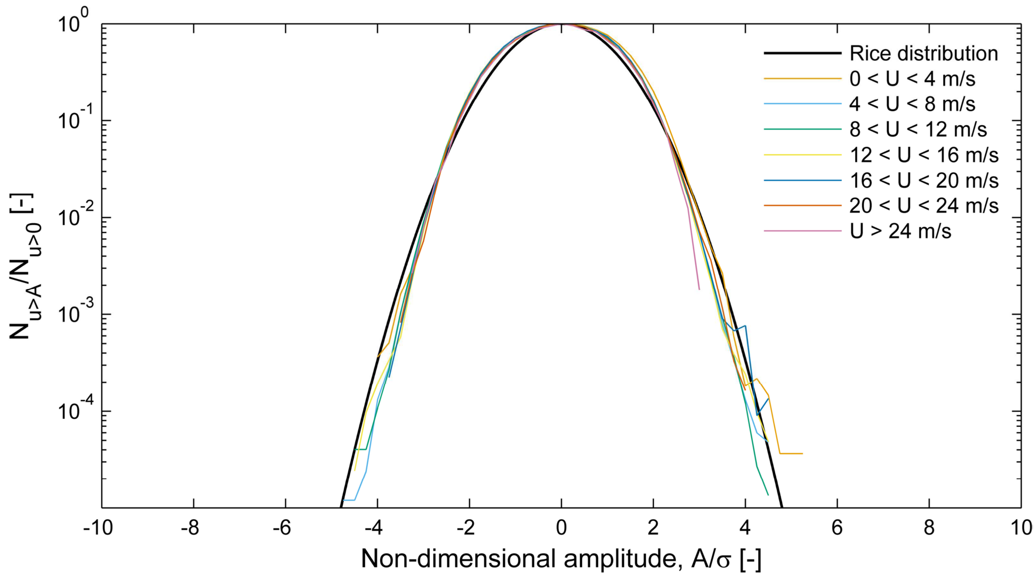


f) Level excursions for λ = 20 s.

Figure 9 (cont.): Normalized frequency of level excursions where the velocity is averaged over a temporal window λ. Shown here are theoretical Rice distribution based on 10-minute statistics, compared to the actual counts in the OWEZ data, separated by mean wind speeds. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.



g) Level excursions for λ = 30 s.



h) Level excursions for λ = 60 s.

Figure 9 (cont.): Normalized frequency of level excursions where the velocity is averaged over a temporal window λ. Shown here are theoretical Rice distribution based on 10-minute statistics, compared to the actual counts in the OWEZ data, separated by mean wind speeds. Data is taken from the sonic anemometer at 116 m and filtered for wind directions from 205 to 295°.

5 APPLICATION

To make this report as accessible as possible, this chapter contains a brief guide to the method, targeted at readers without a strong background in wind field modeling (i.e., the *users*).

5.1 The method in a nutshell

C*onstrained stochastic simulation* allows one to generate wind fields containing extreme events (i.e., gusts). These events arise from the statistics of "regular" turbulence. Therefore, the resulting wind fields are a representation of what can be found in very long time series. When the probability of such an event is known, it becomes possible to target the extreme load behavior of a structure, for example by using Monte Carlo methods with importance sampling.

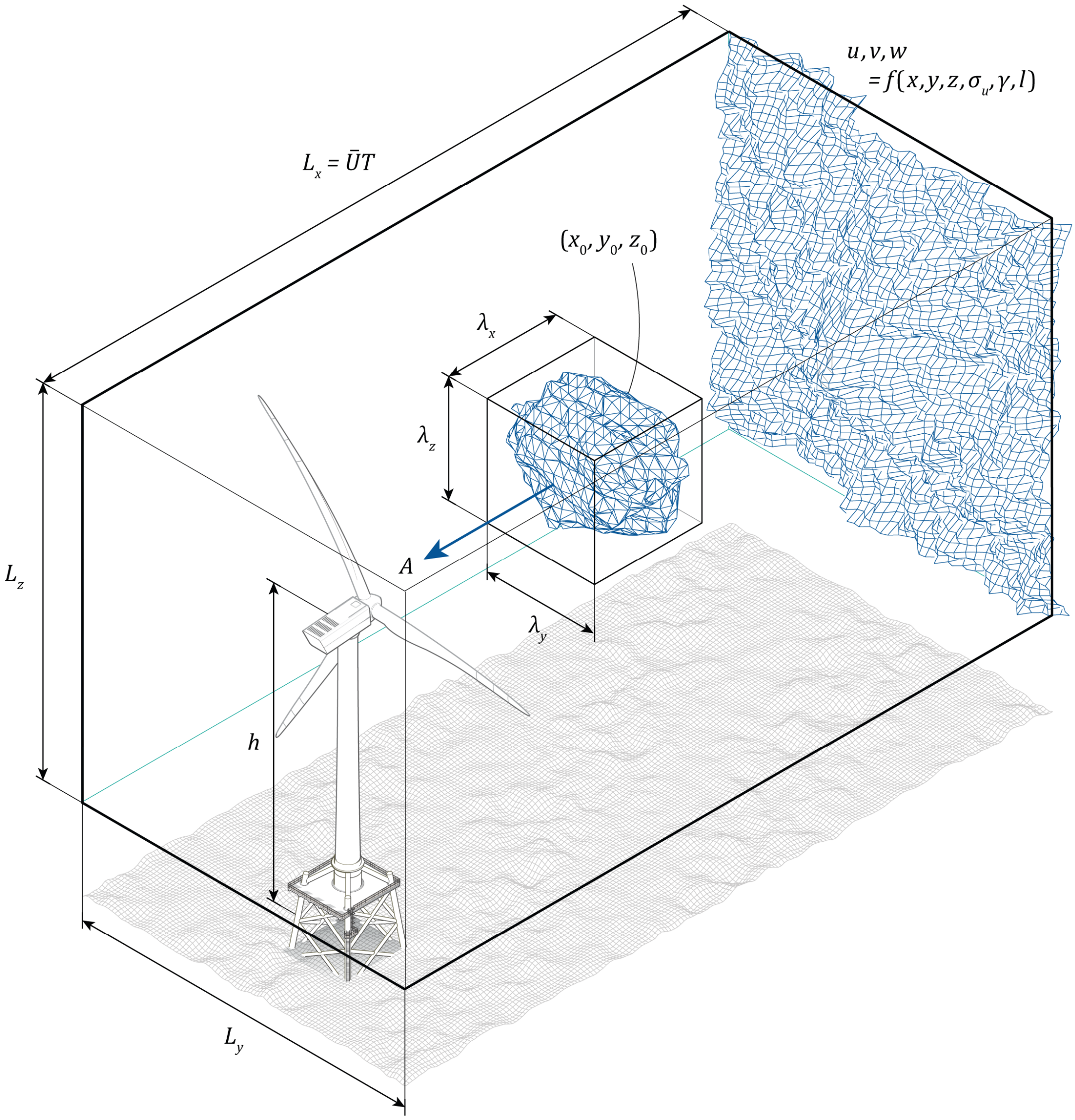


Figure 10: Sketch of the wind field with all the relevant parameters. Note that, in the IEC standards, the anisotropy parameter is set to $\boldsymbol{\gamma}$ = 3.9 and the turbulent length scale, $\boldsymbol{l}$, is given by $\mathbf{0.8\Lambda_1}$, where $\mathbf{\Lambda_1}$ is a height-dependent turbulence scale parameter.

Table 1: Relevant parameters with their descriptions.

Symbol			Unit	Description
used here	IEC	in code		
A	—	`a`	[m/s]	Gust amplitude
h	z	`h`	[m]	Hub height
L_x, L_y, L_z	—	`Lx, Ly, Lz`	[m]	Wind field length, width, and height
l	l	—	[m]	Turbulent length scale (see IEC 61400-1 Annex B)
$N_{u>A}$	—	`N`	[-]	Number of events where $u > A$
M_0	—	`M0`	[m²/s²]	Filtered zeroth-order spectral moment
$\mathbf{M}_2$	—	`M2`	[1/s²]	Filtered second-order spectral moment
T	—	`T`	[s]	Simulated time period
$\bar{U}$	V_{hub}	`U`	[m/s]	Mean wind speed at hub height
u, v, w	u_1, u_2, u_3	`u, v, w`	[m/s]	Velocity components
x, y, z	x, y, z	`x, y, z`	[m]	Coordinates of the spatial grid points
x_0, y_0, z_0	—	`x0, y0, z0`	[m]	Position of the gust's center
γ	γ	—	[-]	Anisotropy parameter (see IEC 61400-1 Annex B)
$\lambda_x, \lambda_y, \lambda_z$	—	`lx, ly, lz`	[m]	Gust length scales
σ_u	σ_1	`sigma1`	[m/s]	Longitudinal standard deviation

5.2 A step-by-step guide

Figure 10 shows a sketch of a wind field with all the parameters that are needed to simulate a gust. In addition, table 1 contains a nomenclature for this chapter. Note that, in the IEC standards, the anisotropy parameter is set to $\gamma = 3.9$ and the turbulent length scale is given by $0.8\Lambda_1$, where Λ_1 is a height-dependent turbulence scale parameter.

1. Set the mean wind speed, $\bar{U}$, the longitudinal standard deviation, σ_1, and the hub height, h.

```
U = 11.4;                % Mean wind speed [m/s]
I = 0.16;                % Reference turbulence intensity [-]
sigma1 = I*(0.75*U+5.6);% Longitudinal standard deviation [m/s]
h = 119;                 % Hub height [m]
```

2. Define a rectangular domain with a size of $\bar{U}T \times L_y \times L_z$ to fit the wind field.

```
T = 120;                 % Time period [s]
Lx = U*T;                % Domain length [m]
Ly = 200;                % Domain width [m]
Lz = 200;                % Domain height [m]
Nx = 2^10;               % Number of points in x-direction [-]
Ny = 2^5;                % Number of points in y-direction [-]
Nz = 2^5;                % Number of points in z-direction [-]
```

3. Pad the domain in all directions to avoid periodicity[19]. Moreover, ensure that $\bar{U}T, L_y, L_z > 8l$.

```
py = 12;                  % Array padding in y-direction [-]
pz = 12;                  % Array padding in z-direction [-]
dx = Lx/Nx;               % Step size in x-direction [m]
dy = Ly/Ny;               % Step size in y-direction [m]
dz = Lz/Nz;               % Step size in z-direction [m]
[x,y,z] = meshgrid(...
    dx:dx:Lx, ...
    (-(Ny+py-1)/2:(Ny+py-1)/2)*dy, ...
    (-(Nz+pz-1)/2:(Nz+pz-1)/2)*dz + h);
```

4. Define a gust event by setting its position, (x_0, y_0, z_0), amplitude, A, and the box over which the amplitude is averaged, $\lambda_x \times \lambda_y \times \lambda_z$.

```
A = 6;                    % Gust amplitude [m/s]
x0 = 60*U;                % Gust x-position [m]
y0 = 0;                   % Gust y-position [m]
z0 = h;                   % Gust z-position [m]
lx = 3*U;                 % Gust longitudinal length scale [m]
ly = 25;                  % Gust lateral length scale [m]
lz = 25;                  % Gust vertical length scale [m]
```

5. Compute the velocity field, $\mathbf{u}(\mathbf{x})$, together with the zeroth- and second-order spectral moments[20], M_0 and $\mathbf{M}_2$ (see appendix B).

```
[u,v,w,M0,M2] = ConstrainedIEC(x,y,z,sigma1,h,A,lx,ly,lz,x0,y0,z0);
```

6. Retrieve the unpadded domain size.

```
x = dx:dx:Lx;
y = (-(Ny-1)/2:(Ny-1)/2)*dy;
z = (-(Nz-1)/2:(Nz-1)/2)*dz + h;
u = u(py/2+(1:Ny),1:Nx,pz/2+(1:Nz));
v = v(py/2+(1:Ny),1:Nx,pz/2+(1:Nz));
w = w(py/2+(1:Ny),1:Nx,pz/2+(1:Nz));
```

7. Output the 3D velocity field to a binary file format and feed to a turbine model, such as Bladed, FAST, or HAWC2 (how is well-explained in appendix D and E of the TurbSim manual[21]).

[19] Mann, J. (1998). “Wind Field Simulation.” Probabilistic Engineering Mechanics 13 (4): 269–282. doi:10.1016/S0266-8920(97)00036-2.
[20] Please note that M_0 and $\mathbf{M}_2$ are used here to denote σ^2 and $\mathbf{R}$ in chapter 3. This is to avoid confusion with the unfiltered longitudinal variance, σ_1^2.
[21] Jonkman, B. J. and L. Kilcher (2012). “TurbSim User’s Guide (draft version)”. Technical Report. National Renewable Energy Laboratory, Golden, CO, United States.

8. Compute the expected number of level exceedances, $\mathrm{E}[N_{u>A}]$.

```
c1 = (sqrt(M2(1,1))*Lx + sqrt(M2(2,2))*Ly +
sqrt(M2(3,3))*Lz)/(2*pi*sqrt(M0));
c2 = ((sqrt(det(M2([1,2],[1,2])))*Lx*Ly + sqrt(det(M2([1,3],[1,3])))*Lx*Lz +
sqrt(det(M2([2,3],[2,3])))*Ly*Lz)/((2*pi)^(3/2) * M0));
c3 = Lx*Ly*Lz/((2*pi)^2 * M0^(3/2))*sqrt(det(M2));
N = (1-normcdf(A/sqrt(M0),0,1)) + exp(-A.^2/(2*M0)) .* (c1 +
c2.*(A/sqrt(M0)) + c3.*(A.^2/M0-1));
```

9. Compute the probability of occurrence with a Poisson distribution, $\mathrm{P}(N_{u>A} = k)$, using $k = 1$.

```
P = N*exp(-N);    % Probability of occurrence [-]
```

6 CONCLUSION

This report has shown how gusts can be generated in a spatial domain and connected to a probability of occurrence. The method can be used to simulate extreme events arising in the IEC DLC 1.1 and 1.3 load cases.

Gusts are set-up by an amplitude, position, and a low pass filter defining the volume over which the amplitude is averaged. The use of low-pass filters has appeared to be a vital ingredient of this method. By cutting off part of the high-wavenumber region of the turbulence spectrum, the second-order spectral moments become finite, which allows the statistics to be quantified. Furthermore, under the influence of low-pass filters, simulated gusts become larger and rarer. This may lead to interesting load cases when being fed to a turbine model.

The method presented here is an efficient way of simulating extreme events without having to extract them from very long time series. Moreover, simulated gusts do not rely on the assumption of a uniform velocity across the yz-plane (as with the Mexican hat wavelet), which leads to more realistic load cases.

A APPENDIX: EXAMPLE MATLAB CODES

The MATLAB scripts presented here are meant for the reader to be able to reproduce the various examples in this report. Each contains the full code for one example, but does not include any plotting commands. A new script for each example means that this appendix contains a lot of similar code snippets. However, it does not outweigh the convenience of being able to copy-paste things directly.

When checking the results, please be aware that the gust position may be located in between grid points, causing the amplitude to be slightly off in the case of large grid spacings.

Examples 7 relies on the function `pfq(a,b,z,d)`, the generalized hypergeometric function, ${}_pF_q$, which is not part of the standard MATLAB version. It can be found in the MATLAB file exchange[22].

A.1 A local velocity maximum (*example 1*)

```
%% Variables
% Turbulence properties
L = 33.6;                % Turbulence length scale
sigma_iso = 1.5;         % Isotropic standard deviation [m/s]

% Gust parameters
uc = 8;                  % Gust amplitude [m/s]
duc = 0;                 % First derivative
x0 = 100;                % Gust position [m]


%% Domain
% Spatial domain
Lx = 200;                % Sample length [m]
N = 2^12;                % Number of points
x = linspace(0,Lx,N);    % Position vector [m]
dx = x(2) - x(1);        % Step size [m]

% Nondimensional wave numbers
m = ifftshift(-N/2:N/2-1);
k = 2*pi*m*L/Lx;


%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Match with variance
E = 2*pi*E*L/Lx * sigma_iso^2 / (trapz(k,E));


%% Solve for constraints
% Constraint vector
b = [uc; duc];

```

[22] Huntley, J. (2012). “Generation of Random Variates”. URL: www.mathworks.com/matlabcentral/fileexchange/35008-generation-of-random-variates/content/pfq.m (Accessed 27 November 2013).

```
% DFT matrix with derivatives
A = [sqrt(E).*exp(1i * 2*pi*m/N * x0/dx); ...
     sqrt(E).*(1i*k).*exp(1i * 2*pi*m/N * x0/dx)];

% Unconstrained stochastic variable
n = randn(N,1) + 1i*randn(N,1);

% Constrained stochastic variable
nc = n + (A')/(A*(A')) * (b - A*n);


%% Reconstruct time series
u = N*real(ifft(nc.*sqrt(E')));
```

A.2 A local velocity maximum in 3D space (*example 2*)

```
%% Variables
% Turbulence properties
L = 33.6;                % Turbulence length scale
sigma_iso = 1.5;         % Isotropic standard deviation [m/s]

% Gust parameters
uc = [8; 0; 0];          % Gust amplitude [m/s]
duc = [0; 0; 0];         % First derivatives
x0 = [75; 125; 100];     % Gust position [m]


%% Domain
% Spatial domain
Lx = 200;                % Sample length in x-direction [m]
Ly = 200;                % Sample length in y-direction [m]
Lz = 200;                % Sample length in z-direction [m]
Nx = 2^6;                % Number of points in x-direction
Ny = 2^6;                % Number of points in y-direction
Nz = 2^6;                % Number of points in z-direction
x = linspace(0,Lx,Nx);   % Position vector in x-direction [m]
y = linspace(0,Ly,Ny);   % Position vector in y-direction [m]
z = linspace(0,Lz,Nz);   % Position vector in z-direction [m]
dx = x(2) - x(1);        % Step size in x-direction [m]
dy = y(2) - y(1);        % Step size in x-direction [m]
dz = z(2) - z(1);        % Step size in x-direction [m]
[x,y,z] = meshgrid(x,y,z);

% Non-dimensional wave numbers (the +1e-6 is added to prevent a zero
% wave number)
[mx, my, mz] = meshgrid(...
    -Nx/2:Nx/2-1, ...
    -Ny/2:Ny/2-1, ...
    -Nz/2:Nz/2-1);
mx = ifftshift(mx+1e-6);
my = ifftshift(my+1e-6);
mz = ifftshift(mz+1e-6);
kx = 2*pi*mx*L/Lx;
ky = 2*pi*my*L/Ly;
```

```
kz = 2*pi*mz*L/Lz;
k = sqrt(kx.^2 + ky.^2 + kz.^2);


%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Correlation matrix
B = sigma_iso * sqrt(2*pi^2 * L^3 * E ./ (Lx*Ly*Lz * k.^4));
C = zeros([3,3,size(x)]);
C(1,2,:,:,:) = B .* kz;
C(1,3,:,:,:) = B .* -ky;
C(2,1,:,:,:) = B .* -kz;
C(2,3,:,:,:) = B .* kx;
C(3,1,:,:,:) = B .* ky;
C(3,2,:,:,:) = B .* -kx;


%% Solve for constraints
% Constraint vector
b = [uc; duc];

% Reshape wave numbers to vector
mx = reshape(mx,1,[]);
my = reshape(my,1,[]);
mz = reshape(mz,1,[]);
C = reshape(C,3,[]);

% DFT matrix with first derivatives
A = [C .* kron(exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron((2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3))];

% Unconstrained stochastic variable
n = randn(size(A,2),1) + 1i*randn(size(A,2),1);

% Constrained stochastic variable
nc = n + A'/(A*A') * (b - A*n);

% Stochastic field (alternatively, the nested for-loops can be
% replaced by dZ = mtimesx(C,nc) to speed up the code)
dZ = zeros([3, size(x)]);
nc = reshape(nc, [3, 1, size(x)]);
C  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = C(:,:,n1,n2,n3) * nc(:,:,n1,n2,n3);
        end
    end
end
```

```
%% Reconstruct time series
u = Nx*Ny*Nz*real(ifftn(squeeze(dZ(1,:,:,:))));
v = Nx*Ny*Nz*real(ifftn(squeeze(dZ(2,:,:,:))));
w = Nx*Ny*Nz*real(ifftn(squeeze(dZ(3,:,:,:))));
```

A.3 A local maximum averaged over a cubic subvolume (*example 3*)

```
%% Variables
% Turbulence properties
L = 33.6;                % Turbulence length scale
sigma_iso = 1.5;         % Isotropic standard deviation [m/s]

% Gust parameters
uc = [5; 0; 0];          % Gust amplitude [m/s]
duc = [0; 0; 0];         % First derivatives
x0 = [100; 100; 100];    % Gust position [m]
lx = 50;                 % Gust longitudinal length scale [m]
ly = 50;                 % Gust lateral length scale [m]
lz = 50;                 % Gust vertical length scale [m]


%% Domain
% Spatial domain
Lx = 200;                % Sample length in x-direction [m]
Ly = 200;                % Sample length in y-direction [m]
Lz = 200;                % Sample length in z-direction [m]
Nx = 2^6;                % Number of points in x-direction
Ny = 2^6;                % Number of points in y-direction
Nz = 2^6;                % Number of points in z-direction
x = linspace(0,Lx,Nx);   % Position vector in x-direction [m]
y = linspace(0,Ly,Ny);   % Position vector in y-direction [m]
z = linspace(0,Lz,Nz);   % Position vector in z-direction [m]
dx = x(2) - x(1);        % Step size in x-direction [m]
dy = y(2) - y(1);        % Step size in y-direction [m]
dz = z(2) - z(1);        % Step size in z-direction [m]
[x,y,z] = meshgrid(x,y,z);

% Non-dimensional wave numbers (the +1e-6 is added to prevent a zero
% wave number)
[mx, my, mz] = meshgrid(...
    -Nx/2:Nx/2-1, ...
    -Ny/2:Ny/2-1, ...
    -Nz/2:Nz/2-1);
mx = ifftshift(mx+1e-6);
my = ifftshift(my+1e-6);
mz = ifftshift(mz+1e-6);
kx = 2*pi*mx*L/Lx;
ky = 2*pi*my*L/Ly;
kz = 2*pi*mz*L/Lz;
k = sqrt(kx.^2 + ky.^2 + kz.^2);

```

```
%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Correlation matrix
B = sigma_iso * sqrt(2*pi^2 * L^3 * E ./ (Lx*Ly*Lz * k.^4));
C = zeros([3,3,size(x)]);
C(1,2,:,:,:) = B .* kz;
C(1,3,:,:,:) = B .* -ky;
C(2,1,:,:,:) = B .* -kz;
C(2,3,:,:,:) = B .* kx;
C(3,1,:,:,:) = B .* ky;
C(3,2,:,:,:) = B .* -kx;


%% Solve for constraints
% Constraint vector
b = [uc; duc];

% Reshape wave numbers to vector
mx = reshape(mx,1,[]);
my = reshape(my,1,[]);
mz = reshape(mz,1,[]);
C = reshape(C,3,[]);

% Low pass filter
G = sinc(mx*lx/(Nx*dx)).*sinc(my*ly/(Ny*dy)).*sinc(mz*lz/(Nz*dz));

% DFT matrix with first derivatives
A = [C .* kron(G .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron(G .* (2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron(G .* (2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron(G .* (2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x0(1)-x(1))/dx + my/Ny * (x0(2)-y(1))/dy + mz/Nz * (x0(3)-z(1))/dz)), ones(1,3))];

% Unconstrained stochastic variable
n = randn(size(A,2),1) + 1i*randn(size(A,2),1);

% Constrained stochastic variable
nc = n + A'/(A*A') * (b - A*n);

% Stochastic field (alternatively, the nested for-loops can be
% replaced by dZ = mtimesx(C,nc) to speed up the code)
dZ = zeros([3, size(x)]);
nc = reshape(nc, [3, 1, size(x)]);
C  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = C(:,:,n1,n2,n3) * nc(:,:,n1,n2,n3);
        end
    end
end
```

```
%% Reconstruct time series
u = Nx*Ny*Nz*real(ifftn(squeeze(dZ(1,:,:,:))));
v = Nx*Ny*Nz*real(ifftn(squeeze(dZ(2,:,:,:))));
w = Nx*Ny*Nz*real(ifftn(squeeze(dZ(3,:,:,:))));
```

A.4 One-dimensional velocity jump over a distance (*example 4*)

```
%% Variables
% Turbulence properties
L = 33.6;                % Turbulence length scale
sigma_iso = 1.5;         % Isotropic standard deviation [m/s]

% Gust parameters
uc1 = -6;                % Amplitude of first gust [m/s]
duc1 = 0;                % First derivative of first gust
x1 = 90;                 % Position of first gust [m]
uc2 = 8;                 % Amplitude of second gust [m/s]
duc2 = 0;                % First derivative of second gust
x2 = 110;                % Position of second gust [m]


%% Domain
% Spatial domain
Lx = 200;                % Sample length [m]
N = 2^12;                % Number of points
x = linspace(0,Lx,N);    % Position vector [m]
dx = x(2) - x(1);        % Step size [m]

% Nondimensional wave numbers
m = ifftshift(-N/2:N/2-1);
k = 2*pi*m*L/Lx;


%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Match with variance
E = 2*pi*E*L/Lx * sigma_iso^2 / (trapz(k,E));


%% Solve for constraints
% Constraint vector
b = [uc1; duc1; uc2; duc2];

% DFT matrix with derivatives
A = [sqrt(E).*exp(1i * 2*pi*m/N * x1/dx); ...
     sqrt(E).*(1i*k).*exp(1i * 2*pi*m/N * x2/dx); ...
     sqrt(E).*exp(1i * 2*pi*m/N * x2/dx); ...
     sqrt(E).*(1i*k).*exp(1i * 2*pi*m/N * x2/dx)];

% Unconstrained stochastic variable
```

```
n = randn(N,1) + 1i*randn(N,1);

% Constrained stochastic variable
nc = n + (A')/(A*(A')) * (b - A*n);


%% Reconstruct time series
u = N*real(ifft(nc.*sqrt(E')));
```

A.5 A cluster of three single-point velocity maxima (*example 5*)

```
%% Variables
% Turbulence properties
L = 33.6;                 % Turbulence length scale
sigma_iso = 1.5;          % Isotropic standard deviation [m/s]

% Gust parameters
uc1 = [6; 0; 0];          % Amplitude of first gust [m/s]
duc1 = [0; 0; 0];         % First derivatives of first gust
x1 = [40; 90; 100];       % Position of first gust [m]
uc2 = [7; 0; 0];          % Amplitude of second gust [m/s]
duc2 = [0; 0; 0];         % First derivatives of second gust
x2 = [100; 150; 100];     % Position of second gust [m]
uc3 = [8; 0; 0];          % Amplitude of third gust [m/s]
duc3 = [0; 0; 0];         % First derivatives of third gust
x3 = [160; 30; 100];      % Position of third gust [m]


%% Domain
% Spatial domain
Lx = 200;                 % Sample length in x-direction [m]
Ly = 200;                 % Sample length in y-direction [m]
Lz = 200;                 % Sample length in z-direction [m]
Nx = 2^6;                 % Number of points in x-direction
Ny = 2^6;                 % Number of points in y-direction
Nz = 2^6;                 % Number of points in z-direction
x = linspace(0,Lx,Nx);    % Position vector in x-direction [m]
y = linspace(0,Ly,Ny);    % Position vector in y-direction [m]
z = linspace(0,Lz,Nz);    % Position vector in z-direction [m]
dx = x(2) - x(1);         % Step size in x-direction [m]
dy = y(2) - y(1);         % Step size in y-direction [m]
dz = z(2) - z(1);         % Step size in z-direction [m]
[x,y,z] = meshgrid(x,y,z);

% Non-dimensional wave numbers (the +1e-6 is added to prevent a zero
% wave number)
[mx, my, mz] = meshgrid(...
    -Nx/2:Nx/2-1, ...
    -Ny/2:Ny/2-1, ...
    -Nz/2:Nz/2-1);
mx = ifftshift(mx+1e-6);
my = ifftshift(my+1e-6);
mz = ifftshift(mz+1e-6);
kx = 2*pi*mx*L/Lx;
```

```
ky = 2*pi*my*L/Ly;
kz = 2*pi*mz*L/Lz;
k = sqrt(kx.^2 + ky.^2 + kz.^2);


%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Correlation matrix
B = sigma_iso * sqrt(2*pi^2 * L^3 * E ./ (Lx*Ly*Lz * k.^4));
C = zeros([3,3,size(x)]);
C(1,2,:,:,:) = B .* kz;
C(1,3,:,:,:) = B .* -ky;
C(2,1,:,:,:) = B .* -kz;
C(2,3,:,:,:) = B .* kx;
C(3,1,:,:,:) = B .* ky;
C(3,2,:,:,:) = B .* -kx;


%% Solve for constraints
% Constraint vector
b = [uc1; duc1; uc2; duc2; uc3; duc3];

% Reshape wave numbers to vector
mx = reshape(mx,1,[]);
my = reshape(my,1,[]);
mz = reshape(mz,1,[]);
C = reshape(C,3,[]);

% DFT matrix with first derivatives
A = [C .* kron(exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron((2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3)); ...
     C .* kron(exp(1i * 2*pi*(mx/Nx * (x2(1)-x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron((2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3)); ...
     C .* kron(exp(1i * 2*pi*(mx/Nx * (x3(1)-x(1))/dx + my/Ny * (x3(2)-y(1))/dy + mz/Nz * (x3(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron((2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x3(1)-x(1))/dx + my/Ny * (x3(2)-y(1))/dy + mz/Nz * (x3(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x3(1)-x(1))/dx + my/Ny * (x3(2)-y(1))/dy + mz/Nz * (x3(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron((2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x3(1)-x(1))/dx + my/Ny * (x3(2)-y(1))/dy + mz/Nz * (x3(3)-z(1))/dz)), ones(1,3))];

% Unconstrained stochastic variable
```

```
n = randn(size(A,2),1) + 1i*randn(size(A,2),1);

% Constrained stochastic variable
nc = n + A'/(A*A') * (b - A*n);

% Stochastic field (alternatively, the nested for-loops can be
% replaced by dZ = mtimesx(C,nc) to speed up the code)
dZ = zeros([3, size(x)]);
nc = reshape(nc, [3, 1, size(x)]);
C  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = C(:,:,n1,n2,n3) * nc(:,:,n1,n2,n3);
        end
    end
end


%% Reconstruct time series
u = Nx*Ny*Nz*real(ifftn(squeeze(dZ(1,:,:,:))));
v = Nx*Ny*Nz*real(ifftn(squeeze(dZ(2,:,:,:))));
w = Nx*Ny*Nz*real(ifftn(squeeze(dZ(3,:,:,:))));
```

A.6 Velocity jump between two planes (*example 6*)

```
%% Variables
% Turbulence properties
L = 33.6;                % Turbulence length scale
sigma_iso = 1.5;         % Isotropic standard deviation [m/s]

% Gust parameters
uc1 = [-5; 0; 0];        % Amplitude of first gust [m/s]
duc1 = [0; 0; 0];        % First derivatives of first gust
x1 = [75; 100; 100];     % Position of first gust [m]
uc2 = [5; 0; 0];         % Amplitude of second gust [m/s]
duc2 = [0; 0; 0];        % First derivatives of second gust
x2 = [125; 100; 100];    % Position of second gust [m]
lx = 0;                  % Gust longitudinal length scale [m]
ly = 50;                 % Gust lateral length scale [m]
lz = 50;                 % Gust vertical length scale [m]


%% Domain
% Spatial domain
Lx = 200;                % Sample length in x-direction [m]
Ly = 200;                % Sample length in y-direction [m]
Lz = 200;                % Sample length in z-direction [m]
Nx = 2^6;                % Number of points in x-direction
Ny = 2^6;                % Number of points in y-direction
Nz = 2^6;                % Number of points in z-direction
x = linspace(0,Lx,Nx);  % Position vector in x-direction [m]
y = linspace(0,Ly,Ny);  % Position vector in y-direction [m]
z = linspace(0,Lz,Nz);  % Position vector in z-direction [m]
```

```
dx = x(2) - x(1);       % Step size in x-direction [m]
dy = y(2) - y(1);       % Step size in y-direction [m]
dz = z(2) - z(1);       % Step size in z-direction [m]
[x,y,z] = meshgrid(x,y,z);

% Non-dimensional wave numbers (the +1e-6 is added to prevent a zero wave number)
[mx, my, mz] = meshgrid(...
    -Nx/2:Nx/2-1, ...
    -Ny/2:Ny/2-1, ...
    -Nz/2:Nz/2-1);
mx = ifftshift(mx+1e-6);
my = ifftshift(my+1e-6);
mz = ifftshift(mz+1e-6);
kx = 2*pi*mx*L/Lx;
ky = 2*pi*my*L/Ly;
kz = 2*pi*mz*L/Lz;
k = sqrt(kx.^2 + ky.^2 + kz.^2);


%% Spectrum function
% Non-dimensional von Kármán isotropic turbulence spectrum
E = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

% Correlation matrix
B = sigma_iso * sqrt(2*pi^2 * L^3 * E ./ (Lx*Ly*Lz * k.^4));
C = zeros([3,3,size(x)]);
C(1,2,:,:,:) = B .* kz;
C(1,3,:,:,:) = B .* -ky;
C(2,1,:,:,:) = B .* -kz;
C(2,3,:,:,:) = B .* kx;
C(3,1,:,:,:) = B .* ky;
C(3,2,:,:,:) = B .* -kx;


%% Solve for constraints
% Constraint vector
b = [uc1; duc1; uc2; duc2];

% Reshape wave numbers to vector
mx = reshape(mx,1,[]);
my = reshape(my,1,[]);
mz = reshape(mz,1,[]);
C = reshape(C,3,[]);

% Low pass filter
G = sinc(mx*lx/(Nx*dx)).*sinc(my*ly/(Ny*dy)).*sinc(mz*lz/(Nz*dz));

% DFT matrix with first derivatives
A = [C .* kron(G .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(3)); ...
     C(1,:) .* kron(G .* (2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3)); ...
     C(1,:) .* kron(G .* (2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3)); ...
```

```
    C(1,:) .* kron(G .* (2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x1(1)-
x(1))/dx + my/Ny * (x1(2)-y(1))/dy + mz/Nz * (x1(3)-z(1))/dz)), ones(1,3));
...
    C .* kron(G .* exp(1i * 2*pi*(mx/Nx * (x2(1)-x(1))/dx + my/Ny * (x2(2)-
y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(3)); ...
    C(1,:) .* kron(G .* (2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-
x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3));
...
    C(1,:) .* kron(G .* (2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-
x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3));
...
    C(1,:) .* kron(G .* (2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * (x2(1)-
x(1))/dx + my/Ny * (x2(2)-y(1))/dy + mz/Nz * (x2(3)-z(1))/dz)), ones(1,3))];

% Unconstrained stochastic variable
n = randn(size(A,2),1) + 1i*randn(size(A,2),1);

% Constrained stochastic variable
nc = n + A'/(A*A') * (b - A*n);

% Stochastic field (alternatively, the nested for-loops can be
% replaced by dZ = mtimesx(C,nc) to speed up the code)
dZ = zeros([3, size(x)]);
nc = reshape(nc, [3, 1, size(x)]);
C  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = C(:,:,n1,n2,n3) * nc(:,:,n1,n2,n3);
        end
    end
end


%% Reconstruct time series
u = Nx*Ny*Nz*real(ifftn(squeeze(dZ(1,:,:,:))));
v = Nx*Ny*Nz*real(ifftn(squeeze(dZ(2,:,:,:))));
w = Nx*Ny*Nz*real(ifftn(squeeze(dZ(3,:,:,:))));
```

A.7 Return levels for wind speeds in a domain using the IEC NTM *(example 7)*

```
%% Problem
% Turbine properties
D = 178.332;            % Rotor diameter [m]
R = D/2;                % Rotor radius [m]
H = 119.03;             % Hub height

% Frontal area
T = 4*60;               % Sample time period [s]
By = 100;               % Width of frontal area [m]
Bz = 100;               % Height of frontal area [m]


%% IEC class 1A
```

```
% Wind class parameters
Vref = 50;              % Reference wind speed [m/s]
Iref = 0.16;            % Turbulence intensity [-]
Gamma = 3.9;            % Anisotropy parameter [-]
alpha = 0.2;            % Shear exponent [-]
Vave = 0.2*Vref;        % Average wind speed [m/s]

% Turbulence length scale
if H <= 60
    Lambda1 = 0.7*H;
else
    Lambda1 = 42;
end
L = 0.8*Lambda1;

% Pre-allocate vector
N = zeros(1,length(-10:0.01:10));


%% Run through all wind speeds
for U = 1:30

    % Rayleigh distribution
    fU = raylpdf(U,sqrt(2/pi)*Vave);

    % Standard deviations
    sigma1 = Iref*(0.75*U + 5.6);
    sigma_iso = 0.55*sigma1;
    sigma2 = 0.7*sigma1;
    sigma3 = 0.5*sigma1;


    %% Domain
    % Spatial domain
    Lx = U*T;               % Sample length in x-direction [m]
    Ly = 8*L;               % Sample length in y-direction [m]
    Lz = 8*L;               % Sample length in z-direction [m]
    Nx = 2^9;               % Number of points in x-direction
    Ny = 2^6;               % Number of points in y-direction
    Nz = 2^6;               % Number of points in z-direction
    x = linspace(0,Lx,Nx);  % Position vector in x-direction [m]
    y = linspace(0,Ly,Ny);  % Position vector in y-direction [m]
    z = linspace(0,Lz,Nz);  % Position vector in z-direction [m]
    dx = x(2) - x(1);       % Step size in x-direction [m]
    dy = y(2) - y(1);       % Step size in y-direction [m]
    dz = z(2) - z(1);       % Step size in z-direction [m]
    [x,y,z] = meshgrid(x,y,z);

    % Filter
    lx = 3*U;               % Gust longitudinal length scale [m]
    ly = 5;                 % Gust lateral length scale [m]
    lz = 5;                 % Gust vertical length scale [m]

    % Non-dimensional wave numbers (the +1e-6 is added to prevent a
    zero wave number)
    [mx, my, mz] = meshgrid(...
        -Nx/2:Nx/2-1, ...
```

```
        -Ny/2:Ny/2-1, ...
        -Nz/2:Nz/2-1);
    mx = ifftshift(mx+1e-6);
    my = ifftshift(my+1e-6);
    mz = ifftshift(mz+1e-6);
    kx = 2*pi*mx*L/Lx;
    ky = 2*pi*my*L/Ly;
    kz = 2*pi*mz*L/Lz;
    k = sqrt(kx.^2 + ky.^2 + kz.^2);


    %% Mann model
    % Non-dimensional distortion time
    beta = Gamma ./ (k.^(2/3) .* sqrt(real(pfq([1/3, 17/6], 4/3, -
    k.^-2))));
    kz0 = kz + beta.*kx;
    k0 = sqrt(kx.^2 + ky.^2 + kz0.^2);


    %% Spectrum function
    % Low pass filter
    G = sinc(mx*lx/(Nx*dx)) .* sinc(my*ly/(Ny*dy)) .*
    sinc(mz*lz/(Nz*dz));

    % Non-dimensional von Kármán isotropic turbulence spectrum
    E0 = 1.453*k.^4 ./ (1 + k.^2).^(17/6);

    % Shear model parameters
    C1 = (beta.*kx.^2 .* (kx.^2 + ky.^2 - kz.*(kz + beta.*kx))) ./
    (k.^2 .* (kx.^2 + ky.^2));
    C2 = (ky.*k0.^2) ./ (kx.^2 + ky.^2).^(3/2) .*
    atan2(real(beta.*kx.*sqrt(kx.^2 + ky.^2)), real(k0.^2 - (kz +
    beta.*kx).*kx.*beta));
    zeta1 = C1 - ky./kx .* C2;

    % Spectral tensor u-component
    Phi_uu = sigma_iso^2 .* E0./(4*pi*k0.^4) .* (k0.^2 - kx.^2 –
    2*kx.*kz0.*zeta1 + (kx.^2+ky.^2).*zeta1.^2);

    %% Spectral moments
    % Zeroth order
    sigma = sqrt(real(sum(sum(sum(G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz)));

    % Second order
    r_xx = sum(sum(sum(kx/L.*kx/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    r_xy = sum(sum(sum(kx/L.*ky/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    r_xz = sum(sum(sum(kx/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    r_yy = sum(sum(sum(ky/L.*ky/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    r_yz = sum(sum(sum(ky/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    r_zz = sum(sum(sum(kz/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
    Ruu = [r_xx, r_xy, r_xz; ...
           r_xy, r_yy, r_yz; ...
           r_xz, r_yz, r_zz];
    Ruu = real(Ruu);


    %% Expectated number of level excursions
```

```
    A = (-10:0.01:10)*sigma;
    Eu = (1-normcdf(A/sigma,0,1)) + exp(-A.^2/(2*sigma^2)) .*
((sqrt(Ruu(1,1))*Lx + sqrt(Ruu(2,2))*By + sqrt(Ruu(3,3))*Bz)/(2*pi*sigma) +
((sqrt(det(Ruu([1,2],[1,2])))*Lx*By + sqrt(det(Ruu([1,3],[1,3])))*Lx*Bz +
sqrt(det(Ruu([2,3],[2,3])))*By*Bz)/((2*pi)^(3/2) * sigma^2)).*(A/sigma) +
Lx*By*Bz/((2*pi)^2 * sigma^3)*sqrt(det(Ruu)).*(A.^2/sigma^2-1));

    N = N + fU*Eu;

end


%% Plotting
A = -10:0.01:10;
figure()
plot(A, log(T./N))
xlim([2 8])
ylim(log([1, 1000*365*24*3600]))
set(gca, 'YTick', log([1, 60, 600, 3600, 24*3600, 31*24*3600, 365*24*3600,
10*365*24*3600, 50*365*24*3600, 200*365*24*3600, 1000*365*24*3600]),
'YTickLabel', {'1 second', '1 minute', '10 minutes', '1 hour', '1 day', '1
month', '1 year', '10 years', '50 years', '200 years', '1000 years'})
ylabel('Return period')
xlabel('Non-dimensional amplitude, A/\sigma')
```

B MATLAB FUNCTION TO GENERATE IEC-COMPATIBLE GUSTS

The MATLAB function here includes part of the Mann model and requires the function `pfq(a,b,z,d)`, the generalized hypergeometric function, ${}_pF_q$, which is not part of the standard MATLAB version. It can be found in the MATLAB file exchange[23].

```
function[u,v,w,M0,M2] = ConstrainedIEC(x,y,z,sigma1,h,a,lx,ly,lz,x0,y0,z0)
% Constrained stochastic simulation of a spatial wind gust.
%
%   [u,v,w,M0,M2] = ConstrainedIEC(x,y,z,sigma1,h,a,lx,ly,lz,x0,y0,z0)
%
%   Returns a three-dimensional turbulent velocity field with a wind gust
%   embedded into it. The resulting field is obtained by constraining the
%   randomization of the wave numbers, yielding a stochastic velocity field
%   that satisfies a certain input while adhering to the statistics. The
%   spectral properties here are defined by the Mann model described in
%   IEC 61400-1 Annex B.
%
%   Remark 1: The simulated wind field is periodic in 3 directions. If the
%   spatial domain is small in comparison to the gust structure, low-wave
%   number components will become dominant and give unrealistic results.
%
%   Remark 2: In Mann (1998), it is argued that the approximation for the
%   correlation tensor may become poor when the dimensions of the box (in
%   any direction) are less than ~8L, where L is the turbulent length
%   scale.
%
%   Remark 3: The Mann model is derived under the assumption of a uniform
%   shear (i.e. a linear wind profile) and produces a homogeneous field of
%   turbulence. Because of this, the output can quickly become unreliable
%   as vertical separations become large, or if the domain is close to the
%   ground level.
%
%   INPUT:
%   x,y,z   (double)    Position vectors as created by meshgrid [m].
%   sigma1  (double)    Longitudinal standard deviation [m/s].
%   h       (double)    Hub height [m] on which the turbulence scale
%                       parameter is based.
%   a       (double)    Gust amplitude averaged over volume lx*ly*lz [m/s].
%   lx      (double)    Gust longitudinal length scale [m].
%   ly      (double)    Gust lateral length scale [m].
%   lz      (double)    Gust vertical length scale [m].
%   x0      (double)    Longitudinal position of gust center [m].
%   y0      (double)    Lateral position of gust center [m].
%   z0      (double)    Vertical position of gust center [m].
%
%   OUTPUT:
%   u,v,w   (double)    Velocity components [m/s].
%   M0      (double)    Filtered zeroth-order spectral moment [m²/s²].
%   M2      (double)    Filtered second-order spectral moment [1/s²]
%
%   AUTHOR:
```

[23] Huntley, J. (2012). "Generation of Random Variates". URL: www.mathworks.com/matlabcentral/fileexchange/35008-generation-of-random-variates/content/pfq.m (Accessed 27 November 2013).

```
%   René Bos
%   R.Bos-1@tudelft.nl
%   Delft University of Technology
%   May 2014 (last updated January 2015)
%
%   LITERATURE:
%   - Bierbooms, Wim (2005). "Constrained Stochastic Simulation --
%     Generation of Time Series around Some Specific Event in a Normal
%     Process." Extremes 8 (3): pp. 207–224.
%   - IEC (2005). IEC 61400-1 Wind Turbines - Part 1: Design Requirements,
%     3rd ed. International Electrotechnical Commission.
%   - Mann, Jakob (1998). "Wind Field Simulation." Probabilistic
%     Engineering Mechanics 13(4), pp. 269–282.


%% Derived properties (IEC 61400-1)
sigma_iso = 0.55*sigma1;
sigma2 = 0.7*sigma1;
sigma3 = 0.5*sigma1;
Gamma = 3.9;
if h < 60
    Lambda = 0.7*h;
else
    Lambda = 42;
end
L = 0.8*Lambda;


%% Ensure that domain is increasing and starting from zero
flipx = false;
flipy = false;
flipz = false;

if x(end) < x(1)
    flipx = true;
    x = -x;
    x0 = -x0;
end
if y(end) < y(1)
    flipy = true;
    y = -y;
    y0 = -y0;
end
if z(end) < z(1)
    flipz = true;
    z = -z;
    z0 = -z0;
end

x0 = x0 - x(1);
y0 = y0 - y(1);
z0 = z0 - z(1);

x = x - x(1);
y = y - y(1);
z = z - z(1);

```

```
%% Domain size
dx = x(1,2,1) - x(1,1,1);
dy = y(2,1,1) - y(1,1,1);
dz = z(1,1,2) - z(1,1,1);

Lx = abs(x(end) - x(1));
Ly = abs(y(end) - y(1));
Lz = abs(z(end) - z(1));

Nx = size(x,2);
Ny = size(y,1);
Nz = size(z,3);


%% Wave number discretization
[mx, my, mz] = meshgrid(...
    -Nx/2:Nx/2-1, ...
    -Ny/2:Ny/2-1, ...
    -Nz/2:Nz/2-1);

mx = ifftshift(mx+1e-6);
my = ifftshift(my+1e-6);
mz = ifftshift(mz+1e-6);

kx = 2*pi*mx*L/Lx;
ky = 2*pi*my*L/Ly;
kz = 2*pi*mz*L/Lz;

k = sqrt(kx.^2 + ky.^2 + kz.^2);


%% Mann model
% Non-dimensional distortion time
beta = Gamma ./ (k.^(2/3) .* sqrt(real(pfq([1/3, 17/6], 4/3, -k.^-2))));
kz0 = kz + beta.*kx;
k0 = sqrt(kx.^2 + ky.^2 + kz0.^2);

% Non-dimensional von Kármán isotropic turbulence spectrum
E0 = 1.453*k0.^4 ./ (1 + k0.^2).^(17/6);

% Shear model parameters
C1 = (beta.*kx.^2 .* (kx.^2 + ky.^2 - kz.*(kz + beta.*kx))) ./ (k.^2 .* (kx.^2
+ ky.^2));
C2 = (ky.*k0.^2) ./ (kx.^2 + ky.^2).^(3/2) .* atan2(real(beta.*kx.*sqrt(kx.^2 +
ky.^2)), real(k0.^2 - (kz + beta.*kx).*kx.*beta));
zeta1 = C1 - ky./kx .* C2;
zeta2 = C2 + ky./kx .* C1;
B = sigma_iso * sqrt(2*pi^2 * L^3 * E0 ./ (Lx*Ly*Lz * k0.^4));

% Correlation matrix
C = zeros([3,3,size(x)]);
C(1,1,:,:,:) = B .* ky.*zeta1;
C(1,2,:,:,:) = B .* (kz0 - kx.*zeta1);
C(1,3,:,:,:) = B .* -ky;
C(2,1,:,:,:) = B .* (ky.*zeta2 - kz0);
C(2,2,:,:,:) = B .* -kx.*zeta2;
C(2,3,:,:,:) = B .* kx;
```

```
C(3,1,:,:,:) = B .* ky.*k0.^2./k.^2;
C(3,2,:,:,:) = B .* -kx.*k0.^2./k.^2;

% Spectral tensor u-component
Phi_uu = sigma_iso^2 .* E0./(4*pi*k0.^4) .* (k0.^2 - kx.^2 - 2*kx.*kz0.*zeta1 + (kx.^2+ky.^2).*zeta1.^2);


%% Spectral moments
% Low pass filter
G = sinc(mx*lx/(Nx*dx)) .* sinc(my*ly/(Ny*dy)) .* sinc(mz*lz/(Nz*dz));

% Zeroth order
M0 = real(sum(sum(sum(G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz));

% Second order
m_xx = sum(sum(sum(kx/L.*kx/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
m_xy = sum(sum(sum(kx/L.*ky/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
m_xz = sum(sum(sum(kx/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
m_yy = sum(sum(sum(ky/L.*ky/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
m_yz = sum(sum(sum(ky/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
m_zz = sum(sum(sum(kz/L.*kz/L.*G.^2.*Phi_uu))) * (2*pi)^3*L^3/(Lx*Ly*Lz);
M2 = [m_xx, m_xy, m_xz; ...
      m_xy, m_yy, m_yz; ...
      m_xz, m_yz, m_zz];
M2 = real(M2);


%% Unconditioned velocity field
% Unconstrained stochastic variable
n = randn(3*Nx*Ny*Nz,1) + 1i*randn(3*Nx*Ny*Nz,1);

% Stochastic field
dZ = zeros([3, size(x)]);
ni = reshape(n, [3, 1, size(x)]);
Ci  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = Ci(:,:,n1,n2,n3) * ni(:,:,n1,n2,n3);
        end
    end
end

% Reconstruct time series
ui = Nx*Ny*Nz*real(ifftn(squeeze(dZ(1,:,:,:))));
vi = Nx*Ny*Nz*real(ifftn(squeeze(dZ(2,:,:,:))));
wi = Nx*Ny*Nz*real(ifftn(squeeze(dZ(3,:,:,:))));

% Determine factors to correct for component variance
Ku = sigma1/std(ui(:));
Kv = sigma2/std(vi(:));
Kw = sigma3/std(wi(:));


%% Set up constraints
% Scaled amplitude constraint
```

```
uc = [a; 0; 0] ./ [Ku; Kv; Kw];
duc = [0; 0; 0];
b = [uc; duc];

% Position constraint
x0 = [x0; y0; z0];

% Reshape wave numbers to vector
mx = reshape(mx,1,[]);
my = reshape(my,1,[]);
mz = reshape(mz,1,[]);
C = reshape(C,3,[]);

% Low pass filter
G = sinc(mx*lx/(Nx*dx)) .* sinc(my*ly/(Ny*dy)) .* sinc(mz*lz/(Nz*dz));

% Complete Fourier transform matrix
A = [C .* kron(G.* exp(1i * 2*pi*(mx/Nx * x0(1)/dx + my/Ny * x0(2)/dy + mz/Nz *
x0(3)/dz)), ones(3)); ...
     C(1,:) .* kron(G .* (2i*pi*mx/Lx) .* exp(1i * 2*pi*(mx/Nx * x0(1)/dx +
my/Ny * x0(2)/dy + mz/Nz * x0(3)/dz)), ones(1,3)); ...
     C(1,:) .* kron(G .* (2i*pi*my/Ly) .* exp(1i * 2*pi*(mx/Nx * x0(1)/dx +
my/Ny * x0(2)/dy + mz/Nz * x0(3)/dz)), ones(1,3)); ...
     C(1,:) .* kron(G .* (2i*pi*mz/Lz) .* exp(1i * 2*pi*(mx/Nx * x0(1)/dx +
my/Ny * x0(2)/dy + mz/Nz * x0(3)/dz)), ones(1,3))];


%% Conditioned velocity field
% Constrained stochastic variable
nc = n + A'/(A*A') * (b - A*n);

% Stochastic field
dZ = zeros([3, size(x)]);
nc = reshape(nc, [3, 1, size(x)]);
C  = reshape(C,  [3, 3, size(x)]);
for n1 = 1:size(y,1)
    for n2 = 1:size(x,2)
        for n3 = 1:size(z,3)
            dZ(:,n1,n2,n3) = C(:,:,n1,n2,n3) * nc(:,:,n1,n2,n3);
        end
    end
end

% Reconstruct time series
u = Ku * Nx*Ny*Nz * real(ifftn(squeeze(dZ(1,:,:,:))));
v = Kv * Nx*Ny*Nz * real(ifftn(squeeze(dZ(2,:,:,:))));
w = Kw * Nx*Ny*Nz * real(ifftn(squeeze(dZ(3,:,:,:))));

% If necessary, flip domains to match original input
if flipx
    u = flipdim(u,2);
    v = flipdim(v,2);
    w = flipdim(w,2);
end
if flipy
    u = flipdim(u,1);
    v = flipdim(v,1);
```

```
    w = flipdim(w,1);
end
if flipz
    u = flipdim(u,3);
    v = flipdim(v,3);
    w = flipdim(w,3);
end
if sign(abs(max(u(:))) - abs(min(u(:)))) ~= sign(a)
    u = -u;
end
```