Chapter 12
Spatial Process Simulation

Dirk P. Kroese and Zdravko I. Botev

Abstract The simulation of random spatial data on a computer is an important tool for understanding the behavior of spatial processes. In this chapter we describe how to simulate realizations from the main types of spatial processes, including Gaussian and Markov random fields, point processes, spatial Wiener processes, and Lévy fields. Concrete MATLAB code is provided.

12.1 Introduction

The collection and analysis of spatially arranged measurements and patterns is of interest to many scientific and engineering disciplines, including the earth sciences, materials design, urban planning, and astronomy. Examples of spatial data are geostatistical measurements, such as groundwater contaminant concentrations, temperature reports from different cities, maps of the locations of meteorite impacts or geological faults, and satellite images or demographic maps.

From a mathematical point of view, a spatial process is a collection of random variables $\{X_t, t \in \mathcal{T}\}$ where the *index set* $\mathcal{T}$ is some subset of the d-dimensional Euclidean space $\mathbb{R}^d$. Then, X_t can be a random quantity associated with a spatial position t rather than time. The set of possible values of X_t is called the *state space* of the spatial process. Thus, spatial processes can be classified into four types, based on whether the index set and state space are continuous or discrete. An example of a spatial process with a discrete index set and a discrete state space is the Ising model in statistical mechanics, where sites arranged on a grid are assigned either a positive

Dirk P. Kroese
School of Mathematics and Physics, The University of Queensland, Brisbane 4072, Australia e-mail: kroese@maths.uq.edu.au

Zdravko I. Botev
School of Mathematics and Statistics, The University of New South Wales, Sydney 2052, Australia e-mail: botev@unsw.edu.au

V. Schmidt (ed.), *Stochastic Geometry, Spatial Statistics and Random Fields*,
Lecture Notes in Mathematics 2120, DOI 10.1007/978-3-319-10064-7_12

or negative "spin"; see, for example, [387]. Image analysis, where a discrete number of pixels is assigned a continuous gray scale, provides an example of a process with a discrete index set and a continuous state space. A random configuration of points in $\mathbb{R}^d$ can be viewed as an example of a spatial process with a continuous index set and discrete state space $\{0,1\}$. If, in addition, continuous measurements are recorded at these points (e.g., rainfall data), one obtains a process in which both the index set and the state space are continuous.

Spatial processes can also be classified according to their distributional properties. For example, if any choice of random variables in a spatial process has jointly a multivariate normal distribution, the process is said to be *Gaussian*. Another important property is the *Markov* property, which deals with the local conditional independence of the random variables in the spatial process. A prominent class of spatial processes is that of the *point processes*, which are characterized by the random positions of points in space. The most important example is the *Poisson process*, whose foremost property is that the (random) numbers of points in nonoverlapping sets are independent of each other. *Lévy fields* are spatial processes that generalize this independence property. Throughout this chapter we provide computer implementation of the algorithms in MATLAB. Because of its simple syntax, excellent debugging tools, and extensive toolboxes, MATLAB is the de facto choice for numerical analysis and scientific computing. Moreover, it has one of the fastest implementations of the FFT, which is used in this chapter.

The rest of this chapter is organized as follows. We discuss in Sect. 12.2 the simulation of spatial processes that are both Gaussian and Markov. In Sect. 12.3 we consider various classes of spatial point processes, including the Poisson, compound Poisson, cluster, and Cox processes, and explain how to simulate these. Sect. 12.4 looks at ways of simulating spatial processes based on the Wiener process. Finally, Sect. 12.5 deals with the simulation of Lévy processes and fields.

12.2 Gaussian Markov Random Fields

A spatial stochastic process on $\mathbb{R}^2$ or $\mathbb{R}^3$ is often called a *random field*. Fig. 12.1 depicts realizations of three different types of random fields that are characterized by *Gaussian* and *Markovian* properties, which are discussed below.

12.2.1 Gaussian Property

A stochastic process $\{\tilde{X}_t, t \in \mathcal{T}\}$ is said to be *Gaussian* if all its finite-dimensional distributions are Gaussian (normal). That is, if for any choice of n and $t_1, \ldots, t_n \in \mathcal{T}$, we have

$$X \stackrel{\text{def}}{=} (X_1, \ldots, X_n)^\intercal \stackrel{\text{def}}{=} (\tilde{X}_{t_1}, \ldots, \tilde{X}_{t_n})^\intercal \sim N(\mu, \Sigma) \tag{12.1}$$

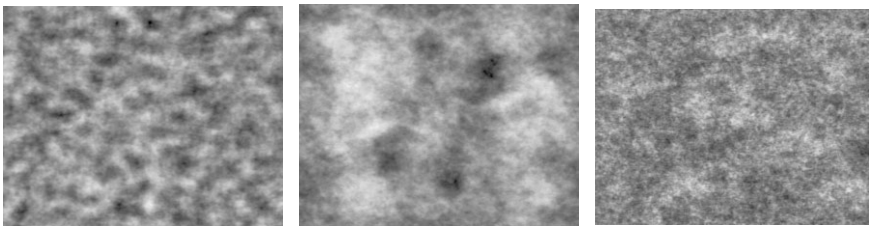

Fig. 12.1 Illustrations of zero-mean Gaussian random fields. Left: Moving average spatial process. Middle: Stationary Gaussian random field on the torus. Right: Gaussian Markov random field

for some *expectation vector* μ and *covariance matrix* Σ. Hence, any linear combination $\sum_{i=1}^{n} b_i \tilde{X}_{t_i}$ has a normal distribution. A Gaussian process is determined completely by its *expectation function* $\tilde{\mu}_t = \mathbf{E}\tilde{X}_t$ and *covariance function* $\tilde{\Sigma}_{s,t} = \mathbf{cov}(\tilde{X}_s, \tilde{X}_t)$. To generate a Gaussian process with expectation function $\tilde{\mu}_t$ and covariance function $\tilde{\Sigma}_{s,t}$ at positions $t_1, \ldots, t_n$, one can sample the multivariate normal random vector in (12.1) via the following algorithm.

Algorithm 12.1 (Gaussian process generator).
1. Construct the mean vector $\mu = (\mu_1, \ldots, \mu_n)^\intercal$ and covariance matrix $\Sigma = (\Sigma_{ij})$ by setting $\mu_i = \tilde{\mu}_{t_i}, i = 1, \ldots, n$ and $\Sigma_{ij} = \tilde{\Sigma}_{t_i,t_j}$, $i, j = 1, \ldots, n$.
2. Find a square root A of Σ, so that $\Sigma = AA^\intercal$.
3. Generate independent random variables $Z_1, \ldots, Z_n \sim N(0,1)$. Let $Z = (Z_1, \ldots, Z_n)^\intercal$.
4. Output $X = \mu + AZ$.

Using Cholesky's square-root method, it is always possible to find a real-valued matrix A such that $\Sigma = AA^\intercal$. Sometimes it is easier to work with a decomposition of the form $\Sigma = BB^*$, where $B = B_1 + \mathrm{i}B_2$ is a complex matrix with conjugate transpose $B^* = B_1^\intercal - \mathrm{i}B_2^\intercal$. Let $Z = Z_1 + \mathrm{i}Z_2$, where Z_1 and Z_2 are independent standard normal random vectors, as in Step 3 above. Then, the random vector $X = \Re(BZ) = B_1 Z_1 - B_2 Z_2$ has covariance matrix Σ.

A Gaussian vector $X \sim N(\mu, \Sigma)$ with invertible covariance matrix Σ can also be simulated using its *precision matrix* $\Lambda = \Sigma^{-1}$. Let $Z = DY$, where $Z \sim_{\mathrm{iid}} N(0,1)$ and $DD^\intercal$ is the Cholesky factorization of Λ. Then Y is a zero-mean multivariate normal vector with covariance matrix

$$\mathbf{E}\,(YY^\intercal) = D^{-1}\mathbf{E}\,(ZZ^\intercal)\,(D^{-1})^\intercal = (D^\intercal D)^{-1} = (\Lambda^{-1})^\intercal = \Sigma\,.$$

The following algorithm describes how a Gaussian process can be generated using a precision matrix.

Algorithm 12.2 (Gaussian process generator using a precision matrix).
1. Derive the Cholesky decomposition $\Lambda = DD^\intercal$ of the precision matrix.
2. Draw independent random variables $Z_1, \ldots, Z_n \sim_{\mathrm{iid}} N(0,1)$. Let $Z = (Z_1, \ldots, Z_n)^\intercal$.
3. Solve Y from $Z = DY$, using forward substitution.
4. Output $X = \mu + Y$.

The Cholesky decomposition of a general $n \times n$ covariance or precision matrix takes $\mathcal{O}(n^3)$ floating point operations. The simulation of high-dimensional Gaussian vectors becomes very time-consuming for large n, unless some extra structure is introduced. In some cases the Cholesky decomposition can be altogether avoided by utilizing the fact that any Gaussian vector can be written as an *affine transformation* $X = \mu + AZ$ of a "white noise" vector $Z \sim N(o, I)$; as in Step 4 of Algorithm 12.1, where o is the n-dimensional zero vector and I the n-dimensional identity matrix. An example where such a transformation can be carried out directly is the following *moving average* Gaussian process $X = \{X_t, t \in \mathcal{T}\}$, where $\mathcal{T}$ is a two-dimensional grid of equally-spaced points. Here each X_t is equal to the average of all white noise terms Z_s with s lying in a disc of radius r around t. That is,

$$X_t = \frac{1}{N_r} \sum_{s\,:\,\|t-s\| \le r} Z_s \,,$$

where N_r is the number of grid points in the disc. Such spatial processes have been used to describe rough energy surfaces for charge transport [51, 223]. The following MATLAB program produces a realization of this process on a 200×200 grid, using a radius $r = 6$. A typical outcome is depicted in the left pane of Fig. 12.1.

```
n = 300;
r = 6; % radius (maximal 49)
noise =  randn(n);
[x,y]=meshgrid(-r:r,-r:r);
mask=((x.^2+y.^2)<=r^2);  %(2*r+1)x(2*r+1) bit mask
x = zeros(n,n);
nmin = 50; nmax = 250;
for i=nmin:nmax
    for j=nmin:nmax
        A = noise((i-r):(i+r), (j-r):(j+r));
        x(i,j) = sum(sum(A.*mask));
    end
end
Nr = sum(sum(mask)); x = x(nmin:nmax, nmin:nmax)/Nr;
imagesc(x); colormap(gray)
```

12.2.2 Generating Stationary Processes via Circulant Embedding

Another approach to efficiently generate Gaussian spatial processes is to exploit the structural properties of stationary Gaussian processes. A Gaussian process $\{\tilde{X}_t, t \in \mathbb{R}^d\}$ is said to be *stationary* if the expectation function, $\mathbf{E}\tilde{X}_t$, is constant and the covariance function, $\mathrm{cov}(\tilde{X}_s, \tilde{X}_t)$, is invariant under translations; that is $\mathbf{cov}(\tilde{X}_{s+u}, \tilde{X}_{t+u}) = \mathbf{cov}(\tilde{X}_s, \tilde{X}_t)$.

An illustrative example is the simulation of a stationary Gaussian process on the unit *torus*; that is, the unit square $[0,1] \times [0,1]$ in which points on opposite sides are identified with each other. In particular, we wish to generate a zero-mean Gaussian random process $\{\tilde{X}_t\}$ on each of the grid points $\{(i,j)/n, i = 0,\ldots,n-1, j = 0,\ldots,n-1\}$ corresponding to a covariance function of the form

$$\mathbf{cov}(\tilde{X}_s, \tilde{X}_t) = \exp\{-c\,\|s-t\|_T^{\alpha}\}, \tag{12.2}$$

where $\|s-t\|_T = \|(s_1-t_1, s_2-t_2)\|_T \stackrel{\text{def}}{=} \sqrt{\sum_{k=1}^{2}(\min\{|s_k-t_k|, 1-|s_k-t_k|\})^2}$ is the Euclidean distance between $s = (s_1,s_2)$ and $t = (t_1,t_2)$ on the torus. Notice that this renders the process not only stationary but also *isotropic* (that is, the distribution remains the same under rotations).

We can arrange the grid points in the order

$$(0,0),(0,1/n),\ldots,(0,1-1/n),(1/n,0),\ldots,(1-1/n,1-1/n).$$

The values of the Gaussian process can be accordingly gathered in an $n^2 \times 1$ vector X or, alternatively an $n \times n$ matrix $\underline{X}$. Let Σ be the $n^2 \times n^2$ covariance matrix of X. The key to efficient simulation of X is that Σ is a symmetric *block-circulant matrix with circulant blocks*. That is, Σ has the form

$$\Sigma = \begin{pmatrix} C_1 & C_2 & C_3 & \ldots & C_n \\ C_n & C_1 & C_2 & \ldots & C_{n-1} \\ & & \ldots & & \\ C_2 & C_3 & \ldots & C_n & C_1 \end{pmatrix},$$

where each C_i is a circulant matrix $\mathrm{circ}(c_{i1},\ldots,c_{in})$. The matrix Σ is thus completely specified by its first row, which we gather in an $n \times n$ matrix G. The eigenstructure of block-circulant matrices with circulant blocks is well known; see, for example, [30]. Specifically, Σ is of the form

$$\Sigma = P^* \mathrm{diag}(\gamma) P, \tag{12.3}$$

where P^* denotes the complex conjugate transpose of P, and P is the Kronecker product of two *discrete Fourier transform* matrices; that is, $P = F \otimes F$, where $F_{jk} = \exp(-2\pi \mathrm{i} jk/n)/\sqrt{n}, j,k = 0,1,\ldots,n-1$. The vector of eigenvalues $\gamma = (\gamma_1,\ldots,\gamma_{n^2})^{\mathsf{T}}$ ordered as an $n \times n$ matrix Γ satisfies $\Gamma = nF^*GF$. Since Σ is a covariance matrix, the component-wise square root $\sqrt{\Gamma}$ is well-defined and real-valued. The matrix $B = P^* \mathrm{diag}(\sqrt{\gamma})$ is a complex square root of Σ, so that X can be generated by drawing $Z = Z_1 + \mathrm{i}Z_2$, where Z_1 and Z_2 are independent standard normal random vectors, and returning the real part of BZ. It will be convenient to gather Z into an $n \times n$ matrix $\underline{Z}$.

The evaluation of both Γ and $\underline{X}$ can be done efficiently by using the (appropriately scaled) *two-dimensional fast Fourier transform* (FFT2). In particular, Γ is the FFT2 of the matrix G and $\underline{X}$ is the real part of the FFT2 of the matrix $\sqrt{\Gamma} \odot \underline{Z}$, where $\odot$ denotes component-wise multiplication. The following MATLAB program

generates the outcome of a stationary Gaussian random field on a 256×256 grid, for a covariance function of the form (12.2), with $c = 8$ and $\alpha = 1$. A realization is shown in the middle pane of Fig. 12.1.

```
n = 2^8;
t1 = [0:1/n:1-1/n]; t2 = t1;
for i=1:n   % first row of cov. matrix, arranged in a matrix
  for j=1:n
       G(i,j)=exp(-8*sqrt(min(abs(t1(1)-t1(i)), ...
       1-abs(t1(1)-t1(i)))^2 + min(abs(t2(1)-t2(j)), ...
       1-abs(t2(1)-t2(j)))^2));
  end;
end;
Gamma = fft2(G);   % the eigenvalue matrix n*fft2(G/n)
Z = randn(n,n) + sqrt(-1)*randn(n,n);
X = real(fft2(sqrt(Gamma).*Z/n));
imagesc(X); colormap(gray)
```

The simulation of general stationary Gaussian processes in $\mathbb{R}^d$ with covariance function

$$\mathbf{cov}(\tilde{X}_s, \tilde{X}_t) = \rho(s-t)$$

is discussed in [100] and [65, 418]. Recall that if $\rho(s-t) = \rho(\|s-t\|)$, then the random field is not only stationary, but isotropic.

In [100], the method of *circulant embedding* is proposed, which allows the efficient simulation of a stationary Gaussian field via the FFT. The idea is to embed the covariance matrix into a block circulant matrix with each block being circulant itself (as in the last example), and then construct the matrix square root of the block circulant matrix using FFT techniques. The FFT permits the fast simulation of the Gaussian field with this block circulant covariance matrix. Finally, the marginal distributions of appropriate sub-blocks of this Gaussian field have the desired covariance structure.

Here we consider the two-dimensional case. The aim is to generate a zero-mean stationary Gaussian field over the $n \times m$ rectangular grid

$$\mathcal{G} = \{(i\Delta_x, j\Delta_y),\ i = 0,\dots,n-1,\ j = 0,\dots,m-1\},$$

where Δ_x and Δ_y denote the corresponding horizontal and vertical spacing along the grid. The algorithm can broadly be described as follows.

Step 1 (Building and storing the covariance matrix). The grid points can be arranged into a column vector of size mn to yield the $mn \times mn$ covariance matrix $\Omega_{i,j} = \rho(s_i - s_j)$, $i, j = 1,\dots,mn$, where

$$s_k \stackrel{\text{def}}{=} \left((k-1) \mod m,\quad \left\lfloor \frac{k}{m} \right\rfloor \right),\quad k = 1,\dots,mn\,.$$

The matrix Ω has symmetric block-Toeplitz structure, where each block is a Toeplitz matrix (not necessarily symmetric). For example, the left panel of Fig. 12.2

shows the block-Toeplitz covariance matrix with $m = n = 3$. Each 3×3 block is itself a Toeplitz matrix with entry values coded in color. For instance, we have $\Omega_{2,4} = \Omega_{3,5} = c$. The matrix Ω is thus uniquely characterized by its first block row $(R_1, \ldots, R_n)$, where each of the n blocks is an $m \times m$ Toeplitz matrix. The k-th $m \times m$ Toeplitz matrix consists of the sub-block of Ω with entries

$$\Omega_{i,j}, \quad i = 1, \ldots, m, \; j = (km+1), \ldots, (k+1)m \, .$$

Notice that each block R_k is itself characterized by its first row and column. (Each Toeplitz block R_k will be characterized by the first row only provided the covariance function has the form $\rho(s-t) = \rho(\|s\|, \|t\|)$, in which case each R_k is symmetric.) Thus, in general, the covariance matrix can be completely characterized by the entries of a pair of $m \times n$ and $n \times m$ matrices storing the first columns and rows of all n Toeplitz blocks $(R_1, \ldots, R_n)$. In typical applications, the computation of these two matrices is the most time-consuming step.

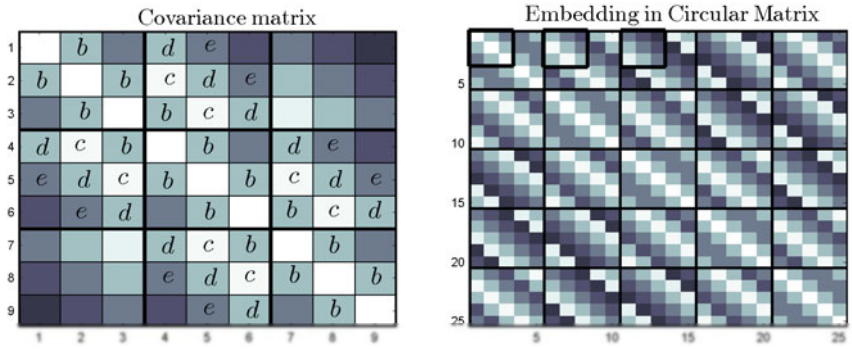

Fig. 12.2 Left panel: the symmetric block-Toeplitz covariance matrix Ω with $n = m = 3$. Right panel: block-circulant matrix Σ of size $((2n-1) \times (2m-1))^2 = 25 \times 25$. The first three block rows of Ω are embedded in the upper left corners of the first three block columns of the circulant matrix Σ

Step 2 (Embedding in block circulant matrix). Each Toeplitz matrix R_k is embedded in the upper left corner of an $2m+1$ circulant matrix C_k. For example, in the right panel of Fig. 12.2 the embedded blocks R_1, R_2, R_3 are shown within bold rectangles. Finally, let Σ be the $(2n-1)(2m-1) \times (2n-1)(2m-1)$ block circulant matrix with first block row given by $(C_1, \ldots, C_n, C_n^{\mathsf{T}}, \ldots, C_2^{\mathsf{T}})$. This gives the *minimal embedding* in the sense that there is no block circulant matrix of smaller size that embeds Ω.

Step 3 (Computing the square root of the block circulant matrix). After the embedding we are essentially generating a Gaussian process on a torus with covariance matrix Σ as in the last example. The block circulant matrix Σ can be diagonalized as in (12.3) to yield $\Sigma = P^* \mathrm{diag}(\gamma) P$, where P is the $(2n-1)(2m-1) \times (2n-1)(2m-1)$ two-dimensional discrete Fourier transform matrix. The vector of

eigenvalues γ is of length $(2n-1)(2m-1)$ and is arranged in a $(2m-1)\times(2n-1)$ matrix Γ so that the first column of Γ consists of the first $2m-1$ entries of γ, the second column of Γ consists of the next $2m-1$ entries and so on. It follows that if G is an $(2m-1)\times(2n-1)$ matrix storing the entries of the first block row of Σ, then Γ is the FFT2 of G. Assuming that $\gamma > 0$, we obtain the square root factor $B = P^* \mathrm{diag}(\sqrt{\gamma})$, so that $\Sigma = B^* B$.

Step 4 (Extracting the appropriate sub-block). Next, we compute the FFT2 of the array $\sqrt{\Gamma} \odot \underline{Z}$, where the square root is applied component-wise to Γ and $\underline{Z}$ is an $(2m-1)\times(2n-1)$ complex Gaussian matrix with entries $\underline{Z}_{j,k} = U_{j,k} + \mathrm{i}V_{j,k}$, $U_{j,k}, V_{j,k} \sim N(0,1)$ for all j and k. Finally, the first $m \times n$ sub-blocks of the real and imaginary parts of $\mathrm{FFT2}(\sqrt{\Gamma} \odot \underline{Z})$ represent two independent realization of a stationary Gaussian field with covariance Σ on the grid $\mathcal{G}$. If more realizations are required, we store the values $\sqrt{\Gamma}$ and we repeat Step 4 only. The complexity of the circulant embedding inherits the complexity of the FFT approach, which is of order $\mathcal{O}(mn\log(m+n))$, and compares very favorably with the standard Cholesky decomposition method of order $\mathcal{O}(m^3 n^3)$.

As a numerical example (see [100]), Fig. 12.3 shows a realization of a stationary nonisotropic Gaussian field with $m = 512, n = 384$, $\Delta_x = \Delta_y = 1$, and covariance function

$$\rho(s-t) = \rho(h) = \left(1 - \frac{h_1^2}{50^2} - \frac{h_1 h_2}{50 \times 15} - \frac{h_2^2}{15^2}\right) \exp\left(-\frac{h_1^2}{50^2} - \frac{h_2^2}{15^2}\right). \quad (12.4)$$

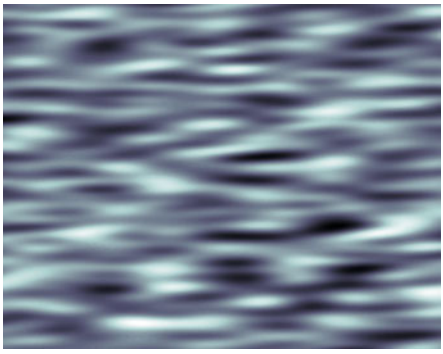

Fig. 12.3 Realization of a stationary nonisotropic Gaussian field with covariance function given by (12.4)

The following MATLAB code implements the procedure described above with covariance function given by (12.4).

```
n=384; m=512; % size of grid is m*n
% size of covariance matrix is m^2*n^2
tx=[0:n-1]; ty=[0:m-1]; % create grid for field
```

```
rho=@(x,y)((1-x^2/50^2-x*y/(15*50)-y^2/15^2)...
      *exp(-(x^2/50^2+y^2/15^2)));
Rows=zeros(m,n); Cols=Rows;
for i=1:n
    for j=1:m
        Rows(j,i)=rho(tx(i)-tx(1),ty(j)-ty(1)); % rows of blocks
        Cols(j,i)=rho(tx(1)-tx(i),ty(j)-ty(1)); % columns
    end
end
% create the first row of the block circulant matrix with
% circulant blocks and store it as a matrix suitable for fft2
BlkCirc_row=[Rows, Cols(:,end:-1:2);
    Cols(end:-1:2,:), Rows(end:-1:2,end:-1:2)];
% compute eigen-values
lam=real(fft2(BlkCirc_row))/(2*m-1)/(2*n-1);

if abs(min(lam(lam(:)<0)))>10^-15
    error('Could not find positive definite embedding!')
else
    lam(lam(:)<0)=0; lam=sqrt(lam);
end
% generate field with covariance given by block circulant matrix
F=fft2(lam.*complex(randn(2*m-1,2*n-1),randn(2*m-1,2*n-1)));
F=F(1:m,1:n); % extract sub-block with desired covariance
field1=real(F); field2=imag(F); % two independent fields
imagesc(tx,ty,field1), colormap bone
```

Extensions to three and more dimensions are possible, see [100]. For example, in the three-dimensional case the correlation matrix Ω will be symmetric block Toeplitz matrix with each block satisfying the properties of a two-dimensional covariance matrix.

Throughout the discussion so far we have always assumed that the block circulant matrix Σ is a covariance matrix itself. If this is the case, then we say that we have a *nonnegative definite embedding* of Ω. A nonnegative definite embedding ensures that the square root of γ is real. Generally, if the correlation between points on the grid that are sufficiently far apart is zero, then a non-negative embedding will exist, see [100]. A method that exploits this observation is Stein's *intrinsic embedding* method proposed in [377] (see also [143]). Stein's method depends on the construction of a compactly supported covariance function that yields to a nonnegative circulant embedding. The idea is to modify the original covariance function so that it decays smoothly to zero. In more detail, suppose we wish to simulate a process with covariance function ρ over the set $\{h : \|h\| \leq 1, h > 0\}$. To achieve this, we simulate a process with covariance function

$$\psi(h) = \begin{cases} c_0 + c_2\|h\|^2 + \rho(h), & \text{if } \|h\| \leq 1, \\ \varphi(h), & \text{if } 1 \leq \|h\| \leq R, \\ 0, & \text{if } \|h\| \geq R, \end{cases} \tag{12.5}$$

where the constants c_0, c_2, $R \geq 1$ and function φ are selected so that ψ is a continuous (and as many times differentiable as possible), stationary and isotropic covariance function on $\mathbb{R}^2$. The process will have covariance structure of $c_0 + c_2\|h\|^2 + \rho(h)$ in the disk $\{h : \|h\| \leq 1, h > 0\}$, which can then be easily transformed into a process with covariance function $\rho(h)$. We give an example of this in Sect. 12.4.4, where we generate fractional Brownian surfaces via the intrinsic embedding technique. Generally, the smoother the original covariance function, the harder it is to embed via Stein's method, because a covariance function that is smoother close to the origin has to be even smoother elsewhere.

12.2.3 Markov Property

A *Markov random field* is a spatial stochastic process $\{X_t, t \in \mathcal{T}\}$ that possesses a *Markov property*, in the sense that

$$(X_t \,|\, X_s, s \in \mathcal{T} \setminus \{t\}) \sim (X_t \,|\, X_s, s \in \mathcal{N}_t),$$

where $\mathcal{N}_t$ is the set of "neighbors" of t. Thus, for each $t \in \mathcal{T}$ the conditional distribution of X_t given all other values X_s is equal to the conditional distribution of X_t given only the neighboring values.

Assuming that the index set $\mathcal{T}$ is finite, Markov random fields are often defined via an undirected graph $\mathcal{G} = (V, E)$. In such a *graphical model* the vertices of the graph correspond to the indices $t \in \mathcal{T}$ of the random field, and the edges describe the dependencies between the random variables. In particular, there is *no* edge between nodes s and t in the graph if and only if X_s and X_t are conditionally independent, given all other values $\{X_u, u \neq i, j\}$. In a Markov random field described by a graph $\mathcal{G}$, the set of neighbors $\mathcal{N}_t$ of t corresponds to the set of vertices that share an edge with t; that is, those vertices that are *adjacent* to t. An example of a graphical model for a 2-dimensional Markov random field is shown in Fig. 12.4. In this case vertex corner nodes have two neighbors and interior nodes have four neighbors.

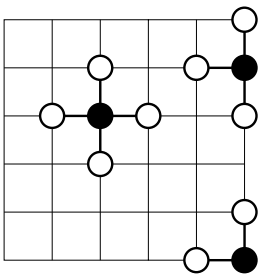

Fig. 12.4 A graphical model for 2D spatial process. Each vertex has at most four neighbors.

Of particular importance are Markov random fields that are also Gaussian. Suppose $\{X_t, t \in \mathcal{T}\}$ is such a *Gaussian Markov random field* (GMRF), with corresponding graphical model $\mathcal{G} = (V, E)$. We may think of $\{X_t, t \in \mathcal{T}\}$ as a column vector of $n = |V|$ elements, or as a spatial arrangement of random variables (pixel values), as in Fig. 12.4. Without loss of generality we assume that $\mathbf{E}X_t = 0$ for each $t \in \mathcal{T}$, and that the index set $\mathcal{T}$ is identified with the vertex set V, whose elements are labeled as $1, 2, \ldots, n$. Because $\{X_t, t \in \mathcal{T}\}$ is Gaussian, the probability density function of $\{X_t, t \in \mathcal{T}\}$ is given by

$$f(x) = (2\pi)^{-n/2} \sqrt{\det(\Lambda)}\, \mathrm{e}^{-\frac{1}{2} x^{\mathsf{T}} \Lambda x} ,$$

where $\Lambda = (\lambda_{ij})$ is the precision matrix. In particular, the joint distribution of two components X_i and X_j is

$$\tilde{f}(x_i, x_j) \propto \exp\left(-\frac{1}{2} (\lambda_{ii} x_i^2 + \lambda_{ij} x_i x_j + \lambda_{jj} x_j^2) \right) .$$

This shows that X_i and X_j are conditionally independent given $\{X_k, k \neq i, j\}$, if and only if $\lambda_{ij} = 0$. Consequently, (i, j) is an edge in the graphical model if and only if $\lambda_{ij} \neq 0$. In typical applications (for example in image analysis) each vertex in the graphical model only has a small number of adjacent vertices, as in Fig. 12.4. In such cases the precision matrix is thus *sparse*, and the Gaussian vector can be generated efficiently using, for example, sparse Cholesky factorization [146].

As an illustrative example, consider the graphical model of Fig. 12.4 on a grid of $n = m^2$ points: $\{1, \ldots, m\} \times \{1, \ldots, m\}$. We can efficiently generate a GMRF on this grid via Algorithm 12.2, provided the Cholesky decomposition can be carried out efficiently. For this we can use for example the *band Cholesky* method [146], which takes $n(p^2 + 3p)$ floating point operations, where p is the bandwidth of the matrix; that is, $p = \max_{i,j}\{|i - j| : \lambda_{ij} = 0\}$. The right pane of Fig. 12.1 shows a realization of the GMRF on a 250×250 grid, using parameters $\lambda_{ii} = 2$ for all $i = 1, \ldots, n$ and $\lambda_{ij} = -0.5$ for all neighboring elements $j \in \mathcal{N}_j$ of $i = 1, \ldots, n$. Further details on such construction methods for GMRFs may be found, for example, in [336].

Exercise 12.1. Write MATLAB code to generate a zero-mean Gaussian Markov random field for a 200×200 grid with a neighborhood structure similar to Fig. 12.4, that is, each internal vertex has four neighbors, the boundary ones have three, and the corner ones have two. Choose $\lambda_{ii} = 1$ and $\lambda_{ij} = -0.25$ for each neighboring element j of i. *Hint.* See the MATLAB code in Sect. 5.1 of [237].

12.3 Point Processes

Point processes on $\mathbb{R}^d$ are spatial processes describing random configurations of d-dimensional points. Spatially distributed point patterns occur frequently in nature and in a wide variety of scientific disciplines, such as spatial epidemiology,

materials science, forestry, and geography. The positions of accidents on a highway during a fixed time period and the times of earthquakes in Japan are examples of one-dimensional spatial point processes. Two-dimensional examples include the positions of cities on a map, the positions of farms with Mad Cow Disease in the UK, and the positions of broken connections in a communications or energy network. In three dimensions, we observe the positions of stars in the universe, the positions of mineral deposits underground, or the times and positions of earthquakes in Japan. Spatial processes provide excellent models for many of these point patterns [15, 16, 93, 94, 102]. Spatial processes also have an important role in stochastic modeling of complex microstructures, for example, graphite electrodes used in Lithium-ion batteries [380].

Mathematically, point processes can be described in three ways: 1. as random sets of points, 2. as random-sized vectors of random positions, and 3. as random counting measures. In this section we discuss some of the important classes of point processes and their generalizations, including Poisson processes, marked point processes, and cluster processes.

12.3.1 Poisson Process

Poisson processes are used to model random configurations of points in space and time. Let E be some Borel subset of $\mathbb{R}^d$ and let $\mathcal{E}$ be the collection of Borel sets on E. To any collection of random points $\{X_1,\ldots,X_N\}$ in E corresponds a *random counting measure* $X(A), A \in \mathcal{E}$ defined by

$$X(A) = \sum_{i=1}^{N} \mathbf{1}_{\{X_i \in A\}}, \quad A \in \mathcal{E}\,, \tag{12.6}$$

which counts the random number of points in A. We may *identify* the random measure X defined in (12.6) with the random set $\{X_i, i = 1,\ldots,N\}$, or with the random vector $(X_1,\ldots,X_N)$. The measure $\mu : \mathcal{E} \to [0,\infty]$ given by $\mu(A) = \mathbf{E}X(A), A \in \mathcal{E}$ is called the *mean measure* of X. In many cases the mean measure μ has a density $\lambda : E \to [0,\infty)$, called the *intensity function*; so that

$$\mu(A) = \mathbf{E}X(A) = \int_A \lambda(x)\,\mathrm{d}x\,.$$

We will assume from now on that such an intensity function exists.

The most important class of point processes which holds the key to the analysis of point pattern data is the Poisson process. A random counting measure X is said to be a *Poisson random measure* with locally finite mean measure μ if the following properties hold:

1. For any bounded set $A \in \mathcal{E}$ the random variable $X(A)$ has a Poisson distribution with mean $\mu(A)$. We write $X(A) \sim \mathrm{Pois}(\mu(A))$.

2. For any disjoint sets $A_1,\ldots,A_N \in \mathcal{E}$, the random variables $X(A_1),\ldots,X(A_N)$ are independent.

The Poisson process is said to be *stationary* if the intensity function is constant. When the intensity function is a constant, we simply refer to it as the intensity. An important corollary of Properties 1 and 2 is the following result. Suppose that $0 < \mu(E) < \infty$. Then,

3. conditional upon $X(E) = N$, the points $X_1,\ldots,X_N$ are independent of each other and have the probability density function (pdf) $g(x) = \lambda(x)/\mu(E)$.

This result is the key to generating a Poisson random measure on $E \subset \mathbb{R}^d$ with finite mean measure $\mu(E) < \infty$.

Algorithm 12.3 (Generating a Poisson random measure).
1. Generate a Poisson random variable $N \sim \text{Pois}(\mu(E))$.
2. Draw $X_1,\ldots,X_N \sim g$, where $g(x) = \lambda(x)/\mu(E)$, and return these as the points of the Poisson random measure.

As a specific example, consider the simulation of a 2-dimensional Poisson process with intensity function $\lambda(x_1,x_2) = 300(x_1^2+x_2^2)$ on the unit square $E = [0,1]^2$. Since the probability density function $g(x_1,x_2) = \lambda(x_1,x_2)/\mu(E) = 3(x_1^2+x_2^2)/2$ is bounded by 3, drawing from g can be done simply via the acceptance–rejection method [334].

Exercise 12.2. Write MATLAB code that implements this acceptance-rejection method. That is, in Step 2 of Algorithm 12.3, draw (X_1,X_2) uniformly on E and Z uniformly on $[0,3]$, and accept (X_1,X_2) if $g(X_1,X_2) \le Z$; otherwise repeat.

An alternative, but equivalent, method is to generate a stationary Poisson process on E, with intensity $\lambda = 600$, and to *thin out* the points by accepting each point (x_1,x_2) with probability $\lambda(x_1,x_2)/\lambda$. The following MATLAB code implements this thinning procedure. A typical realization is given in Fig. 12.5.

```
lambda = @(x) 300*(x(:,1).^2 + x(:,2).^2);
lamstar = 600;
N=poissrnd(lamstar); x = rand(N,2); % homogeneous PP
ind = find(rand(N,1) < lambda(x)/lamstar);
xa = x(ind,:); % thinned PP
plot(xa(:,1),xa(:,2))
```

12.3.2 Marked Point Processes

A natural way to extend the notion of a point process is to associate with each point $X_i \in \mathbb{R}^d$ a (random) *mark* $Y_i \in \mathbb{R}^m$, representing an attribute such as width, velocity, weight etc. The collection $\{(X_i,Y_i)\}$ is called a *marked point process*. In

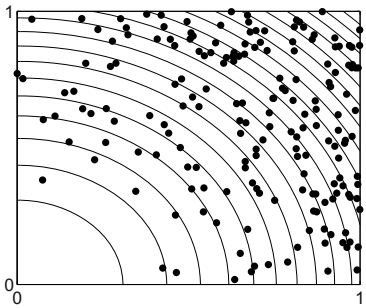

Fig. 12.5 A realization of a non-stationary Poisson process with intensity function $\lambda(x_1,x_2) = 300(x_1^2+x_2^2)$ (contour lines shown) on the unit square

a marked point process with *independent marking* the marks are assumed to be independent of each other and of the points, and distributed according to a fixed mark distribution. The following gives a useful connection between marked point processes and Poisson processes; see, for example, [93, 94].

Theorem 12.1. *If* $\{(X_i,Y_i)\}$ *is a Poisson process on* $\mathbb{R}^d \times \mathbb{R}^m$ *with intensity function* $\zeta : \mathbb{R}^d \times \mathbb{R}^m \to [0,\infty)$*, and*

$$K(x) = \int \zeta(x,y)\,dy < \infty \quad \text{for all } x \in \mathbb{R}^d, \tag{12.7}$$

then $\{X_i\}$ *is a Poisson process on* $\mathbb{R}^d$ *with intensity function* $K : \mathbb{R}^d \to [0,\infty)$ *given in* (12.7)*, and* $\{(X_i,Y_i)\}$ *is a marked Poisson process, where the density function of the marks is given by* $\zeta(x,\cdot)/K(x)$ *on* $\mathbb{R}^m$*.*

A (spatial) marked Poisson process with independent marking is an important example of a (spatial) *Lévy process*: a stochastic process with independent and stationary increments (discussed in more detail in Sect. 12.5).

The simulation of a marked Poisson process with independent marks is virtually identical to that of an ordinary (that is, non-marked) Poisson process. The only difference is that for each point the mark has to be drawn from the mark distribution. An example of a realization of a marked Poisson process is given in Fig. 12.6. Here the marks are uniformly distributed on [0,0.1], and the underlying Poisson process is stationary with intensity 100.

12.3.3 Cluster Process

In applications one often observes point patterns that display clustering. An example is the spread of plants from a weed species where the plants are initially introduced

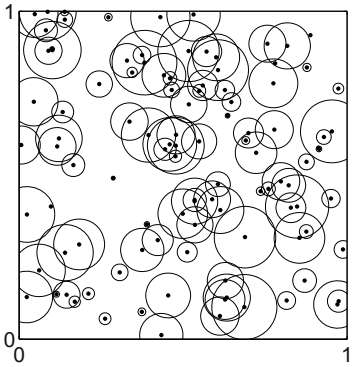

Fig. 12.6 A realization of a marked Poisson process with independent marking on the unit square. The Poisson process has intensity 100. The marks are uniformly distributed on [0,0.1] and correspond to the radii of the circles

by birds at a number of geographically dispersed locations and the plants then spread themselves locally.

Let C be a point process of "centers" and associate with each $c \in C$ a point process X^c, which may include c. The combined set of points $X = \cup_{c \in C} X^c$ constitutes a *cluster process*. If C is a Poisson process, then X is called a *Poisson cluster process*.

As a specific example, consider the following *Hawkes process*. Here, the center process C is a Poisson process on $\mathbb{R}^d$ (or a subset thereof) with some intensity function $\lambda(\cdot)$. The clusters are generated as follows. For each $c \in C$, independently generate "first-generation offspring" according to a Poisson process with intensity function $\rho(x-c)$, where $\rho(\cdot)$ is a positive function on $\mathbb{R}^d$ with integral less than unity. Then, for each first-generation offspring $c^{(1)}$ generate a Poisson process with intensity function $\rho(x-c^{(1)})$, and so on. The collection of all generated points forms the Hawkes process. The requirement that $\int \rho(y)\,\mathrm{d}y < 1$ simply means that the expected number of offspring of each point is less than one.

Fig. 12.7 displays a realization for $\mathbb{R}^2$ with the cluster centers forming a Poisson process on the unit square with intensity $\lambda = 30$. The offspring intensity function is here

$$\rho(x_1,x_2) = \frac{\alpha}{2\pi\sigma^2}\,\mathrm{e}^{-\frac{1}{2\sigma^2}(x_1^2+x_2^2)}, \quad (x_1,x_2) \in \mathbb{R}^2, \tag{12.8}$$

with $\alpha = 0.9$ and with $\sigma = 0.02$. This means that the number N of offspring for a single generation from a parent at position (x_1,x_2) has a Poisson distribution with parameter α. And given $N = n$, the offspring locations are independent and identically distributed according to a bivariate normal random vector with independent components with means x_1 and x_2 and both variances σ^2. In Fig. 12.7 the cluster centers are indicated by circles. Note that the process possibly has points outside the displayed box. The MATLAB code is given below.

```
lambda = 30;           %intensity of initial points (centers)
mean_offspring = 0.9;   %mean number of offspring of each point
X = zeros(10^5,2);     %initialise the points
N = poissrnd(lambda);  %number of centers
X(1:N,:) = rand(N,2);  %generate the centers
total_so_far = N;      %total number of points generated
next = 1;
while next < total_so_far
    nextX = X(next,:);   %select next point
    N_offspring = poissrnd(mean_offspring); %number of offspring
    NewX=repmat(nextX,N_offspring,1)+0.02*randn(N_offspring,2);
    X(total_so_far+(1:N_offspring),:) = NewX; %update point list
    total_so_far = total_so_far+N_offspring;
    next = next+1;
end
X=X(1:total_so_far,:); %cut off unused rows
plot(X(:,1),X(:,2),'.')
```

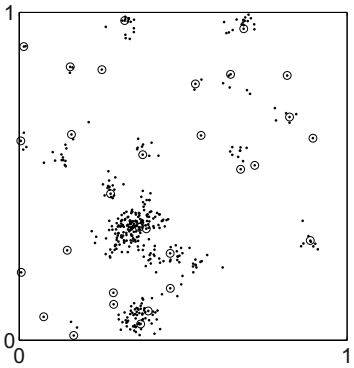

Fig. 12.7 Realization of a two-dimensional Hawkes process with centers (encircled) forming a Poisson process on $[0,1] \times [0,1]$ with intensity $\lambda = 30$. The offspring intensity function is given in (12.8)

12.3.4 Cox Process

Suppose we wish to model a point pattern of trees given that we know the soil quality $h(x_1, x_2)$ at each point (x_1, x_2). We could use a non-stationary Poisson process with an intensity function λ that is an increasing function of h; for example, $\lambda(x_1, x_2) = \mathrm{e}^{\alpha+\beta h(x_1,x_2)}$ for some known $\alpha > 0$ and $\beta > 0$. In practice, however, the soil quality itself could be random, and be described via a random field. Consequently, one could

could try instead to model the point pattern as a Poisson process with a random intensity function Λ. Such processes were introduced in [85] as doubly stochastic Poisson processes and are now called Cox processes.

More precisely, we say that X is a *Cox process* driven by the random intensity function Λ if conditional on $\Lambda = \lambda$ the point process X is Poisson with intensity function λ. Simulation of a Cox process on a set $\mathcal{T} \subset \mathbb{R}^d$ is thus a two-step procedure.

Algorithm 12.4 (Simulation of a Cox process).
1. Simulate a realization $\lambda = \{\lambda(x), x \in \mathcal{T}\}$ of the random intensity function Λ.
2. Given $\Lambda = \lambda$, simulate X as an non-stationary Poisson process with intensity function λ.

Fig. 12.8 (a) shows a realization of a Cox process on the unit square $\mathcal{T} = [0,1] \times [0,1]$, with a random intensity function whose realization is given in Fig. 12.8 (b). The random intensity at position (x_1, x_2) is either 3000 or 0, depending on whether the value of a random field on $\mathcal{T}$ is negative or not. The random field that we used is the stationary Gaussian process on the torus described in Sect. 12.2.

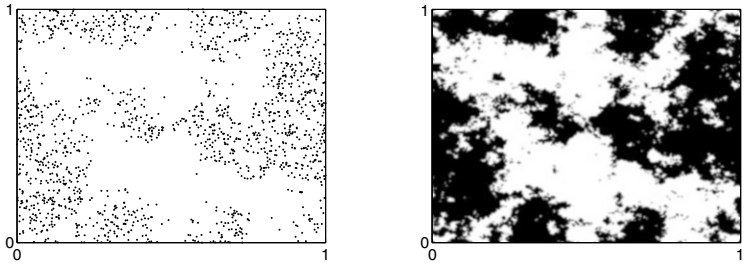

Fig. 12.8 Realization of a Cox process: (a) on a square generated by a the random intensity function given in (b). The black points have intensity 3000 and the white points have intensity 0

Given an outcome of the random intensity function, this Cox process (sometimes called random-set Cox or interrupted Poisson process) is constructed by generating a stationary Poisson process with rate 3000 on $\mathcal{T}$ and accepting only those points for which the random intensity function is non-zero. The values of the intensity function are taken to be constant within each square $\{(i+u)/n, (j+v)/n), 0 \le u, v \le 1\}$, $i, j = 0, \ldots, n-1$. The following MATLAB code is to be appended to the code used for the stationary Gaussian process simulation on the torus in Sect. 12.2.

```
Lambda = ones(n,n).*(X < 0); %random intensity function
%imagesc(Lambda); set(gca,'YDir','normal')
Lambda = Lambda(:); %reshape as a column vector
N = poissrnd(3000);
```

```
P = rand(N,2); %generate homogenous PP
Pn = ceil(P*n); %PP scaled by factor n
K = (Pn(:,1)-1)*n + Pn(:,2); % indices of scaled PP
ind = find(Lambda(K)); %indices for which intensity is 1
Cox = P(ind,:);  %realization of the Cox process
plot(Cox(:,1),Cox(:,2),'.');
```

In [299] the following Cox process has been applied to cosmology; see, however, [178], who show the limitations of the model. Used to model the positions of stars in the universe, it now bears the name *Neyman–Scott process*. Suppose C is a stationary Poisson process in $\mathbb{R}^d$ with constant intensity κ. Let the random intensity function Λ be given by

$$\Lambda(x) = \alpha \sum_{c \in C} k(x-c)$$

for some $\alpha > 0$ and some d-dimensional probability density function (pdf) k. Such a Cox process is also a Poisson cluster process. Note that in this case the cluster centers are not part of the Cox process. Given a realization of the cluster center process C, the cluster of points originating from $c \in C$ form a non-stationary Poisson process with intensity function $k(x-c), x \in \mathbb{R}^d$, independently of the other clusters. Drawing such a cluster via Algorithm 12.3 simply means that (1) the number of points N_c in the cluster has a $\text{Pois}(\alpha)$ distribution, and (2) these N_c points are independent and identically distributed according to the density function $k(x-c)$.

A common choice for k is $k(x) \propto \mathbf{1}_{\{\|x\| \le r\}}$, first proposed in [262]. The resulting process is called a *Matérn process*. Thus, for a Matérn process each point in the cluster with center c is uniformly distributed within a ball of radius r at c. If instead a $N(c, \sigma^2 I)$ distribution is used, where I is the d-dimensional identity matrix, then the process is known as a (modified) *Thomas process* [287].

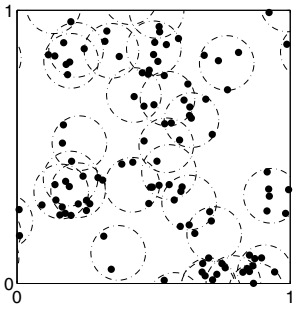

Fig. 12.9 A realization of a Matérn process with parameters $\kappa = 20$, $\alpha = 5$, and $r = 0.1$. The process extends beyond the shown window. The cluster centers (the centers of the circles) are not part of the point pattern

Fig. 12.9 shows a realization of a Matérn process with parameters $\kappa = 20$, $\alpha = 5$ and $r = 0.1$ on $[0,1] \times [0,1]$. Note that the cluster centers are assumed to lie on

the whole of $\mathbb{R}^2$. To show a genuine outcome of the process within the window $[0,1]\times[0,1]$ it suffices to consider only the points that are generated from centers lying in square $[-r,1+r]\times[-r,1+r]$.

The following MATLAB program was used.

```
X = zeros(10^5,2);  %initialise the points
kappa = 20; alpha = 5; r = 0.1; %parameters
meanpts=kappa*(1 + 2*r)^2;
N = poissrnd(meanpts);      %number of cluster centers
C = rand(N,2)*(1+2*r) - r;  ;%draw cluster centers
total_so_far = 0;
for c=1:N
    NC = poissrnd(alpha);  %number of points in cluster
    k = 0;
    while k < NC %draw uniformly in the n-ball via accept-reject
        Y = 2*r*rand(1,2) - r; %candidate point
        if norm(Y) < r
            X(total_so_far+k+1,:) = C(c,:) + Y;
            k = k+1;
        end
    end
    total_so_far = total_so_far + NC;
end
X = X(1:total_so_far,:);    %cut off unused rows
plot(X(:,1),X(:,2),'.')
axis([0, 1,0, 1])
```

A versatile generalization the Neyman–Scott process is the *shot-noise Cox process* [280], where the random intensity function is of the form

$$\Lambda(x) = \sum_{(c_j,\gamma_j)\in Z} \gamma_j k(c_j,x)\,,$$

and $\{(c_j,\gamma_j)\}$ are the points of an non-stationary Poisson process Z on $\mathbb{R}^d\times\mathbb{R}_+$ with intensity function ζ, and k is a *kernel function*; that is, $k(c,\cdot)$ is a probability density function (on $\mathbb{R}^d$) for each $c\in\mathbb{R}^d$. By Theorem 12.1, if there exists a function $K(c):\mathbb{R}^d\to\mathbb{R}_+$ such that

$$K(c) = \int_0^\infty \zeta(c,\gamma)\,\mathrm{d}\gamma < \infty\,,$$

then $C=\{c:(c,\gamma)\in Z\}$ is a Poisson process with intensity function K, and Z is a marked Poisson process with mark density $\zeta(c,\cdot)/K(c)$. This decomposition of the shot-noise Cox process into a marked Poisson process suggests a method for simulation.

Algorithm 12.5 (Simulation of a shot-noise Cox process).
1. Draw the points C of an non-stationary Poisson process with intensity function $K(c)$.

2. For each $c_j \in C$, draw the corresponding mark γ_j from the density $\zeta(c_j,\cdot)/K(c_j)$.
3. For each $c_j \in C$, draw $N_j \sim \text{Pois}(\gamma_j)$.
4. Draw for each $c_j \in C$, N_j points from the kernel $k(c_j,x)$. The collection of all points drawn at this step constitutes a realization from the shot-noise Cox process.

A special case of the shot-noise Cox process, and one that appears frequently in the literature, is the *shot-noise Gamma Cox process*. Here the intensity function ζ is of the form

$$\zeta(c,\gamma) = \beta\gamma^{\alpha-1}\exp(-\lambda\gamma)/\Gamma(1+\alpha)\,,$$

where $\beta > 0, \alpha > 0, \lambda > 0$, and

$$K(c) = \int_0^\infty \zeta(c,\gamma)\,\mathrm{d}\gamma = \beta\lambda^{-\alpha}/\alpha < \infty\,.$$

Hence, Z is a marked Poisson process, where the intensity function of the centers is $K(c) = \beta\lambda^{-\alpha}/\alpha$ and the marks are independently and identically $Gamma(\alpha,\lambda)$ distributed; see [53] and [287] for more information.

12.3.5 Point Process Densities

Let X be a point process with mean measure μ on a bounded region $E \subset \mathbb{R}^d$. We assume that the expected total number of points $\mu(E)$ is finite. We may view X as a random object taking values in the space $\mathscr{X} = \cup_{n=0}^\infty (\{n\} \times E^n)$, where E^n is the Cartesian product of n copies of E. Note that in this representation the coordinates of X are ordered: $X = (N,(X_1,\ldots,X_N))$. We identify each vector (N,X) with X. If, for every set $\{n\} \times A$ with A a measurable set in E^n we can write

$$\mathbf{P}(X \in \{n\} \times A) = \int_A f(x_1,\ldots,x_n)\,\mathrm{d}x_1 \ldots \mathrm{d}x_n\,,$$

then $f(x)$ is the probability density function or simply *density* of X on $\mathscr{X}$ (with respect to the Lebesgue measure on $\mathscr{X}$). Using the identification $(n,x) = x$, we can view $f(x)$ as the joint density of the random variable N and the N-dimensional vector X, where each component of X takes values in E. Using a Bayesian notation convention where all probability density functions and conditional probability density functions are indicated by the same symbol f, we have $f(x) = f(n,x) = f(n)f(x\,|\,n)$, where $f(n)$ is the (discrete) probability density function of the random number of points N, and $f(x\,|\,n)$ is the joint probability density function (pdf) of $X_1,\ldots,X_n$ given $N = n$. As an example, for the Poisson process on E with intensity function $\lambda(x)$ and mean measure μ we have, in correspondence to Algorithm 12.3,

$$f(x) = f(n)\,f(x\,|\,n) = \frac{\mathrm{e}^{-\mu(E)}\{\mu(E)\}^n}{n!}\prod_{i=1}^n \frac{\lambda(x_i)}{\mu(E)} = \frac{\mathrm{e}^{-\mu(E)}}{n(x)!}\prod_{i=1}^{n(x)}\lambda(x_i)\,, \qquad (12.9)$$

where $n(x)$ is the number of components in x. Conversely, the expression for the pdf in (12.9) shows immediately how X can be generated; that is, via Algorithm 12.3.

A general recipe for generating a point process X is thus:

1. draw N from the discrete pdf $f(n)$;
2. given $N = n$, draw $(X_1, \ldots, X_n)$ from the conditional pdf $f(x \mid n)$.

Unfortunately, the pdf $f(n)$ and $f(x \mid n)$ may not be available explicitly. Sometimes $f(x)$ is known up to an unknown normalization constant. In such cases one case use *Markov Chain Monte Carlo* (MCMC) to simulate from $f(x)$. The basic idea of MCMC is to run a Markov chain long enough so that its limiting distribution is close to the target distribution. The most well-known MCMC algorithm is the following; see, for example, [334].

Algorithm 12.6. (Metropolis–Hastings algorithm) Given a *transition density* $q(y \mid x)$, and starting from an initial state X_0, repeat the following steps for $t = 1, 2, \ldots$:

1. Generate a candidate $Y \sim q(y \mid X_t)$.
2. Generate $U \sim \text{Unif}(0,1)$ and set

$$X_{t+1} = \begin{cases} Y, & \text{if } U \leq \alpha(X_t, Y), \\ X_t, & \text{otherwise,} \end{cases} \tag{12.10}$$

where $\alpha(x,y)$ is the *acceptance probability*, given by:

$$\alpha(x,y) = \min\left\{ \frac{f(y)\, q(x \mid y)}{f(x)\, q(y \mid x)}, 1 \right\}. \tag{12.11}$$

This produces a sequence $X_1, X_2, \ldots$ of *dependent* random vectors, with X_t approximately distributed according to $f(x)$, for large t. Since Algorithm 12.6 is of the acceptance–rejection type, its efficiency depends on the acceptance probability $\alpha(x,y)$. Ideally, one would like the proposal transition density $q(y \mid x)$ to reproduce the desired pdf $f(y)$ as faithfully as possible. For a *random walk sampler* the proposal state Y, for a given current state x, is given by $Y = x + Z$, where Z is typically generated from some spherically symmetrical distribution. In that case the proposal transition density pdf is symmetric; that is $q(y \mid x) = q(x \mid y)$. It follows that the acceptance probability is:

$$\alpha(x,y) = \min\left\{ \frac{f(y)}{f(x)}, 1 \right\}. \tag{12.12}$$

As a specific example, suppose we wish to generate a *Strauss process* [222, 384]. This is a point process with density of the form

$$f(x) \propto \beta^{n(x)} \gamma^{s(x)},$$

where $\beta, \gamma \geq 0$ and $s(x)$ is the number of pairs of points where the two points are within distance r of each other. As before, $n(x)$ denotes the number of points. The

process exists (that is, the normalization constant is finite) if $\gamma \leq 1$; otherwise, it does not exist in general [287].

We first consider simulating from $f(x|n)$ for a fixed n. Thus, $f(x|n) \propto \gamma^{s(x)}$, where $x = (x_1, \ldots, x_n)$. The following MATLAB program implements a Metropolis–Hastings algorithm for simulating a (conditional) Strauss process with $n = 200$ points on the unit square $[0,1] \times [0,1]$, using the parameter values $\gamma = 0.1$ and $r = 0.2$. Given a current state $x = (x_1, \ldots, x_n)$, the proposal state $Y = (Y_1, \ldots, Y_n)$ is identical to x except for the J-th component, where J is a uniformly drawn index from the set $\{1, \ldots, n\}$. Specifically, $Y_J = x_J + Z$, where $Z \sim N(0, (0.1)^2)$. The proposal Y for this random walk sampler is accepted with probability $\alpha(x, Y) = \min\{\gamma^{s(Y)}/\gamma^{s(x)}, 1\}$. The function $s(x)$ is implemented below as `numpairs.m`.

```
gam = 0.1;
r = 0.2;
n = 200;
x = rand(n,2); %initial pp
K = 10000;
np= zeros(K,1);
for i=1:K
    J = ceil(n*rand);
    y = x;
    y(J,:) = y(J,:) + 0.1*randn(1,2); %proposal
    if (max(max(y)) > 1 || min(min(y)) <0)
        alpha =0; %don't accept a point outside the region
    elseif (numpairs(y,r) < numpairs(x,r))
        alpha =1;
    else
        alpha = gam^numpairs(y,r)/gam^numpairs(x,r);
    end
    R = (rand < alpha);
    x = R*y + (1-R)*x; %new x-value
    np(i) = numpairs(x,r);
    plot(x(:,1),x(:,2),'.');
    axis([0,1,0,1])
    refresh; pause(0.0001);
end
```

```
function s = numpairs(x,r)
n = size(x,1);
D = zeros(n,n);
for i = 1:n
   D(i,:) = sqrt(sum((x(i*ones(n,1),:) - x).^2,2));
end
D = D + eye(n);
s = numel(find((D < r)))/2;
end
```

A typical realization of the conditional Strauss process is given in the left pane of Fig. 12.10. We see that the $n = 200$ points are clustered together in groups. This

pattern is quite different from a typical realization of the unconditional Strauss process, depicted in the right pane of Fig. 12.10. Not only are there typically far fewer points, but also these points tend to "repel" each other, so that the number of pairs within a distance of r of each other is small. The radius of each circle in the figure is $r/2 = 0.1$. We see that in this case $s(x) = 3$, because 3 circle pairs overlap.

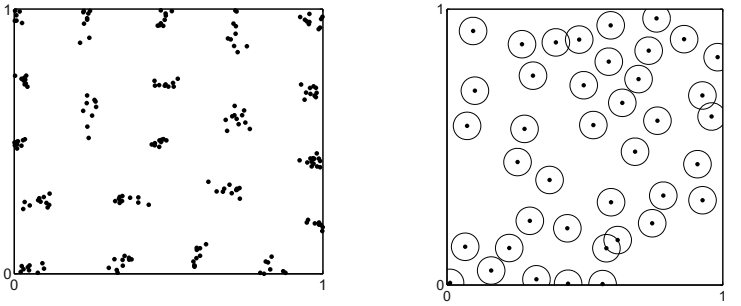

Fig. 12.10 Left: Conditional Strauss process with $n = 200$ points and parameters $\gamma = 0.1$ and $r = 0.2$. Right: Strauss process with $\beta = 100$, $\gamma = 0.1$, and $r = 0.2$

To generate the (unconditional) Strauss process using the Metropolis–Hastings sampler, the sampler needs to be able to "jump" between different dimensions n. The *reversible jump sampler* [150, 334] is an extension of the Metropolis–Hastings algorithm designed for this purpose. Instead of one transition density $q(y \,|\, \mathbf{x})$, it requires a transition density for n, say $q(n \,|\, m)$, and for each n a transition density $q(y \,|\, x, n)$ to jump from x to an n-dimensional vector y.

Given a current state X_t of dimension m, Step 1 of Algorithm 12.6 is replaced with

1a. Generate a candidate dimension $n \sim q(n \,|\, m)$.
1b. Generate an n-dimensional vector $Y \sim q(y \,|\, X_t, n)$.

And in Step 2 the acceptance ratio in (12.11) is replaced with

$$\alpha(x,y) = \min\left\{ \frac{f(n,y)\, q(m \,|\, n)\, q(x \,|\, y,m)}{f(m,x)\, q(n \,|\, m)\, q(y \,|\, x,n)},\, 1 \right\}. \tag{12.13}$$

The MATLAB program below implements a simple version of the reversible jump sampler, suggested in [136]. From a current state x of dimension m, a candidate dimension is chosen to be either $m+1$ or $m-1$, with equal probability. Thus, $q(m+1 \,|\, m) = q(m-1 \,|\, m) = 1/2, m = 1, 2, \ldots$. The first transition corresponds to the birth of a new point; the second to the death of a point. On the occasion of a birth, a candidate Y is generated by simply appending a uniform point on $[0,1]^2$ to x. The corresponding transition density is therefore $q(y \,|\, x, m) = 1$. On the occasion of a death, the candidate Y is obtained from x by removing one of the points of x at random and uniformly. The transition density is thus $q(y \,|\, x, m) = 1/n(x)$. This gives

the acceptance ratios:
1. Birth:

$$\alpha(x,y) = \min\left\{\frac{\beta^{n(y)}\gamma^{s(y)}\frac{1}{n(y)}}{\beta^{n(x)}\gamma^{s(x)}1}, 1\right\} = \min\{(\beta\gamma^{s(y)-s(x)})/n(y), 1\}\,.$$

2. Death:

$$\alpha(x,y) = \min\left\{\frac{\beta^{n(y)}\gamma^{s(y)}1}{\beta^{n(x)}\gamma^{s(x)}\frac{1}{n(x)}}, 1\right\} = \min\{(\gamma^{s(y)-s(x)}n(x))/\beta, 1\}\,.$$

```
r = 0.1; gam = 0.2; beta = 100; %parameters
n = 200; x = rand(n,2); %initial pp
K = 1000;
for i=1:K
    n = size(x,1);
    B = (rand < 0.5);
    if B %birth
        xnew = rand(1,2);
        y = [x;xnew];
        n = n+1;
    else %death
        y = setdiff(x,x(ceil(n*rand),:),'rows');
    end
    if (max(max(y)) > 1 || min(min(y)) <0)
        alpha =0; %don't accept a point outside the region
    elseif (numpairs(y,r) < numpairs(x,r))
        alpha =1;
    elseif B %birth
        alpha = beta*gam^(numpairs(y,r) - numpairs(x,r))/n;
    else  %death
        alpha = n*gam^(numpairs(y,r) - numpairs(x,r))/beta;
    end
    if (rand < alpha)
        x = y;
    end
    plot(x(:,1),x(:,2),'.');
    axis([0,1,0,1]); refresh; pause(0.0001);
```

For more on Strauss processes, see [401].

12.4 Wiener Surfaces

Brownian motion is one of the simplest continuous-time stochastic processes, and as such has found myriad applications in the physical sciences [247]. As a first step toward constructing Brownian motion we introduce the one-dimensional Wiener

process $\{W_t, t \in \mathbb{R}_+\}$, which can be viewed as a spatial process with a continuous index set on $\mathbb{R}_+$ and with a continuous state space $\mathbb{R}$.

12.4.1 Wiener Process

A one-dimensional Wiener process is a stochastic process $\{W_t, t \geq 0\}$ characterized by the following properties:

1. the increments of W_t are stationary and normally distributed, that is, $W_t - W_s \sim N(0, t-s)$ for all $t \geq s \geq 0$;
2. W has independent increments, that is, for any $t_1 < t_2 \leq t_3 < t_4$, the increments $W_{t_4} - W_{t_3}$ and $W_{t_2} - W_{t_1}$ are independent random variables (in other words, $W_t - W_s$, $t > s$ is independent of the past history of $\{W_u,\ 0 \leq u \leq s\}$);
3. continuity of paths, that is, $\{W_t\}$ has continuous paths with $W_0 = 0$.

The simplest algorithm for generating the process uses the Markovian (independent increments) and Gaussian properties of the Wiener process. Let $0 = t_0 < t_1 < t_2 < \cdots < t_n$ be the set of distinct times for which simulation of the process is desired. Then, the Wiener process is generated at times $t_1, \ldots, t_n$ via

$$W_{t_k} = \sum_{i=1}^{k} \sqrt{t_k - t_{k-1}}\, Z_i, \quad k = 1, \ldots, n\,,$$

where $Z_1, \ldots, Z_n \overset{\text{iid}}{\sim} N(0,1)$. To obtain a continuous path approximation to the path of the Wiener process, one could use linear interpolation on the points $W_{t_1}, \ldots, W_{t_n}$. A realization of a Wiener process is given in the middle panel of Fig. 12.11. Given the Wiener process W_t, we can now define the d-dimensional *Brownian motion process* via

$$\tilde{X}_t = \mu\, t + \Sigma^{1/2} W_t, \quad W_t = (W_t^{(1)}, \ldots, W_t^{(d)})^\intercal, \quad t \geq 0\,, \tag{12.14}$$

where $W_t^{(1)}, \ldots, W_t^{(d)}$ are independent Wiener processes and Σ is a $d \times d$ covariance matrix. The parameter $\mu \in \mathbb{R}^d$ is called the *drift* parameter and Σ is called the *diffusion matrix*.

Exercise 12.3. Generate and plot a realization of a three dimensional Wiener process at times $0, 1/n, 2/n, \ldots, 1$ for $n = 10^4$.

One approach to generalizing the Wiener process conceptually or to higher spatial dimensions is to use its characterization as a zero-mean Gaussian process (see Sect. 12.2.1) with continuous sample paths and covariance function $\mathbf{cov}(W_t, W_s) = \min\{t,s\}$ for $t, s > 0$. Since $\frac{1}{2}(|t| + |s| - |t-s|) = \min\{t,s\}$, we can consider the covariance function $\frac{1}{2}(|t| + |s| - |t-s|)$ as a basis for generalization. The first generalization is obtained by considering a continuous zero-mean Gaussian process with covariance $\rho(t,s) = \frac{1}{2}(|t|^\alpha + |s|^\alpha - |t-s|^\alpha)$, where α is a parameter such that

$\alpha = 1$ yields the Wiener process. This generalization gives rise to fractional Brownian motion discussed in the next section.

12.4.2 Fractional Brownian Motion

A continuous zero-mean Gaussian process $\{W_t, t \geq 0\}$ with covariance function

$$\mathbf{cov}(W_t, W_s) = \frac{1}{2}\left(|t|^\alpha + |s|^\alpha - |t-s|^\alpha\right), \quad t, s \geq 0 \tag{12.15}$$

is called *fractional Brownian motion* (fBm) with roughness parameter $\alpha \in (0,2)$. The process is frequently parameterized with respect to $H = \alpha/2$, in which case $H \in (0,1)$ is called the *Hurst* or *self-similarity* parameter. The notion of self-similarity arises, because fBm satisfies the property that the rescaled process $\{c^{-H} W_{ct}, t \geq 0\}$ has the same distribution as $\{W_t, t \geq 0\}$ for all $c > 0$.

Simulation of fBm on the uniformly spaced grid $0 = t_0 < t_1 < t_2 < \cdots < t_n = 1$ can be achieved by first generating the increment process $\{X_1, X_2, \ldots, X_n\}$, where $X_i = W_i - W_{i-1}$, and then delivering the cumulative sum

$$W_{t_i} = c^H \sum_{k=1}^{i} X_k, \; i = 1, \ldots, n, \quad c = 1/n \,.$$

The increment process $\{X_1, X_2, \ldots, X_n\}$ is called *fractional Gaussian noise* and can be characterized as a discrete zero-mean stationary Gaussian process with covariance

$$\mathbf{cov}(X_i, X_{i+k}) = \frac{1}{2}\left(|k+1|^\alpha - 2|k|^\alpha + |k-1|^\alpha\right), \quad k = 0, 1, 2, \ldots$$

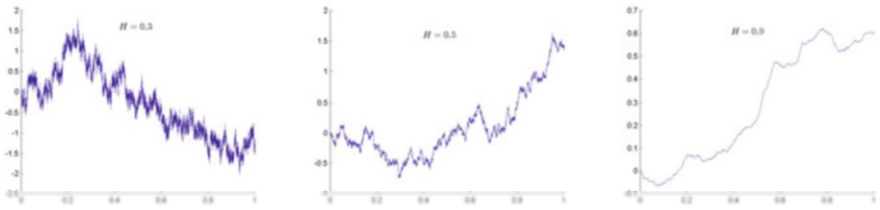

Fig. 12.11 Fractional Brownian motion for different values of the Hurst parameter H. From left to right we have $H = 0.3, 0.5, 0.9$

Since the fractional Gaussian noise is stationary, we can generate it efficiently using the circulant embedding approach in Sect. 12.2.2. First, we compute the first row $(r_1, \ldots, r_{n+1})$ of the symmetric Toeplitz $(n+1) \times (n+1)$ covariance matrix Ω with elements $\Omega_{i+1,j+1} = \mathbf{cov}(X_i, X_j)$, $i, j = 0, \ldots, n$. Second, we build the first row

of the $2n \times 2n$ circulant matrix Σ, which embeds Ω in the upper left $(n+1) \times (n+1)$ corner. Thus, the first row of Σ is given by $r = (r_1, \ldots, r_{n+1}, r_n, r_{n-1}, \ldots, r_2)$. We now seek a factorization of the form (12.3). Here, λ is the one-dimensional FFT of r defined as the linear transformation $\lambda = Fr$ with $F_{j,k} = \exp(-2\pi \mathrm{i} jk/(2n))/\sqrt{2n}$, $j,k = 0,1,\ldots,2n-1$. Finally, the real and imaginary parts of the first $n+1$ components of $F^* \mathrm{diag}(\sqrt{\lambda})Z$, where Z is a $2n \times 1$ complex valued Gaussian vector, yield two independent realizations of fractional Brownian noise. Fig. 12.11 shows how the smoothness of fBm depends on the Hurst parameter. Note that $H = 0.5$ corresponds to the Wiener process.

The following MATLAB code implements the circulant embedding method for fBm.

```
n=2^15; % grid points
H = 0.9; %Hurst parameter
r=nan(n+1,1); r(1) = 1;
for k=1:n
    r(k+1) = 0.5*((k+1)^(2*H) - 2*k^(2*H) + (k-1)^(2*H));
end
r=[r; r(end-1:-1:2)]; % first rwo of circulant matrix
lambda=real(fft(r))/(2*n); % eigenvalues
W=fft(sqrt(lambda).*complex(randn(2*n,1),randn(2*n,1)));
W = n^(-H)*cumsum(real(W(1:n+1))); % rescale
plot((0:n)/n,W);
```

12.4.3 Fractional Wiener Sheet in $\mathbb{R}^2$

A simple spatial generalization of the fractional Brownian motion is the fractional Wiener sheet in two dimensions. The *fractional Wiener sheet* process on the unit square is the continuous zero-mean Gaussian process $\{W_t, t \in [0,1]^2\}$ with covariance function

$$\mathbf{cov}(W_t, W_s) = \frac{1}{4}(|s_1|^\alpha + |t_1|^\alpha - |s_1 - t_1|^\alpha)(|s_2|^\alpha + |t_2|^\alpha - |s_2 - t_2|^\alpha)\,, \qquad (12.16)$$

where $t = (t_1, t_2)$ and $s = (s_1, s_2)$. Note that (12.16) is simply a product form extension of (12.15).

As in the one-dimensional case of fBm, we can consider the *two-dimensional fractional Gaussian noise* process $\{X_{i,j},\ i,j = 1,\ldots,n\}$, which can be used to construct a fractional Wiener sheet on a uniformly spaced square grid via the cumulative sum

$$W_{t_i,t_j} = n^{-2H} \sum_{k=1}^{i} \sum_{l=1}^{j} X_{k,l}\,, \quad i,j = 1,\ldots,n\,.$$

Note that this process is self-similar in the sense that $(mn)^{-H} \sum_{k=1}^{m} \sum_{l=1}^{n} X_{k,l}$ has the same distribution as $X_{1,1}$ for all m, n, and H, see [260].

Generating a the two-dimensional fractional Gaussian noise process requires that we generate a zero-mean stationary Gaussian process with covariance [324]

$$\mathbf{cov}(X_{i,j}, X_{i+k,j+l}) = \frac{|k+1|^\alpha - 2|k|^\alpha + |k-1|^\alpha}{2} \times \frac{|l+1|^\alpha - 2|l|^\alpha + |l-1|^\alpha}{2}$$

for $k, l = 0, 1, \ldots, n$. We can thus proceed to generate this process using the circulant embedding method in Sect. 12.2.2. The simulation of the Wiener sheet for $H = 0.5$ is particularly easy since then all of the $X_{i,j}$ are independent standard normally distributed. Fig. 12.12 shows realizations of fractional Wiener sheets for $H = \frac{1}{2}\alpha \in \{0.2, 0.5, 0.8\}$ with $n = 2^9$.

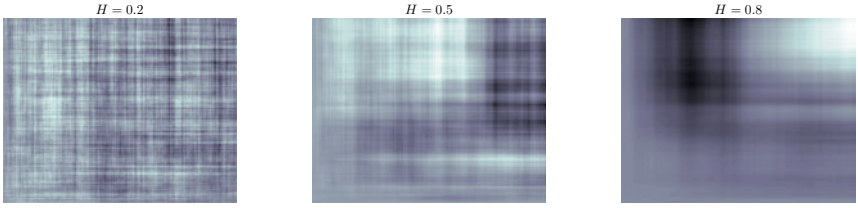

Fig. 12.12 Fractional Wiener sheets with different Hurst parameter H

Exercise 12.4. Show that for the special case of $\alpha = 1$ (that is, $H = 1/2$), we can write the covariance function (12.16) as $\mathbf{cov}(W_t, W_s) = \min\{t_1, s_1\} \min\{t_2, s_2\}$.

Two objects which are closely related to W_t are the *Wiener pillow* and the *Wiener bridge*, which are zero-mean Gaussian processes on $[0,1]^d$ with covariance functions $\mathbf{cov}(W_t, W_s) = \prod_{i=1}^d (\min(t_i, s_i) - t_i s_i)$ and $\prod_{i=1}^d \min(t_i, s_i) - \prod_{i=1}^d t_i s_i$, respectively.

12.4.4 Fractional Brownian Field

Fractional Brownian surface or field in two dimensions can be defined as the zero-mean Gaussian process $\{\tilde{X}_t, t \in \mathbb{R}^2\}$ with nonstationary covariance function

$$\mathbf{cov}(\tilde{X}_s, \tilde{X}_t) = \tilde{\rho}(s,t) = \|s\|^\alpha + \|t\|^\alpha - \|s-t\|^\alpha . \tag{12.17}$$

The parameter $H = \frac{\alpha}{2} \in (0,1)$ is the Hurst parameter controlling the roughness of the random field or surface. Contrast this isotropic generalization of the covariance (12.15) with the product form extension of the covariance in (12.16).

Simulation of $\tilde{X}_t$ over a unit (quarter) disk in the first quadrant involves the following steps. First, we use Dietrich and Newsam's method (Sect. 12.2.2) to generate a stationary Gaussian field $\breve{X}_t$ with covariance function over the quarter disk

$\{h : \|h\| \le 1, h > 0\}$

$$\check{\rho}(s,t) = c_0 + c_2\|s-t\|^2 - \|s-t\|^\alpha$$

for some constants $c_0, c_2 \ge 0$ whose selection will be discussed later. Once we have generated $\check{X}_t$, the process $\tilde{X}_t$ is obtained via the adjustment:

$$\tilde{X}_t = \check{X}_t - \check{X}_0 + \sqrt{2c_2}\, t^\intercal Z, \quad Z = (Z_1, Z_2)^\intercal, \quad Z_1, Z_2 \overset{\text{iid}}{\sim} N(0,1)\,.$$

This adjustment ensures that the covariance structure of $\tilde{X}_t$ over the disk $\{h : \|h\| \le 1, h > 0\}$ is given by (12.17).

Exercise 12.5. Show that that the covariance structure of $\tilde{X}_t$ is indeed given by $\tilde{\rho}(s,t)$, i.e., prove that for any $s,t \in \mathbb{R}^2$

$$\mathbf{cov}(\tilde{X}_s, \tilde{X}_t) = \tilde{\rho}(s,t)\,.$$

Hint. Verify that

$$\begin{aligned}
\mathbf{cov}(\tilde{X}_s, \tilde{X}_t) &= \mathbf{cov}(\check{X}_s - \check{X}_0 + \sqrt{2c_2}\, s^\intercal Z, \check{X}_t - \check{X}_0 + \sqrt{2c_2}\, t^\intercal Z) \\
&= \check{\rho}(s,t) - \check{\rho}(s,0) - \check{\rho}(0,t) + \check{\rho}(0,0) + 2c_2\, \mathbf{cov}(s^\intercal Z, t^\intercal Z) \\
&= \check{\rho}(s,t) - (c_0 + c_2\|t\|^2 - \|s\|^\alpha) - (c_0 + c_2\|t\|^2 - \|t\|^\alpha) + c_0 + 2c_2\, s^\intercal t \\
&= \tilde{\rho}(s,t) + \underbrace{c_2\|s-t\|^2 - c_2\|t\|^2 - c_2\|s\|^2 + 2c_2\, s^\intercal t}_{=0}\,.
\end{aligned}$$

It now remains to explain how we generate the process $\check{X}_t$. The idea is to generate the process on $[0,R]^2$, $R \ge 1$ via the intrinsic embedding of Stein (see (12.5)) using the covariance function:

$$\psi(h) = \begin{cases} c_0 + c_2\|h\|^2 - \|h\|^\alpha, & \text{if } \|h\| \le 1, \\ \frac{\beta(R-\|h\|)^3}{\|h\|}, & \text{if } 1 \le \|h\| \le R, , \\ 0, & \text{if } \|h\| \ge R, \end{cases} \tag{12.18}$$

where depending on the value of α, the constants $R \ge 1, \beta \ge 0, c_2 > 0, c_0 \ge 0$ are defined in Table 12.1.

Note that for $\alpha > 1.5$, the parameters needed for a nonnegative embedding are more complex, because a covariance function that is smoother close to the origin has to be even smoother elsewhere. In particular, for $\alpha > 1.5$ the choice of constants ensures that ψ is twice continuously differentiable as a function of $\|h\|$. Notice that while we generate the process $\tilde{X}_t$ over the square grid $[0,R]^2$, we are only interested in $\tilde{X}_t$ restricted inside the quarter disk with covariance $\check{\rho}(s,t)$. Thus, in order to reduce the computational effort, we would like to have $R \ge 1$ as close as possible to 1. While the optimal choice $R = 1$ guarantees a nonnegative embedding for all $\alpha \le 1.5$, in general we need $R > 1$ to ensure the existence of a minimal embedding for $\alpha > 1.5$. The choice $R = 2$ given in Table 12.1 is the most conservative one that

Table 12.1 Parameter values needed to ensure that (12.18) allows for a nonnegative circulant embedding.

	$0 < \alpha \leq 1.5$	$1.5 < \alpha < 2$
R	1	2
β	0	$\frac{\alpha(2-\alpha)}{3R(R^2-1)}$
c_2	$\frac{1}{2}\alpha$	$\frac{\alpha-\beta(R-1)^2(R+2)}{2}$
c_0	$1-c_2$	$\beta(R-1)^3+1-c_2$

guarantees a nonnegative circulant embedding for $\alpha > 1.5$. Smaller values of R that admit a nonnegative circulant embedding can be determined numerically [377, Table 1]. As a numerical example consider generating a fractional Brownian surface with $m = n = 1000$ and for Hurst parameter $H \in (0.2, 0.5, 0.8)$. Fig. 12.13 shows the effect of the parameter on the smoothness of the surface, with larger values providing a smoother surface. We used the following MATLAB code for the generation

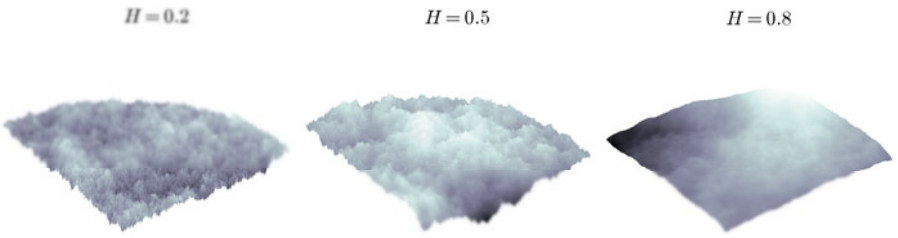

Fig. 12.13 Fractional Brownian fields with different roughness parameter $\alpha = 2H$

of the surfaces.

```
H=0.8;          % Hurst parameter
R=2;            % [0,R]^2 grid, may have to extract only  [0,R/2]^2
n=1000; m=n; % size of grid is m*n; covariance matrix is m^2*n^2
tx=[1:n]/n*R; ty=[1:m]/m*R; % create grid for field
Rows=zeros(m,n);
for i=1:n
    for j=1:m % rows of blocks of cov matrix
        Rows(j,i)=rho([tx(i),ty(j)],[tx(1),ty(1)],R,2*H);
    end
end
BlkCirc_row=[Rows, Rows(:,end-1:-1:2);
    Rows(end-1:-1:2,:), Rows(end-1:-1:2,end-1:-1:2)];
% compute eigen-values
lam=real(fft2(BlkCirc_row))/(4*(m-1)*(n-1));
lam=sqrt(lam);
% generate field with covariance given by block circulant matrix
Z=complex(randn(2*(m-1),2*(n-1)),randn(2*(m-1),2*(n-1)));
```

```
F=fft2(lam.*Z);
F=F(1:m,1:n); % extract sub-block with desired covariance
[out,c0,c2]=rho([0,0],[0,0],R,2*H);
field1=real(F); field2=imag(F); % two independent fields
field1=field1-field1(1,1); % set field zero at origin
% make correction for embedding with a term c2*r^2
field1=field1 + kron(ty'*randn,tx*randn)*sqrt(2*c2);
[X,Y]=meshgrid(tx,ty);
field1((X.^2+Y.^2)>1)=nan;
surf(tx(1:n/2),ty(1:m/2),field1(1:n/2,1:m/2),'EdgeColor','none')
colormap bone
```

The code uses the function `rho.m`, which implements the embedding (12.18).

```
function [out,c0,c2]=rho(x,y,R,alpha)
% embedding of covariance function on a [0,R]^2 grid
if alpha<=1.5 % alpha=2*H, where H is the Hurst parameter
    beta=0;c2=alpha/2;c0=1-alpha/2;
else % parameters ensure piecewise function twice differentiable
    beta=alpha*(2-alpha)/(3*R*(R^2-1));
    c2=(alpha-beta*(R-1)^2*(R+2))/2;
    c0=beta*(R-1)^3+1-c2;
end
% create continuous isotropic function
r=sqrt((x(1)-y(1))^2+(x(2)-y(2))^2);
if r<=1
    out=c0-r^alpha+c2*r^2;
elseif r<=R
    out=beta*(R-r)^3/r;
else
    out=0;
end
```

12.5 Spatial Levy Processes

Recall from Sect. 12.4.1 that the Brownian motion process (12.14) can be characterized as a continuous sample path process with stationary and independent Gaussian increments. The Lévy process is one of the simplest generalizations of the Brownian motion process, in cases where either the assumption of normality of the increments, or the continuity of the sample path is not suitable for modeling purposes.

12.5.1 Lévy Process

A d-dimensional *Lévy process* $\{X_t, t \in \mathbb{R}_+\}$ with $X_0 = 0$ is a stochastic process with a continuous index set on $\mathbb{R}_+$ and with a continuous state space $\mathbb{R}^d$ defined by the following properties:

1. the increments of $\{X_t\}$ are stationary, that is, $(X_{t+s} - X_t)$ has the same distribution as X_s for all $t, s \geq 0$;
2. the increments of $\{X_t\}$ are independent, that is, $X_{t_i} - X_{t_{i-1}}$, $i = 1, 2, \ldots$ are independent for any $0 \leq t_0 < t_1 < t_2 < \cdots$; and
3. for any $\varepsilon > 0$, we have $\mathbf{P}(\|X_{t+s} - X_t\| \geq \varepsilon) = 0$ as $s \downarrow 0$.

From the definition it is clear that Brownian motion (12.14) is an example of a Lévy process, with normally distributed increments. Brownian motion is the only Lévy process with continuous sample paths. Other basic examples of Lévy processes include the Poisson process $\{N_t, t \geq 0\}$ with intensity $\lambda > 0$, where $N_t \sim \text{Pois}(\lambda t)$ for each t, and the *compound Poisson process* defined via

$$J_t = \sum_{k=1}^{N_t} \delta X_k, \quad N_t \sim \text{Pois}(\lambda t)\,, \tag{12.19}$$

where $\delta X_1, \delta X_2, \ldots$ are independent and identically distributed random variables, independent of $\{N_t, t \geq 0\}$. We can more generally express J_t as $J_t = \int_0^t \int_{\mathbb{R}^d} x N(\mathrm{d}s, \mathrm{d}x)$, where $N(\mathrm{d}s, \mathrm{d}x)$ is a Poisson random counting measure on $\mathbb{R}_+ \times \mathbb{R}^d$ (see Sect. 12.3.1) with mean measure $\mathbf{E}N([0,t] \times A)$, $A \in \mathcal{E}$, equal to the expected number of jumps of size A in the interval $[0,t]$.

A crucial property of Lévy processes is infinite divisibility. In particular, if we define $Y_j^{(n)} = X_{jt/n} - X_{(j-1)t/n}$, then using the stationarity and independence properties of the Lévy process, we obtain that for each $n \geq 2$ the $\{Y_1^{(n)}\}$ are independent and identically distributed random variables with the same distribution as $X_{t/n}$. Thus, for a fixed t we can write

$$X_t \sim Y_1^{(n)} + \cdots + Y_n^{(n)}\,, \text{ for any } n \geq 2\,, \tag{12.20}$$

and hence by definition the random vector X_t is *infinitely divisible* (for a fixed t). The Lévy–Khintchine theorem [338] gives the most general form of the characteristic function of an infinitely divisible random variable. Specifically, the logarithm of the characteristic function of X_t (the so-called *characteristic exponent*) is of the form

$$\log \mathbf{E} \mathrm{e}^{\mathrm{i} s^\intercal X_t} = \mathrm{i} t\, s^\intercal \mu - \frac{1}{2} t\, s^\intercal \Sigma s + t \int_{\mathbb{R}^d} \left(\mathrm{e}^{\mathrm{i} s^\intercal x} - 1 - \mathrm{i}\, s^\intercal x \mathbf{1}_{\{\|x\| \leq 1\}} \right) \nu(\mathrm{d}x)\,, \tag{12.21}$$

for some $\mu \in \mathbb{R}^d$, covariance matrix Σ and measure ν such that $\nu(\{0\}) = 0$,

$$\int_{\|x\|>1} \nu(\mathrm{d}x) < \infty \text{ and } \int_{\|x\| \leq 1} \|x\|^2 \nu(\mathrm{d}x) < \infty \Leftrightarrow \int_{\mathbb{R}^d} \min\{1, \|x\|^2\} \nu(\mathrm{d}x) < \infty\,. \tag{12.22}$$

The triplet (μ, Σ, ν) is referred to as the *characteristic triplet* defining the Lévy process. The measure ν is referred to as the *Lévy measure*. Note that for a general ν satisfying (12.22), the integral $\int_{\mathbb{R}^d} \left(\mathrm{e}^{\mathrm{i}s^\top x} - 1\right)\nu(\mathrm{d}x)$ in (12.21) does not converge separately. In this sense $\mathrm{i}s^\top x \mathbf{1}_{\{\|x\|\le 1\}}$ in (12.21) serves the purpose of enforcing convergence under the very general integrability condition (12.22). However, if in addition to (12.22) the measure ν satisfies $\int_{\mathbb{R}^d} \min\{1, \|x\|\}\nu(\mathrm{d}x) < \infty$ and $\nu(\mathbb{R}^d) < \infty$, then the integral in (12.21) can be separated as $t\int_{\mathbb{R}^d}(\mathrm{e}^{\mathrm{i}s^\top x} - 1)\nu(\mathrm{d}x) - \mathrm{i}t\, s^\top \int_{\|x\|\le 1} x\,\nu(\mathrm{d}x)$, and the characteristic exponent simplifies to

$$\underbrace{\mathrm{i}t\, s^\top \mu^* - \frac{1}{2} t\, s^\top \Sigma\, s}_{\text{Brownian motion term}} + \underbrace{t\int_{\mathbb{R}^d}\left(\mathrm{e}^{\mathrm{i}s^\top x} - 1\right)\nu(\mathrm{d}x)}_{\text{Poisson process term}},$$

where $\mu^* = \mu - \int_{\|x\|\le 1} x\,\nu(\mathrm{d}x)$. We can now recognize this characteristic exponent as the one corresponding to the process $\{X_t\}$ defined by $X_t = t\mu^* + \Sigma^{1/2}W_t + J_t$, where $t\mu^* + \Sigma^{1/2}W_t$ defines a Brownian motion (see (12.14)) and $\{J_t\}$ is a compound Poisson process (12.19) with jump size distribution $\delta X_1 \sim \nu(\mathrm{d}x)/\lambda$. Thus, $\nu(\mathrm{d}x)$ can be interpreted as the intensity function of the jump sizes in this particular Lévy process. In a similar way, it can be shown that the most general Lévy process $\{X_t\}$ with characteristic triplet (μ, Σ, ν) and integrability condition (12.22) can be represented as the limit in probability of a process $\{X_t^{(\varepsilon)}\}$ as $\varepsilon \downarrow 0$, where $X_t^{(\varepsilon)}$ has the *Lévy–Itô decomposition*

$$X_t^{(\varepsilon)} = t\mu + \Sigma^{1/2}W_t + J_t + \left(J_t^{(\varepsilon)} - t\int_{\varepsilon<\|x\|\le 1} x\,\nu(\mathrm{d}x)\right) \tag{12.23}$$

with the following *independent* components:

1. $\{t\mu + \Sigma^{1/2}W_t\}$ is the Brownian motion (12.14), which corresponds to the $\mathrm{i}t\, s^\top\mu - \frac{1}{2}t\, s^\top\Sigma\, s$ part of (12.21).
2. $\{J_t\}$ is a compound Poisson process of the form (12.19) with $\lambda = \int_{\|x\|>1}\nu(\mathrm{d}x)$ and increment distribution $\delta X_1 \sim \nu(\mathrm{d}x)/\lambda$ over $\|x\| > 1$, which corresponds to the $\int_0^t \int_{\|x\|>1}(\mathrm{e}^{\mathrm{i}s^\top x} - 1)\nu(\mathrm{d}x)\,\mathrm{d}t$ part in the characteristic exponent (12.21).
3. $\{J_t^{(\varepsilon)}\}$ is a compound Poisson process with $\lambda = \nu(\varepsilon < \|x\| \le 1)$ and increment distribution $\nu(\mathrm{d}x)/\lambda$ over $\varepsilon < \|x\| \le 1$, so that the *compensated* compound Poisson process $\{J_t^{(\varepsilon)} - t\int_{\varepsilon<\|x\|\le 1} x\,\nu(\mathrm{d}x)\}$ corresponds to the $\int_0^t \int_{\|x\|\le 1}\Big(\mathrm{e}^{\mathrm{i}s^\top x} - 1 - \mathrm{i}s^\top x\Big)\nu(\mathrm{d}x)\,\mathrm{d}t$ part of (12.21) in the limit $\varepsilon \downarrow 0$.

The Lévy–**1**tô decomposition immediately suggests an approximate generation method — we generate $\{X_t^{(\varepsilon)}\}$ in (12.23) for a given small ε, where the Brownian motion part is generated via the methods in Sect. 12.4.1 and the compound Poisson process (12.19) in the obvious way. We are thus throwing away the very small jumps of size less than ε. For $d = 1$ it can then be shown [8] that the error pro-

cess $\{X_t - X_t^{(\varepsilon)}\}$ for this approximation is a Lévy process with characteristic triplet $(0,0,\nu(\mathrm{d}x)\mathbf{1}_{\{|x|<\varepsilon\}})$ and variance $\mathbf{var}(X_t - X_t^{(\varepsilon)}) = t\int_{-\varepsilon}^{\varepsilon} x^2 \nu(\mathrm{d}x)$.

A given approximation $X_t^{(\varepsilon_n)}$ can always be further refined by adding smaller jumps of size $[\varepsilon_{n+1}, \varepsilon_n]$ to obtain:

$$X_t^{(\varepsilon_{n+1})} = X_t^{(\varepsilon_n)} + J_t^{(\varepsilon_{n+1})} - t\int_{\varepsilon_{n+1}<\|x\|\le\varepsilon_n} x\,\nu(\mathrm{d}x)\,, \quad \varepsilon_n > \varepsilon_{n+1} > 0\,,$$

where the compound Poisson process $\{J_t^{(\varepsilon_{n+1})}\}$ has increment distribution given by $\nu(\mathrm{d}x)/\nu(\varepsilon_{n+1} < \|x\| \le \varepsilon_n)$ for all $\varepsilon_{n+1} < \|x\| \le \varepsilon_n$. For a more sophisticated method of refining the approximation, see [8].

As an example consider the Lévy process $\{X_t\}$ with characteristic triplet $(\mu,0,\nu)$, where $\nu(\mathrm{d}x) = \alpha\,\mathrm{e}^{-x}/x\,\mathrm{d}x$ for $\alpha, x > 0$ and $\mu = \int_{|x|\le 1} x\,\nu(\mathrm{d}x) = \alpha(1-\mathrm{e}^{-1})$. Here, in addition to (12.22), the infinite measure ν satisfies the stronger integrability condition $\int_{\mathbb{R}} \min\{1,|x|\}\nu(\mathrm{d}x) < \infty$ and hence we can write $X_t = t\mu - t\int_{|x|\le 1} x\,\nu(\mathrm{d}x) + \int_0^t \int_{\mathbb{R}} xN(\mathrm{d}s,\mathrm{d}x) = \int_0^t \int_{\mathbb{R}_+} xN(\mathrm{d}s,\mathrm{d}x)$, where $\mathbf{E}N(\mathrm{d}s,\mathrm{d}x) = \mathrm{d}s\,\nu(\mathrm{d}x)$.

The leftmost panel of Fig. 12.14 shows an outcome of the approximation $X_t^{(\varepsilon_1)}$ in (12.23) over $t \in [0,1]$ for $\varepsilon_1 = 1$ and $\alpha = 10$. The middle and rightmost panels show the refinements $X_t^{(\varepsilon_2)}$ and $X_t^{(\varepsilon_3)}$, respectively, where $(\varepsilon_1,\varepsilon_2,\varepsilon_3) = (1, 0.1, 0.001)$. Note that the refinements add finer and finer jumps to the path.

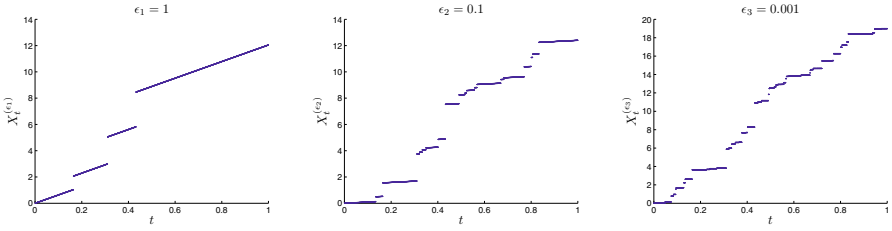

Fig. 12.14 Approximate gamma process realizations obtained by throwing away jumps smaller than ε_i

This process is called a *gamma process* and can be generated more simply using the fact that the increments $\{X_{jt/n} - X_{(j-1)t/n}\}$ have a known gamma distribution [237, p. 212].

Exercise 12.6. Show that the characteristic exponent corresponding to the triplet $(\mu,0,\nu)$ of the above gamma process is given by

$$\log \mathbf{E}\mathrm{e}^{\mathrm{i}sX_t} = \int_0^\infty \left(\mathrm{e}^{\mathrm{i}sx} - 1\right) \frac{\alpha \mathrm{e}^{-x}}{x}\,\mathrm{d}x = -\alpha \log(1 - \mathrm{i}s)\,.$$

Deduce from the characteristic exponent that any increment $X_{t+s} - X_t \sim \Gamma(\alpha s, 1)$. Here $\Gamma(\alpha,\beta)$ denotes the gamma distribution, with probability density function $\beta^\alpha x^{\alpha-1}\mathrm{e}^{-\beta x}/\Gamma(\alpha)$, $x \ge 0$.

The gamma process is an example of an increasing Lévy process (that is, $X_t \geq X_s$ almost surely for all $t \geq s$) called a *Lévy subordinator*. A Lévy subordinator $\{X_t, t \geq 0\}$ has characteristic triplet $(\mu, 0, \nu)$ satisfying the positive jump property $\nu((-\infty, 0]) = 0$ and the positive drift property $0 \leq \mu - \int_0^1 x\nu(\mathrm{d}x) < \infty$.

12.5.2 Lévy Sheet in $\mathbb{R}^2$

One way of generalizing the Lévy process to the spatial case is to insist on the preservation of the infinite divisibility property. In this generalization, a *spatial Lévy process* or simply a *Lévy sheet* $\{X_t, t \in \mathbb{R}^2\}$ possesses the property that $(X_{t_1}, \ldots, X_{t_n})$ is infinitely divisible for all indices $t_1, \ldots, t_n \in \mathbb{R}^2$ and any integer n (see (12.20)). To construct such an infinitely divisible process, consider stochastic integration with respect to a *random infinitely divisible measure* Λ defined as the stochastic process $\{\Lambda(A),\ A \in \mathcal{E}\}$ with the following properties:

1. For any set $A \in \mathcal{E}$ the random variable $\Lambda(A)$ has an infinitely divisible distribution with $\Lambda(\emptyset) = 0$ almost surely.
2. For any disjoint sets $A_1, A_2, \ldots \in \mathcal{E}$, the random variables $\Lambda(A_1), \Lambda(A_2), \ldots$ are independent and $\Lambda(\cup_i A_i) = \sum_i \Lambda(A_i)$.

An example of a random infinitely divisible measure is the Poisson random measure (12.6) in Sect. 12.3.1. As a consequence of the independence and infinite divisibility properties, the characteristic exponent of $\Lambda(A)$, $A \in \mathcal{E}$ has the Lévy–Khintchine representation (12.21):

$$\log \mathbf{E} \mathrm{e}^{\mathrm{i}s\Lambda(A)} = \mathrm{i}s\tilde{\mu}(A) - \frac{1}{2}s^2(\tilde{\sigma}(A))^2 + \int_{\mathbb{R}} \left(\mathrm{e}^{\mathrm{i}sx} - 1 - \mathrm{i}sx\mathbf{1}_{\{|x|\leq 1\}}\right) \tilde{\nu}(\mathrm{d}x, A)\ ,$$

where $\tilde{\mu}$ is an additive set function, $\tilde{\sigma}$ is a measure on the Borel sets $\mathcal{E}$, the measure $\tilde{\nu}(\cdot, A)$ is a Lévy measure for each fixed $A \in \mathcal{E}$ (so that $\tilde{\nu}(\{0\}, A) = 0$ and $\int_{\mathbb{R}} \min\{1, x^2\}\tilde{\nu}(\mathrm{d}x, A) < \infty$), and $\tilde{\nu}(\mathrm{d}x, \cdot)$ is a Borel measure for each fixed $\mathrm{d}x$. For example, $X_t = \Lambda((0,t])$ defines a one-dimensional Lévy process.

We can then construct a Lévy sheet $\{X_t, t \in \mathbb{R}^2\}$ via the stochastic integral

$$X_t = \int_{\mathbb{R}^d} \kappa_t(x)\Lambda(\mathrm{d}x), \quad t \in \mathbb{R}^2\ , \tag{12.24}$$

where $\kappa_t : \mathbb{R}^d \to \mathbb{R}$ is a Hölder continuous *kernel function* for all $t \in \mathbb{R}^2$, which is integrable with respect to the random infinitely divisible measure Λ. Thus, the Lévy sheet (12.24) is a stochastic integral with a deterministic kernel function as integrand (determining the spatial structure) and a random infinitely divisible measure as integrator [220].

Consider simulating the Lévy sheet (12.24) over $t \in [0,1]^2$ for $d = 2$. Truncate the region of integration to a bounded domain, say $[0,1]^2$, so that $\kappa_t(x) = 0$ for each $x \notin [0,1]^2$ and $t \in [0,1]^2$. Then, one way of simulating $\{X_t, t \in [0,1]^2\}$ is to consider

the approximation

$$X_t^{(n)} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \kappa_t(i/n, j/n)\, \Lambda(\triangle_{i,j}), \quad \triangle_{i,j} \equiv \left[\frac{i}{n}, \frac{i+1}{n}\right] \times \left[\frac{j}{n}, \frac{j+1}{n}\right], \quad (12.25)$$

where all $\Lambda(\triangle_{i,j})$ are independent infinitely divisible random variables with characteristic triplet $(\tilde{\mu}(\triangle_{i,j}), \tilde{\sigma}(\triangle_{i,j}), \tilde{\nu}(\cdot, \triangle_{i,j}))$. Under some technical conditions [220], it can be shown that $X_t^{(n)}$ converges to X_t in probability as $n \uparrow \infty$.

As an example, consider generating (12.25) on the square grid $\{(i/m, j/m),\, i, j = 0, \ldots, m-1\}$ with the kernel function

$$\kappa_t(x_1, x_2) = (r^2 - \|x-t\|^2)\, \mathbf{1}_{\{\|x-t\| \le r\}}, \quad t \in [0,1]^2,$$

and $\Lambda(\triangle_{i,j}) \sim \Gamma(\alpha|\triangle_{i,j}|, \beta)$, $|\triangle_{i,j}| = 1/n^2$ for all i, j. The corresponding limiting process $\{X_t\}$ is called a *gamma Lévy sheet*. Fig. 12.15 shows realizations of (12.25) for $m = n = 100$ and $r = 0.05$, $\alpha = \beta \in \{10^2, 10^5\}$, so that we have the scaling $\mathbf{E}\Lambda(\triangle_{i,j}) = \alpha/(\beta n^2) = 1/n^2$.

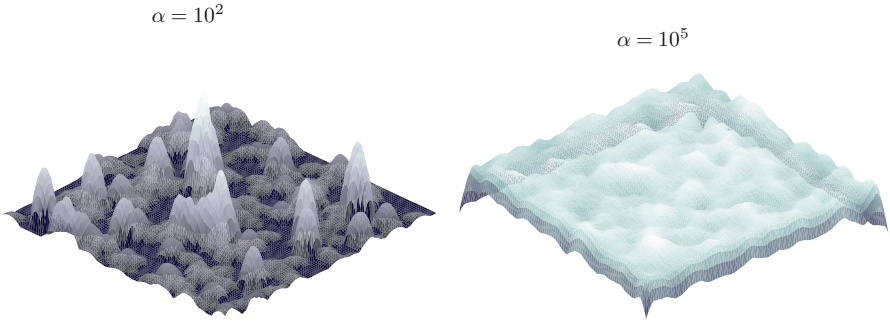

Fig. 12.15 Gamma Lévy random sheet realizations for different values of the shape parameter α

Note that the sheet exhibits more bumps for smaller values of α than for large values of α. For a method of approximately generating (12.24) using wavelets see [220].

Acknowledgements This work was supported by the Australian Research Council under grant number DP0985177.