Dominated Coupling From The Past and Some Extensions of the Area-Interaction Process

By

Graeme K. Ambler

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Science

September 2002

Department of Mathematics

Abstract

Dominated coupling from the past, an example of perfect simulation, is a method for sampling from the equilibrium distribution of a Markov chain. A particular model which may be sampled in this way is the area-interaction point process of Baddeley and van Lieshout (1995). This thesis reviews perfect simulation and some aspects of point process theory. It also reviews some methods for simulating spatial point processes. It then introduces an extension of the area-interaction process which incorporates both attractive and repulsive components at different scales. The properties of this model are investigated using some standard test functions, and the model is fitted to a standard data set. Some discrete analogues of these models are then discussed along with methods of sampling from these models using dominated coupling from the past. While discussing these sampling methods, we highlight some of the pitfalls that exist when one attempts to use dominated coupling from the past. We set out and investigate the use of the discrete analogue of the area-interaction process as a prior distribution on wavelet coefficients in Bayesian non-parametric regression. Finally, we discuss some of the issues that arose while implementing the algorithms, both for simulating from the attractive-repulsive model and for the posterior distribution of the Bayesian wavelet thresholding model.

Acknowledgements

I would like to thank my supervisor, Professor B.W. Silverman, for his guidance and encouragement throughout my Ph.D. I am also grateful to many of the other members of the Statistics group for their advice and support, in particular Dr. Stephen P. Brooks, who encouraged me to start a Ph.D.; Professor Peter J. Green, who suggested that I might study perfect simulation; and Dr E.J. Collins, who was very supportive when I needed to take some time out due to ill health.
I would also like to thank my parents and my friends for their love and encouragement. Finally I must thank my wife Rachel. Without her constant love and support I could never have finished this.

The research was carried out with the financial support of an EPSRC studentship.

Declaration

I, the author, declare that the work in this dissertation was carried out in accordance with the Regulations of the University of Bristol. The work is original except where indicated by special reference in the text and no part of the dissertation has been submitted for any other degree.

The views expressed in the dissertation are those of the author and in no way represent those of the University of Bristol.

The dissertation has not been presented to any other University for examination either in the United Kingdom or overseas.

Graeme K. Ambler

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In 1974, G. S. Watson wrote "Modern statistics might well be defined as the application of computers and mathematics to data analysis"[1]. This is certainly no less true today than it was then. This thesis is concerned with two areas of statistics which have become practical only with the rapid increase in both computer power and availability that we have seen in the last 40 years. The first is spatial modelling, where we are now able to fit models which are sufficiently complex that we may expect to be able to explain many kinds of data with a reasonable level of detail. The second is stochastic simulation, which has become increasingly important with the rise of Markov Chain Monte Carlo methods.

Spatial point processes and their discrete cousin Lattice processes are common in everyday life. Examples of the former are the locations of trees in a forest or stars in the sky. Examples of the latter on a regular lattice include the pixels on a computer screen or in a digital photograph, though many examples on irregular lattices also exist, such as disease counts in the counties of England.

Once we have a proposed model for such data, the next step is to fit that model. One key method for assessing the goodness of fit of a model is the Monte Carlo test. This involves simulating data from the model several times and calculating values of some test statistics for those simulations and for the data. The values obtained from the data are then compared with the values obtained from the

[1] Forward to Matheron (1975).

simulations. Clearly for this method to be successful we must have reliable methods for simulating samples from the model. This is why there is an emphasis on simulation whenever we develop models.

Another area of statistics where it is desirable to have reliable methods of stochastic simulation is in Bayesian modelling. Here we basically express our beliefs about the structure of the data by specifying a *prior distribution* on the values of the model parameters and then obtain a *posterior distribution* by means of Bayes theorem, which expresses the distribution of the parameters given the data. We then simulate from this distribution and use suitable summary statistics, in some ways analogous to the test functions mentioned above, to make inferences about the model parameters.

Unfortunately it is not always simple, or even possible, to derive the exact posterior distribution of many Bayesian models. In many cases the best that we can do is to define it up to a multiplicative constant. Many spatial point process models are also only defined up to a multiplicative constant.

In both of the situations mentioned above it is common to use either discrete time Markov chains or continuous time Markov processes whose stationary distribution is the distribution we wish to sample from. This is possible because the multiplicative constant cancels out when defining the transition kernel of the Markov chain. The method of simulating a Markov chain whose stationary distribution is the distribution we want to sample from is known in general as Markov chain Monte Carlo. The traditional way to proceed is to run the Markov chain (or Markov process) for a 'long time' (called the burn-in time) in the hope that by the end of this period the Markov chain (or Markov process) will be sufficiently close to stationarity that we may assume that we are now sampling from the required distribution. Unfortunately, the question 'how long is long enough?' is in general very difficult to answer.

One partial solution, coupling from the past, was developed by Propp and Wilson (1996). Coupling from the past is a method for sampling from the equilibrium

distribution of a Markov chain through coupling multiple Markov chains with the same equilibrium distribution but different initial states. Other methods of sampling from the equilibrium distribution of a Markov chain have since been developed, and the general class of such algorithms is now commonly referred to as 'perfect simulation', a name first coined by Kendall (1998). Coupling from the past and some other methods of perfect simulation are introduced in Chapter 3.

We then move on to spatial point processes in Chapter 4, paying particular attention to the area-interaction process, which we also extend to enable us to model point patterns incorporating both clustering and repulsion at different scales. Due to this feature, we have chosen to call the new model an "attractive-repulsive model". Each time a model is discussed, it is accompanied by a section explaining how we would go about simulating draws from this model. In the cases of the area-interaction process and our attractive-repulsive model this method relies on an extension of the perfect simulation techniques developed in Chapter 3 called dominated coupling from the past. Various descriptive functions are also introduced, and are used to investigate the properties of our attractive-repulsive process. We also apply our attractive-repulsive process to a standard data set and show that it fits well.

In Chapter 5 we develop discrete analogues of the spatial point processes discussed in Chapter 4. Again we focus on methods of simulating these models, and along the way discuss various pitfalls that exist when one attempts to use perfect simulation. In Chapter 6 we present a Bayesian wavelet thresholding scheme which uses one of the models developed in Chapter 5 as a prior for the distribution of significant wavelet coefficients. We also present a method for perfectly simulating the posterior distribution of this model, and give some examples of the method's performance.

Finally, in Chapter 7 we discuss some of the issues which arose when implementing the algorithms introduced in Chapters 4 and 6.

Chapter 2

Preliminaries

Due to the rather abstract nature of spatial point process theory, it is necessary to begin by introducing some appropriate preliminary mathematical theory. We begin with a brief discussion of topology in Section 2.1, before going on to discuss some basic principles of measure theory as they apply to spatial point processes in Section 2.2. With these in place, we then present a small amount of spatial point process theory in Section 2.3. For the sake of those who are not interested in the precise formulation and wish to treat the subject intuitively, we present in Section 2.3.1 a brief heuristic summary of the concepts dealt with formally in Section 2.3.

For a more complete treatment of the topics covered below see Sutherland (1975) for general topology and the introductory chapters of Matheron (1975) for the topology not covered in Sutherland[1]. Kingman and Taylor (1966) is an excellent text on measure theory although Williams (1991) gives a more intuitive treatment. The material on upper and lower semicontinuous functions comes from Jost (1998). For the material on spatial point process theory see Stoyan et al. (1995), though the material on Markov point processes is only covered briefly.

2.1 Topology

A *topological space* $T = \{A, \mathcal{G}\}$ consists of a non-empty set A together with a fixed collection $\mathcal{G}$ of subsets of A satisfying

[1] Matheron is *not* an introductory text and should not be read unless the reader is familiar with all of the topics covered in Sutherland!

- A, $\phi \in \mathcal{G}$,
- If $G_1, \ldots, G_n \in \mathcal{G}$ is a finite collection of elements of $\mathcal{G}$, then

$$\bigcap_{i=1}^{n} G_i \in \mathcal{G},$$

- The union of any collection of sets in $\mathcal{G}$ is in $\mathcal{G}$.

The collection $\mathcal{G}$ of subsets of A is called a *topology* for A and the members of $\mathcal{G}$ are called the *open sets* of T. If $\mathcal{H}$ is a collection of subsets of A then $\mathcal{H}$ augmented by those sets which follow from $\mathcal{H}$ together with the three rules above is called the topology *generated* by $\mathcal{H}$. Readers familiar with metric space theory should note that if there exists a metric ρ on A which determines the class $\mathcal{G}$ as it's open sets then T is said to be *metrisable*.

If M is a subset of some topological space $T = \{A, \mathcal{G}\}$ then the pair $\{M, \mathcal{G}_M\}$ is a *subspace* of T, where the *induced topology* $\mathcal{G}_M$ on M consists of all sets $M \cap G$ with $G \in \mathcal{G}$.

A subset F of a topological space $T = \{A, \mathcal{G}\}$ is said to be *closed* in T if $A - F$ is open. $A - F$ is sometimes written F^c.

If $T = \{A, \mathcal{G}\}$ is a topological space, $x \in A$ a point and H a subset of A then x is a *limit point* of H if every open set in T containing x also contains some point of H other than x. The *closure* of H, $\bar{H}$, is the union of H and all the limit points of H in T. We say that H is *dense* in T if $\bar{H} = A$.

A topological space $T = \{A, \mathcal{G}\}$ is *separable* if there exists a countable set $H \subset A$ which is dense in T.

A *cover* for a set H is a collection of sets $\mathcal{C}$ such that

$$\bigcup_{C \in \mathcal{C}} C \supset H.$$

$\mathcal{C}_1$ is a *subcovering* of $\mathcal{C}$ if $\mathcal{C}_1 \subset \mathcal{C}$ and $\mathcal{C}_1$ is also a cover for H. A cover is *finite* if there are a finite number of elements in it. If H is a subspace of a topological

space $T = \{A, \mathcal{G}\}$ then a cover $\mathcal{C}$ of H is *open* if $\mathcal{C} \subset \mathcal{G}$ (i.e. $\mathcal{C}$ is an open cover in T if every set $C \in \mathcal{C}$ is open).

A topological space T is *compact* if every open cover for T has a finite subcover. A subset K of T is compact in T if K considered as a subspace of T is compact. T is *locally compact* if for every x in T there exists an open set U such that $x \in U$ and $\bar{U}$ is compact.

A topological space T is *Hausdorff* if given any two distinct points x and y in T there exist distinct open subsets U and V of T containing x and y respectively.

From here on it is assumed that any topological spaces discussed are locally compact, Hausdorff and separable.

If $T_1 = \{A_1, \mathcal{T}_1\}$ and $T_2 = \{A_2, \mathcal{T}_2\}$ are topological spaces and $f : A_1 \to A_2$ is a function then f is *continuous* with respect to $\mathcal{T}_1$ and $\mathcal{T}_2$ if

$$G \in \mathcal{T}_2 \implies f^{-1}(G) \in \mathcal{T}_1.$$

In other words f is continuous if the *inverse image* of an open set is open.

Let $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{K}$ respectively be the closed, open and compact subsets of a given topological space T. Then the *myope topology* (or *myopic topology*) $\mathcal{T}_k$ on $\mathcal{K}$ is generated by the classes

$$\begin{aligned} \mathcal{K}^F &= \{K \in \mathcal{K} : K \cap F = \phi\} \text{ and} \\ \mathcal{K}_G &= \{K \in \mathcal{K} : K \cap G \neq \phi\} \end{aligned}$$

for arbitrary $F \in \mathcal{F}$ and $G \in \mathcal{G}$.

Let $T = \{A, \mathcal{T}\}$ be a topological space and $f_i : A \to A_i$ be an I-indexed family of functions where $T_i = \{A_i, \mathcal{T}_i\}$ is a topological space for each $i \in I$. Then the *weak topology* on A with respect to $\{f_i : i \in I\}$ is the weakest (that is, smallest) topology $\mathcal{T}$ on A such that each f_i is continuous with respect to $\mathcal{T}$ and $\mathcal{T}_i$. In other words it is the topology generated by all sets of the form $f_i^{-1}(U_i)$ where $U_i \in \mathcal{T}_i$ for any $i \in I$.

2.2 Measure theory

Let $\mathcal{F}$ be a set of subsets of a given set A. Then $\mathcal{F}$ is a *σ-field* if

- $A \in \mathcal{F}$,
- $\mathcal{F}$ is closed under intersections and differences, i.e.

$$A, B \in \mathcal{F} \implies A \cap B \in \mathcal{F} \text{ and } A \triangle B \in \mathcal{F}$$

- $\mathcal{F}$ is closed under countable unions, i.e.

$$A_i \in \mathcal{F} \implies \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}.$$

If $\mathcal{F}$ is a σ-field then the pair $(A, \mathcal{F})$ is called a *measurable space*.

In a topological space $T = \{A, \mathcal{T}\}$ the σ-field $\mathcal{B}$ generated by $\mathcal{T}$ is the class of *Borel sets*.

Let A be a set and $\mathcal{C}$ be a collection of subsets of A. Then a function $\mu : \mathcal{C} \to \mathbb{R} \cup \{-\infty, \infty\}$ is said to be *σ-additive* if

- $\mu(\phi) = 0$
- For any disjoint sequence of sets $(A_i)_{i \in \mathbb{N}}$ such that $A_i \in \mathcal{C} \quad \forall i \in \mathbb{N}$ and $\bigcup_{i=1}^{\infty} A_i \in \mathcal{C}$,

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i).$$

The function μ is a *measure* if

- $\mu(S) \geq 0 \ \ \forall S \in \mathcal{C}$ and
- μ is σ-additive.

If $\mathcal{C}$ is a σ-field and μ is a measure then the triple $(A, \mathcal{C}, \mu)$ is called a *measure space*. If $\mu(A) = 1$ then it is a *probability space*, and μ is called a *probability measure*. If $\mu(A) < \infty$ then μ is said to be *finite* (or *totally finite*).

Let $\mathcal{S}$ be a σ-field of subsets of a topological space T which includes both the open subsets $\mathcal{G}$ and the closed subsets $\mathcal{F}$ of T. Then a measure $\mu : \mathcal{S} \to \mathbb{R}^+ \cup \{\infty\}$ is *regular* if for every $\varepsilon > 0$ and $E \in \mathcal{S}$ then:

1. $\exists G \in \mathcal{G}$ s.t. $G \supset E$ and $\mu(G - E) < \varepsilon$ and
2. $\exists F \in \mathcal{F}$ s.t. $F \subset E$ and $\mu(E - F) < \varepsilon$.

Let X be a metric space and $x \in X$. A function $f : X \to \mathbb{R} \cup \{\infty\}$ is called *lower semicontinuous (l.s.c)* at x if $\forall\, c \in \mathbb{R}$ with $c < f(x)$, there exists an open neighbourhood U of x such that $\forall\, y \in U$, $c < f(y)$. A function $g : X \to \mathbb{R} \cup \{-\infty\}$ is called *upper semicontinuous (u.s.c)* at x if $-g$ is lower semicontinuous at x. The function f (g) is called lower (upper) semicontinuous on X if it is lower (upper) semicontinuous at every $x \in X$. This leads to the straightforward result that f is lower semicontinuous if and only if $\forall\, c \in \mathbb{R}$ the set $f^{-1}((c, \infty]) = \{x \in X : c < f(x)\}$ is open in X.

A function $f : T \to \mathbb{R} \cup \{-\infty, +\infty\}$ is *measurable* with respect to a given σ-field $\mathcal{F}$ if

$$f^{-1}(B) \in \mathcal{F}$$

for every Borel set B (that is, Borel set in the standard topology over $\mathbb{R} \cup \{-\infty, +\infty\}$).

From the above definition and that of u.s.c. and l.s.c. it is immediately obvious that if a function is u.s.c. or l.s.c. then it is Borel measurable, since the sets $(c, \infty]$ generate the usual Borel σ-field over $\mathbb{R}$. This fact will be used in Section 4.4.

Suppose that $(\Omega, \mathcal{F}, \mu)$ is a measure space. Then the set function $\nu : \mathcal{F} \to \mathbb{R} \cup \{-\infty, +\infty\}$ is *absolutely continuous with respect to* μ (write $\nu \ll \mu$) if given $E \in \mathcal{F}$, $\nu(E) = 0$ whenever $\mu(E) = 0$.

Suppose we have μ and ν as above. Then there exists $f \in \mathcal{L}^1(\Omega, \mathcal{F}, \mu)$ such that

$$\nu(E) = \int_E f d\mu \qquad \forall E \in \mathcal{F}. \tag{2.1}$$

This is the *Radon-Nikodým theorem*. The function f is called (a version of) the *Radon-Nikodým derivative*, or *density* of ν relative to μ on $(\Omega, \mathcal{F})$. It is unique in the sense that if f and g both obey (2.1) then $f(x) = g(x)$ except on a set of zero μ-measure. We write

$$\frac{d\mu}{d\nu} = f \text{ on } \mathcal{F}, \text{ a.s.}$$

where a.s. stands for *almost surely* and means "except on a set of zero μ-measure".

2.3 Spatial point process theory

We consider only $\mathbb{R}^n$ (for some arbitrary n) here, though these notions are easily extended to any separable metric space.

Let $\mathcal{B}_0$ be the collection of bounded Borel sets in $\mathbb{R}^n$, and define $N = \{x \subset \mathbb{R}^n : n_x(B) < \infty \; \forall B \in \mathcal{B}_0\}$ where $n_x(B)$ is the number of points in x which are in the set B. Then intuitively N is the set of all $\mathcal{B}_0$-finite point configurations in $\mathbb{R}^n$. Now define the σ-algebra

$$\mathcal{N} = \sigma\left(\left\{\{x \in N : n_x(B) = m\} : m \in \mathbb{Z}^+, B \in \mathcal{B}_0\right\}\right),$$

where $\sigma(\cdot)$ denotes the σ-algebra generated by sets of the form $\cdot$. Then a measurable function, X, from some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ into the measurable space $(N, \mathcal{N})$ is called a *point process*.

With this structure in place we now define the *nth order moment measure* of a point process to be

$$M(A_1 \times A_2 \times \cdots \times A_n) = \mathbb{E}(n(A_1)n(A_2)\cdots n(A_n)),$$

for $A_i \in \mathcal{B}_0$ for $i = 1, \ldots, n$. The first order moment measure,

$$M(A) = \mathbb{E}(n(A)),$$

is often called the *intensity measure*.

Let $F \in \mathcal{N}$. Then the *Campbell measure*[2] of a point process X is

$$C(A \times F) = \mathbb{E}(n(A)I(X \in F)),$$

[2] The definition given is sometimes called the *first order* Campbell measure, where the nth order Campbell measure of a point process X is defined to be

$$C^{(n)}(A_1 \times A_2 \times \cdots \times A_n \times F) = \mathbb{E}(n(A_1)n(A_2)\cdots n(A_n)I(X \in F)),$$

where as with the nth order moment measure, $A_i \in \mathcal{B}_0$ for $i = 1, \ldots, n$. Clearly the nth order Campbell and moment measures are equal when $F = N$.

or equivalently

$$C(A \times F) = \mathbb{E}\left(\sum_{\xi \in X} I(\xi \in A, X \in F)\right),$$

where I is the indicator function. The *reduced Campbell measure* is then defined to be

$$C^!(A \times F) = \mathbb{E}\left(\sum_{\xi \in X} I(\xi \in A, X \setminus \xi \in F)\right).$$

Now assume that π is a σ-finite measure on $\mathbb{R}^n$. For each $F \in \mathcal{N}$, $C(\cdot \times F) \ll \pi$, so letting

$$\mathbb{P}_\xi(F) = \frac{dC(\cdot \times F)}{d\pi}(\xi),$$

we see that

$$C(A \times F) = \int_A \mathbb{P}_\xi(F) d\pi(\xi). \tag{2.2}$$

We can choose $\mathbb{P}_\xi(F)$ so that it is

1. measurable for all $F \in \mathcal{N}$
2. a probability measure on $\mathcal{N}$ for all $\xi \in \mathbb{R}^n$.

When so chosen, $\mathbb{P}_\xi$ is called the *Palm distribution at the point* ξ. The *reduced Palm distribution*, $\mathbb{P}^!_\xi$, is defined analogously using the reduced Campbell measure. In particular, (2.2) becomes

$$C^!(A \times F) = \int_A \mathbb{P}^!_\xi(F) d\pi(\xi).$$

Finally, we define Markov point processes (Ripley and Kelly 1977).

Let $\sim$ be a measurable symmetric reflexive relation on $\mathbb{R}^n$. We say that two points $x, y \in \mathbb{R}^n$ are *neighbours* if $x \sim y$. We define the border $\partial(A)$ of some set $A \subset \mathbb{R}^n$ by

$$\partial(A) = \{x : x \sim y \text{ for some } y \in A\},$$

and abbreviate $\partial(\{x\})$ by $\partial(x)$.

Now suppose that a point process X has density f with respect to the unit rate Poisson process. Then X is *Markov with respect to* $\sim$ if

1. $f(x) > 0 \implies f(y) > 0 \; \forall x, y$ s.t. $y \subset x$
2. For all $x \in N$, if $f(x) > 0$ and $u \notin x$ then

$$\lambda(u; x) = \frac{f(x \cup \{u\})}{f(x)}$$

depends only on u and $x \cap \partial(u)$.

The first property is called the *hereditary* property, and $\lambda(u; x)$ is called the *Papangelou conditional intensity* of x at u. For $x_i \in x$, $\lambda(x_i; x)$ is defined as

$$\lambda(x_i; x) = \frac{f(x)}{f(x \setminus \{x_i\})}.$$

The Papangelou conditional intensity is used extensively in simulation and estimation of parameters, as the normalising constant in the top and bottom lines cancels, making it possible to manipulate models where the normalising constant is unknown.

Processes which are Markov with respect to the relation

$$x \sim y \iff d(x, y) \leq r$$

are called *Markov of range* r.

2.3.1 Summary

Health Warning: The purpose of this section is to give heuristic interpretations of the material in the previous section. The statements in this section are *not precise* and should not be thought of as definitions!

The set N is the set of all point configurations over $\mathbb{R}^n$ which have only finite numbers of points in bounded regions. The set $\mathcal{N}$ is a certain collection of subsets of N. A *point process* is a mapping which assigns probability to configurations of points.

Let B be a bounded set. Then under certain regularity conditions on the nature of B (specifically, B should be a Borel set), the *intensity measure*, λ_X, of a spatial

point process is a function which will tell us the expected number of points in B:

$$\lambda_X(B) = \mathbb{E}(n(X_B))$$

where X_B are those points in X which are also in B.

The *Palm distribution* of a point process, $\mathbb{P}_\xi(X)$, can be thought of as the conditional distribution of X given that there is a a point at ξ. The *Reduced Palm distribution*, $\mathbb{P}^!_\xi(X)$ can be thought of as the conditional distribution of $X \setminus \xi$ (i.e. the remainder of the process) given that there is a point at ξ.

The *Papangelou conditional intensity*, $\lambda(u, x)$, of a point process can be thought of as the conditional probability that there is a point at u given the rest of the process, where x is the configuration of the process.

A point process X is *Markov of range* r if its Papangelou conditional intensity, $\lambda(u, x)$, is dependent only on the region within r of u and its density is well behaved in a certain sense.

2.4 Examples

Due to the abstract nature of the previous three sections we now present some examples to provide the reader with a little intuition about the concepts introduced above. We begin with topology.

If we begin with the real number line, $\mathbb{R}$, then the sets which we normally think of as open (i.e. open intervals, collections of open intervals, etc.) form a topology over $\mathbb{R}$ and so $\mathbb{R}$ together with those sets forms a topological space. If we then take the interval $[0, 1]$ together with the open sets in $[0, 1]$ then that forms a subspace of the previous topological space. Examining the open interval $(\frac{1}{4}, \frac{3}{4})$ we see that the points $\frac{1}{4}$ and $\frac{3}{4}$ are (all of the) limit points for this set which are not also members of it, and so the closed interval $[\frac{1}{4}, \frac{3}{4}]$ is the closure of this set. It is then easy to see that the union of all open intervals of the form $(k, k + 1)$ for $k \in \mathbb{Z}$ is dense in $\mathbb{R}$. This is not, however, a countable set so we have not done enough to show that this topological space is separable. In fact it is separable, as $\bar{\mathbb{Q}} = \mathbb{R}$ and $\mathbb{Q}$ is

countable. It is also locally compact and Hausdorff, but is not compact.

We now move on to look at the myope topology induced by the usual topology over $\mathbb{R}^2$. An example of a continuous function $f : \mathbb{R}^2 \to \mathcal{K}$ is any constant function, for example:

$$f(\mathbf{x}) = B(\mathbf{0}; 1),$$

where $B(\mathbf{0}; 1)$ is the closed ball centred at the origin with radius 1. The inverse image of this function for any open set in $\mathcal{K}$ is clearly either the empty set or $\mathbb{R}^2$, both of which are open in the usual topology[3]. A more useful example of a myopically continuous function is Minkowski addition of a constant set, i.e.

$$f(x) = x \oplus G = \{y \in \mathbb{R}^2 : y = x + a \text{ for } a \in G\},$$

where $G \subset \mathbb{R}^2$ is some fixed set.

Moving on to measures now, if we take $\mathbb{Z}$ as our set then the set of all subsets of $\mathbb{Z}$ (denoted $2^{\mathbb{Z}}$) is a σ-field, so $(\mathbb{Z}, 2^{\mathbb{Z}})$ is a measurable space. Note also that $T = \{\mathbb{Z}, 2^{\mathbb{Z}}\}$ is a topological space[4], and so $2^{\mathbb{Z}}$ is the class of Borel sets of T. The set function $\mu : 2^{\mathbb{Z}} \to \mathbb{R}^+ \cup \{+\infty\}$ defined by

$$\mu(X) = n(X),$$

where $n(X)$ is the number of points in X, is a measure over this space[5] so that $(\mathbb{Z}, 2^{\mathbb{Z}}, \mu)$ is a measure space. The measure μ is not finite since if $X = \{x \in \mathbb{Z} : x > 0\}$ then $\mu(X) = \infty$, but it is clearly regular since under the discrete topology[4] every subset of $\mathbb{Z}$ is both open and closed.

As an example of the application of the Radon-Nikodým theorem let μ be counting measure over $\mathbb{Z}^+$ and ν be Poisson measure with parameter λ (i.e. $\nu(x) = e^{-\lambda}\frac{\lambda^x}{x!}$ for $x \in \mathbb{Z}^+$). Then $\nu \ll \mu$ (since the empty set is the only set with zero counting

[3] The reader should note that although we have used a constant function which maps to a set which is closed in the topology on $\mathbb{R}^2$ we could have used *any* constant function which maps to a compact set and the result would still hold.

[4] For any set X, the set 2^X is a topology over X and is called the *discrete* topology over X. $T = \{X, 2^X\}$ is called a *discrete space*.

[5] This simple measure is usually called *counting measure*.

measure and $\nu(\phi) = 0$) and the Radon-Nikodým derivative of ν with respect to μ is simply

$$e^{-\lambda}\frac{\lambda^x}{x!}.$$

In this instance it is unique since $\mu(X) > 0$ whenever X is nonempty. As is probably now obvious, Radon-Nikodým derivatives of probability measures with respect to counting measure over a suitable subset of the integers are normally called *probability mass functions*.

Finally, as a second example of the application of the Radon-Nikodým theorem, let ρ denote standard Lebesgue measure and ν denote the Gaussian measure with mean μ and variance σ^2. Then $\nu \ll \rho$ and

$$\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}$$

is (a version of) the Radon-Nikodým derivative of ν with respect to ρ on $(\mathbb{R}, \mathcal{F})$ where $\mathcal{F}$ is the Borel σ-field over $\mathbb{R}$ with respect to the standard topology. We usually call Radon-Nikodým derivatives of probability measures with respect to Lebesgue measure on $(\mathbb{R}, \mathcal{F})$ *probability density functions*.

Chapter 3

Perfect Simulation

3.1 Introduction

In the field of Markov Chain Monte Carlo (MCMC) (see Brooks (1998) for a good introduction to the field and literature review) it is often useful to be able to determine whether a Markov Chain has reached its *stationary* (or *equilibrium*) distribution. Perfect simulation (sometimes called exact sampling[1]) is a method for ensuring that a Markov chain has reached its stationary distribution.

It has long been considered a failing of MCMC that one can rarely be absolutely sure that the Markov chain which is used for a given simulation has converged to its stationary distribution. This means that it is not possible to generate an unbiased sample[2]. It would be nice, therefore, to find a method for *guaranteeing* that the chain has reached equilibrium, and thus that the resulting sample will be unbiased. In 1992, Asmussen et al. proved that this was possible, provided that the number of states of the Markov chain was known. It was not until 1995, however, that a serious algorithm was developed which converged in polynomial time (see Lovász and Winkler (1995)). Most of this work went largely ignored by the statistical community, until Propp and Wilson (1996) produced their now

[1] The almost interchangeable way the words simulation and sampling are used in the literature may be confusing to some readers. The distinction between the two may be made clearer by the following two statements: We use *simulation* to generate a *sample* from a given distribution. *Sampling* is the process of generating one or more *samples* using *simulation*.

[2] A *biased sample* is one whose distribution is different from the equilibrium distribution of the Markov chain used to generate it, so that the estimate of any quantity depending on the equilibrium distribution may be biased.

famous paper, introducing an algorithm which they called *coupling from the past*. A sudden explosion of papers ensued, and since that time a multitude of papers have been written on the subject. This chapter has two purposes. Firstly, it aims to be an introduction to coupling from the past, and secondly it it aims to be a brief literature review of some of the work which has been done since the publication of Propp and Wilson's original paper. This survey is not intended to be exhaustive, but merely covers those areas which peaked the author's interest.

3.2 Coupling From The Past

Coupling From The Past, or CFTP as it has become known, is an exact sampling algorithm for discrete ergodic Markov chains with a finite number of states n, and was introduced to the literature by Propp and Wilson (1996, 1998). Although attempts have been made to generalise the algorithm to continuous state spaces (notably by Murdoch and Green (1998) and Green and Murdoch (1998)), there is still much work to be done before exact sampling becomes universally, or even generally applicable (for example no truly general methods which work in high, or even moderate, dimensions exist).

The motivation behind CFTP is the following: Suppose that it is desirable to sample from the stationary distribution of an ergodic Markov chain $\{Z_t\}$ on some (finite) state space X with states $1, \ldots, n$. It is clear that if it were possible to go back an infinite amount in time, start the chain running (in state $Z_{-\infty}$) and then return to the present, the chain would (with probability 1) be in its stationary distribution when one returned to the present (i.e. $Z_0 \sim \pi$, where π is the stationary distribution of the chain).

Unfortunately, owing to the impossibility of going back an infinite amount in time, this is not a practical solution to the problem of generating the required sample. However, an alternative is possible, using a method known as *coupling*.

3.2.1 Coupling

The mathematical definition of coupling is the following (taken from Lindvall (1992)):

A *coupling* of the probability measures P and $\tilde{P}$ on a measurable space $(\Omega, \mathcal{B})$ is a probability measure $\hat{P}$ on $(\Omega^2, \mathcal{B}^2)$ such that

$$P = \hat{P}\pi^{-1} \quad and \quad \tilde{P} = \hat{P}\tilde{\pi}^{-1}$$

where $\pi(x, \tilde{x}) = x$ and $\tilde{\pi}(x, \tilde{x}) = \tilde{x}$ for $(x, \tilde{x}) \in \Omega^2$.

For our purposes, however, we need only consider the following interpretation:

Given a transition rule $f(\cdot, U)$ for a Markov transition kernel K, two Markov chains X and Y obeying the transition kernel K are *coupled* if the same random events U determine the outcome of the transition rule f.

One consequence of this definition is that if at any time s we have $X_s = Y_s$ then $X_t = Y_t \quad \forall t \geq s$. This fact is crucial to the development of coupling from the past.

We now return to the issue of how we simulate a chain which has been running for an infinite amount of time. Suppose we were to set not one, but n chains $\{Z_t^{(1)}\}, \ldots, \{Z_t^{(n)}\}$ running at a fixed time $-M$ in the past, where $Z_{-M}^{(i)} = i$ for each chain $\{Z_t^{(i)}\}$, and there are n states in our state space. Now let all the chains be coupled so that if $Z_s^{(i)} = Z_s^{(j)}$ at any time s then $Z_t^{(i)} = Z_t^{(j)} \quad \forall t \geq s$. Then if all the chains ended up in the same state j at time zero (i.e. $Z_0^{(i)} = j \quad \forall i \in X$), we would know that whichever state the chain passing from time minus infinity to zero was in at time $-M$, the chain would end up in state j at time zero. Thus j must be a sample from the stationary distribution of the Markov chain in question.

Suppose for example that the Markov chain which has been running since time $-\infty$ were in state k at time $-M$. Then since both this chain and $\{Z_t^{(k)}\}$ use the

same transition law, and they are in the same state at time $-M$, they will be in the same state at time 0 as well. Thus since all of our Markov chains $\{Z_t^{(1)}\}, \ldots, \{Z_t^{(n)}\}$ are in the same state j at time 0, and one of them must also be in the same state as the chain which was started at time $-\infty$, then j must be a sample from the stationary distribution of the Markov chain.

Owing to the difficulty of explaining this process in abstraction I will now proceed with an example.

3.2.2 A Simple Example

Here we consider the case where our state space has only four states, labelled 1, 2, 3 and 4. Let the transition probabilities for this chain when it is in some state i be $\frac{1}{2}$ that the next state will be state $i+1$ (where $i+1 \equiv i$ if $i=4$), and $\frac{1}{2}$ that the next state will be state $i-1$ (where $i-1 \equiv i$ if $i=1$). We simulate exactly from the stationary distribution of this process by implementing the following algorithm:

1. First of all, we pick some initial value of time, M (for the purposes of this example we have chosen $M=1$).
2. Now consider four Markov chains, $\{Z_t^{(1)}\}$, $\{Z_t^{(2)}\}$, $\{Z_t^{(3)}\}$ and $\{Z_t^{(4)}\}$, starting at time $t=-M$ and running to time $t=0$. We start one in each of the four states. We couple the four chains using the following rule:
   (a) First simulate the transitions for the first chain $\{Z_t^{(1)}\}$ (by tossing a coin, for example).
   (b) Next, use the transitions of $\{Z_t^{(1)}\}$ to determine the transitions of the other three chains as follows:
      i. If $Z_{t+1}^{(1)} = Z_t^{(1)} + 1$, or $Z_{t+1}^{(1)} = Z_t^{(1)}$ and $Z_t^{(1)} = 4$ (i.e. the chain goes 'up'), then $Z_{t+1}^{(i)} = Z_t^{(i)} + 1$ for $i = 2, \ldots, 4$ unless $Z_t^{(i)} = 4$, in which case $Z_{t+1}^{(i)} = Z_t^{(i)}$.
      ii. If $Z_{t+1}^{(1)} = Z_t^{(1)} - 1$, or $Z_{t+1}^{(1)} = Z_t^{(1)}$ and $Z_t^{(1)} = 1$ (i.e. the chain goes 'down'), then $Z_{t+1}^{(i)} = Z_t^{(i)} - 1$ for $i = 2, \ldots, 4$ unless $Z_t^{(i)} = 1$, in which case $Z_{t+1}^{(i)} = Z_t^{(i)}$.

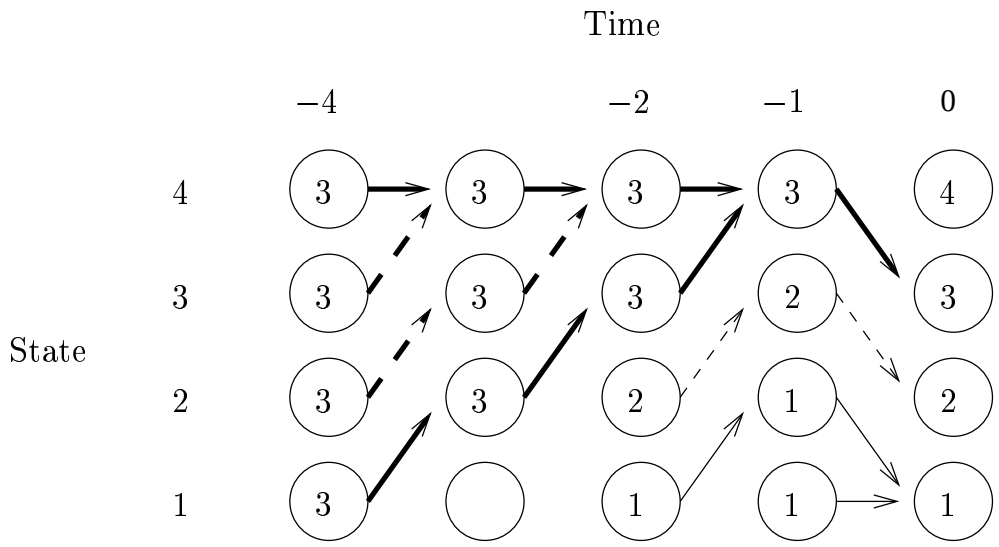

Figure 3.1: A simulation of the example of CFTP described in the text.

3. Finally, examine the state of the four chains at time 0. If all four chains are in the same state then we are done, and the state of the four chains at time 0 is an exact sample from the stationary distribution of the Markov chain. Otherwise, double M and then return to step 2, *retaining the transitions already generated*.

Note that at any given time t, we only generate *one* random event. This explains why two coupled random variables, governed by probability measures P and $\tilde{P}$ in the definition on page 33, are said to be constructed on the same probability space and governed by a single probability measure $\hat{P}$.

The example we have just described was simulated by tossing a coin to determine the transitions of the chains, letting the occurrence of a head force the coupled chains to go 'up', and the occurrence of a tail force the coupled chains to go 'down'. The results of one simulation of this process are shown in Figure 3.1. The numbers in the circles refer to the state which the chains passing through that state at that time end in at time $t = 0$.

The simulation began at time $t = -1$, when a tail was thrown. As can be seen in Figure 3.1, this did not result in coalescence of all the chains, so we then went back to $t = -2$, where a head was tossed. We retained the tail we had tossed at time $t = -1$, and again we see that all of the chains had not coalesced at time $t = 0$, so we then went back to $t = -4$. Here we tossed a head, then a head at $t = -3$ and we retained the head which was tossed at $t = -2$ and the tail which was tossed at $t = -1$. By examining Figure 3.1 once more we see that this time coalescence has occurred. Thus a sample from the stationary distribution of this Markov chain is $\{3\}$.

The reason that there is no number in one of the circles (and no arrow coming from it) is that at no point during the simulation did a chain reach state 1 at time -3. It can be seen in the diagram that the chain actually coalesced at $t = -1$ to state 4. It is important to note, however, that we must not take this point as our sample, as this would not give an unbiased sample. This can be seen clearly in our example, as the chain will *always* coalesce in either state 1 or state 4. This is the reason why we cannot simply start our simulation (at time $t = 0$), wait for all the chains to coalesce, and then assume that we will be generating an unbiased sample from that point on.

The reason we have doubled $-M$ each time the chains have not coalesced is that (as Propp and Wilson (1996) show) for simple algorithms like this one, doubling is close to (within a factor of four of) optimal if we wish to minimise the total running time of the algorithm (which seems like a sensible goal). Our example also exposes something else about CFTP, which is that when the coupling is *monotone* (i.e. when $Z_t^{(i)} \geq Z_t^{(j)}$, then $Z_{t+k}^{(i)} \geq Z_{t+k}^{(j)}$ for all positive k), we need only simulate the top and bottom paths — the others are superfluous. This is indicated in Figure 3.1 by representing the superfluous paths by dashed arrows rather than solid ones. It is precisely this monotone case where Propp and Wilson prove optimality of the doubling scheme for choice of $-M$. In general the question of which scheme to adopt for increasing M between iterations is very difficult to answer (see the

discussion in Green and Murdoch (1998) and Section 6 of Murdoch and Rosenthal (2000)).

3.3 CFTP For Continuous State Spaces

A limited amount of work has been done on exact sampling from a continuous state space. In a paper published in 1998, Murdoch and Green presented a number of methods for sampling exactly from a continuous state space, but their couplers are rather impractical, and really serve as little more than existence proofs, as the authors themselves admit in a follow-up article (Green and Murdoch 1998). In that follow-up paper they present algorithms which seem rather more practical in a (very) small number of dimensions, although there is still a great deal of work to be done before anything which could routinely be used as part of MCMC is developed.

The following methods all exploit ways in which an infinite number of coupled Markov chains can be made to coalesce into a finite number of states in a single step. The first and third are based on partitioning the update function into two pieces (one of which does not depend on the current position of a given chain and the other which does), while the second makes use of rejection sampling[3] on a large number of chains at the same time.

3.3.1 The Multigamma Coupler

The Multigamma Coupler[4] only works when two conditions are satisfied. Firstly, we must know the update kernel of the chain $f(\cdot|x)$, although this is not usually a problem. Secondly, the update kernel must be uniformly bounded from below in x by some non-negative function $r(y)$ for all y in χ, the state space of the chain,

[3] See, for example, Gamerman (1997) page 24 for an introduction to rejection sampling.

[4] It has been argued that the Multigamma coupler might have been called Doeblin's Coupler due to the similarity between the minorisation condition given implicitly in (3.1) and Doeblin's small sets (see Meyn and Tweedie (1993) pp. 100–110). However this concept has also been used by several others in the literature and the authors follow the lead of Lindvall (1992) by not giving the honours to any one person.

where $r(\cdot)$ is *independent of the current state x of the chain.*

If these conditions apply then we can write $f(\cdot|x)$ in terms of two distributions — one which is dependent on the current position of the chain and one which is not. For example in the one dimensional case we can write

$$F(y|x) = \rho R(y) + (1-\rho)Q(y|x) \tag{3.1}$$

where $F(y|x) = \int_{-\infty}^{y} f(v|x)dv$ is the cumulative density function of y given x, $\rho = \int r(v)dv$, and

$$R(y) = \rho^{-1}\int_{-\infty}^{y} r(v)dv \quad \text{and} \quad Q(y|x) = (1-\rho)^{-1}\int_{-\infty}^{y} \{f(v|x) - r(v)\}\, dv.$$

If the inverses of R and $Q(\cdot|x)$ are available, the update function of the CFTP algorithm is

$$\phi(x,U) = \begin{cases} R^{-1}(U^{(2)}) & \text{if } U_t^{(1)} < \rho \\ Q^{-1}(U^{(2)}|X_{t-1}) & \text{otherwise,} \end{cases}$$

where $U = (U^{(1)}, U^{(2)})$ is a pair of independent $U[0,1]$ random numbers. This is because R and $Q(\cdot|x)$ are both cumulative density functions, and $P(F^{-1}(U) \leq y) = P(U \leq F(y)) = F(y)$, for any cumulative density function F.[5] The fact that the update function is as given above means that whenever we use the distribution R, every chain is mapped to a *single point*. From this fact, it is clear that the time taken for all paths in the algorithm to coalesce is distributed geometrically with parameter ρ (since at each time T at which all the chains have not all coalesced to a single point, the probability of all the chains coalescing at the next iteration is ρ).

In practice, it is often not practical to attempt to sample from Q before coalescence, so if the chains have not coalesced, and the decision *not* to sample from R is made, then we merely proceed to the next step of the algorithm, and act as if $\{Q^{-1}(U^{(2)}|X_{t-1}) : X_{t-1} \in \chi\} = \chi$, whether that was the case or not. The update

[5] This an application of the 'method of inverses'. See Ross (1990) pp. 59–60.

function for the CFTP algorithm is thus

$$\Phi(B_{t-1}, U_t) = \begin{cases} R^{-1}(U_t^{(2)}) & \text{if } U_t^{(1)} < \rho \\ Q^{-1}(U_t^{(2)}|X_{t-1}) & \text{if } U_t^{(1)} \geq \rho \text{ and } B_{t-1} \neq \chi \\ \chi & \text{otherwise,} \end{cases}$$

where B_{t-1} is the set of values which we are considering in step $t-1$ of the algorithm.

The fact that coalescence time is geometrically distributed gives an efficient method of using multigamma coupling. Begin the CFTP algorithm at some time $-T$ ($\lfloor 1/\rho \rfloor$ is probably a reasonable choice). First generate a single Geom(ρ) random variate, G_0. If $G_0 \leq T$ then coalescence has occurred. Generate one draw from R. Using this as a starting value, iterate the chain $T - G_0$ times using only draws from Q. The result is a draw from the required distribution. If $G_0 > T$, generate a second Geom(ρ) random variate G_1. If $G_1 \leq T$ do as above, except we must now iterate the chain $2T - G_1$ times. Repeat this procedure until a value $G_j \leq 2^{j-1}T$ is obtained.

The requirement $G_j \leq 2^{j-1}T$ is used rather than, as might have been expected, $G_j \leq 2^jT$ because due to previous iterations, we *know* that the chain has not coalesced between time $-2^{j-1}T$ and time 0. Thus the requirement must be for coalescence between time -2^jT and time $-2^{j-1}T$, i.e. $G_j \leq 2^{j-1}T$ as stated.

The condition $f(\cdot|x) \geq r(y) \quad \forall x, y \in \chi$ can be relaxed slightly to allow for uniform (lower) boundedness on each of a finite collection of (disjoint) sets $\{A_i\}$, where $\bigcup_i A_i = \chi$. For ease of notation the authors also assume that $\rho = \int r_i(v)dv$ is constant for each A_i, although clearly this constraint can be relaxed in practice. With these slightly modified constraints the update function for the CFTP

algorithm becomes

$$\Phi(B_{t-1}, U_t) = \begin{cases} \bigcup_{i:A_i \cap B_{t-1} \neq 0} \left\{ R_i^{-1}(U_t^{(2)}) \right\} & \text{if } U_t^{(1)} < \rho \\ \bigcup_i \left\{ Q_i^{-1}(U_t^{(2)}|x) : x \in A_i \cap B_{t-1} \right\} & \text{if } U_t^{(1)} \geq \rho \ \& \ B_{t-1} < \infty \\ \chi & \text{otherwise.} \end{cases}$$

A modified version of the method using geometric random variates can be used here too, although by necessity it is considerably more complicated. See the discussion in Green and Murdoch (1998) for further details.

From a method where it was necessary to find a uniform lower bound for the transition kernel at each point, we now turn to Murdoch and Green's next method, which requires a uniform *upper* bound.

3.3.2 Rejection Coupling

This next method of coupling introduced to the literature by Murdoch and Green is called rejection coupling because of its strong links to rejection sampling (see Section 3.5 for a brief overview of rejection sampling). In fact, this method is really just rejection sampling for a large set of distributions simultaneously, rather than just one. It is perhaps ideally suited to MCMC, since it is only necessary to know the updating densities of the Markov chains up to a multiplicative constant, which is often the case in MCMC. As with ordinary rejection sampling, it is necessary for there to exist a density $h(y)/\nu$, where $\nu = \int h(y)dy$ is finite, such that if $g(y|x)$ is the un-normalised update density then $g(y|x) \leq h(y)$ for every $y \in \chi$. However in rejection coupling it is also required that the function $h(y)$ bound $g(y|x)$ *for every value of* x (where x is the current position of the Markov chain to be updated).

The idea of this coupler is simple: First sample from $h(y)/\nu$, and independently sample from a $U[0,1]$ distribution. Let Y be the point from the first sample, and U be the point from the second. Then for each value x of χ we perform rejection sampling using the point $(Y, Uh(Y))$ as our possible update. Figure 3.2 gives an

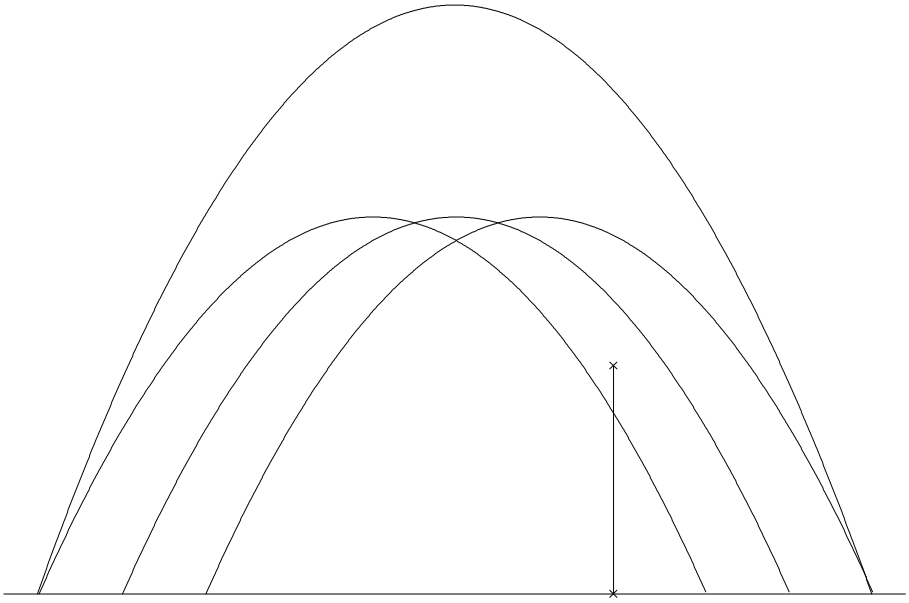

Figure 3.2: An example of rejection coupling: The large outer curve represents the upper bound $h(y)$, the other three curves represent $g(y|x)$ for three specific values of x. The two points joined by a line show the generation of $(Y, Uh(Y))$. The lower point is $(Y, 0)$ and the upper is $(Y, Uh(Y))$. In this case the algorithm will accept the point Y as a valid update for the last two values of x, but not for the first, as the point $(Y, Uh(Y))$ is outside the first curve.

example of this for three values of x. We now sample repeatedly as above until we have accepted an update for every value of x in χ. The set of all accepted Y's is then the set of x's which we consider in the next step of the CFTP algorithm.

The authors also give a number of simple examples to demonstrate the application of these methods. We now proceed with a more practical coupler presented in their second paper (Green and Murdoch 1998) which is based on a clever application of the multigamma coupler to random walk Metropolis sampling

3.3.3 The Bisection Coupler — a special case of the multigamma coupler

As mentioned above the bisection coupler is really just a special case of the partitioned multigamma coupler where we partition the state space into a clever set of pieces. It is applicable whenever the state space χ is a rectangle in any number of dimensions. For the purposes of clarity we consider the case $\chi = [0, 1]$. Suppose

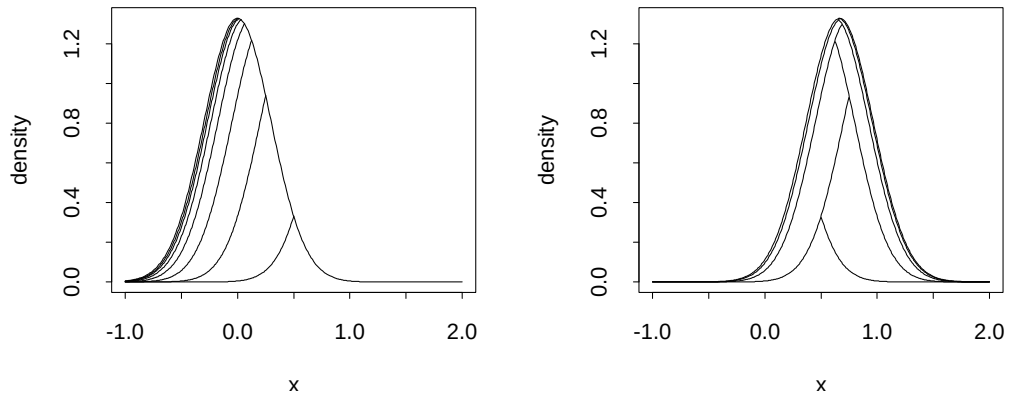

Figure 3.3: An illustration of the bisection coupler. The graphs shown are the decompositions of the $N(0, 0.3^2)$ and $N(2/3, 0.3^2)$ which the algorithm exploits. Reproduced with kind permission of P. J. Green.

that the update kernel $q(\cdot)$ for the Metropolis-Hastings algorithm is a unimodal symmetric random walk (see Gamerman (1997) pp. 166-7). Then it can be shown that we can cut the area under the graph of q into pieces given by overlaps with the graphs of $q(\cdot - 2^{-k})$ for $k \in \mathbb{Z}^+$ and reconstruct the pieces to represent $q(\cdot - x)$ for any $x \in \chi$. We can then sample from under the graph of $q(\cdot)$ to get updates for each $x \in \chi$ by allowing the update for a given point x to be the point under the graph of $q(\cdot - x)$ (which has been constructed from the graph of $q(\cdot)$) which corresponds under the reconstruction to the point which was drawn from $q(\cdot)$. The fact that each partition of $q(\cdot)$ occurs in only a finite number of positions (if it is the partition given by overlap with $q(\cdot - 2^{-k})$ then it only occurs in 2^k different positions) means that the transitions for chains started everywhere in $[0, 1]$ will (almost surely) only result in a finite number of positions after the first transition. If the number of positions is unreasonably large, the usual trick of just pretending that no coalescence at all has occurred is used (see Section 3.3.1 above) to save on both computer time and memory. See Figure 3.3 for an example of a decomposition and reconstruction with Normal proposal densities.

3.3.4 Further results using CFTP on continuous state spaces

The work by Green and Murdoch is not the only work which has attempted to use coupling from the past to sample from general distributions of interest in Bayesian statistics. Other work includes (Møller and Nicholls 1999), which makes use of simulated tempering and seems to be the most generally applicable work so far; (Mira, Møller, and Roberts 2001), which exploits the monotonicity properties of the slice sampler; and (Hobert et al. 1999), which looks at perfect simulation of mixture distributions with two or three components.

3.4 Perfect Simulation For Spatial Processes

We cover perfect simulation of spatial point processes in some depth in Chapter 4. Here we give a brief overview of the field.

Since spatial point processes have a natural partial ordering induced by the number of points in a given region of interest[6], both monotone CFTP and Fill's algorithm (covered in Section 3.5) have been successfully applied to the simulation of these processes. The problem that there are in general no maximum elements in these processes is overcome using the concept of *stochastic domination*, which is similar in concept to stochastic monotonicity introduced on page 36. Recall that a Markov chain (or process) is stochastically monotone with respect to a particular partial ordering $\leq$ of the state space if, given two realisations X^1 and X^2 such that $X_t^1 \leq X_t^2$, then $X_{t+s}^1 \leq X_{t+s}^2$ for every positive s. A Markov chain (or process) Y stochastically dominates a second Markov chain (or process) X with respect to a particular partial ordering $\leq$ of the state space, if $X_t \leq Y_t$ implies that $X_{t+s} \leq Y_{t+s}$ for every positive s.

The first work on applying perfect simulation to spatial point processes was by Kendall (1998), where the idea of using stochastic domination to overcome the

[6] More precisely, two point configurations x and y are related by the partial ordering $x \preceq y$ if $x \subseteq y$

absence of a maximum element was introduced. This work discussed a method of simulating the area-interaction process of Baddeley and van Lieshout (1995). The use of stochastic domination earned this technique the name *dominated CFTP*. The work of this paper is discussed in some depth in Chapter 4.

Häggström et al. (1999) found an alternative way of sampling the attractive area-interaction process exactly by sampling the penetrable sphere model (Widom and Rowlinson 1970) using a two-component Gibbs sampler. They also successfully simulated the continuum random-cluster model, a point process model which has the number of connected components as the weighting exponent. Thönnes (1999) then used the ideas of Häggström et al. (1999), but applied Fill's algorithm (Fill 1998) rather than CFTP in order to obtain a perfect simulation algorithm which is free of *user-impatience bias*. See Fill (1998) for a discussion of this subtle bias introduced by terminating long runs of the algorithm before an exact sample is obtained. We discuss Fill's algorithm in Section 3.5.

In a paper which was actually published before his initial paper, Kendall (1997) extended his work to the case where the weighting exponent is something called a *quermass integral*. This basically means that rather than using the area of the union of the grains of the points in the process as the weighting exponent, either the *perimeter length* or the *Euler functional* of the union of the grains of the points in the process is used. The Euler functional is equal to the number of connected components minus the number of holes in those components, so this model is similar to the continuum random-cluster model considered by Häggström et al. (1999). One slight weakness of this work is that most of it only works for the two dimensional case.

Kendall and Møller (1999) generalised the work in Kendall (1998) to all *locally stable* point processes, i.e. all point processes satisfying the inequality

$$f(X \cup \{x\}) \leq Kf(X), \tag{3.2}$$

where f is the density of the distribution with respect to the unit rate Poisson

process, $X \subseteq \Omega$ is some finite configuration, K is a finite constant and $x \in \Omega \setminus X$ (Ω is some compact window). Clearly equation (3.2) simply states that the Papangelou conditional intensity must be uniformly bounded. This is the case for many point processes including the Strauss process and the area-interaction process, amongst others. For a more complete list see Kendall and Møller (1999).

Further work making use of perfect simulation of spatial point processes has been done by Lund and Thönnes (2000), who discuss a Bayesian model of a noisily observed point configuration; by Loizeaux (2001), who discusses a Bayesian model for finding cluster-centres underlying a data set; and by Berthelsen and Møller (2001), who discuss inference for pairwise interaction point processes. Fernández et al. (1999) also introduce a new algorithm for perfect simulation which does not rely on any aspects of stochastic monotonicity.

3.5 Fill's Algorithm

Fill's algorithm (Fill 1998) is an alternative method of perfect sampling based on rejection sampling.

Rejection sampling is a method of obtaining a sample from some density $\pi(x)$ by sampling from a second density $p(x)$ which is easier to sample from and for which there exists a constant C such that

$$\frac{\pi(x)}{p(x)} \leq C \qquad \forall x \in \Omega.$$

A sample X is drawn from p together with an auxiliary sample U from a $U[0,1]$ distribution. The sample is accepted as a draw from π if

$$U \leq \frac{\pi(X)}{Cp(X)}. \tag{3.3}$$

See Gamerman (1997) Section 1.5.1 for further details.

Fill's algorithm is less general than CFTP because it requires that the state space Ω be finite and for there to exist a partial ordering $\leq_\Omega$ such that the time reversal $\tilde{P}$ of the transition matrix P of the Markov chain is monotone with respect to it.

There must also exist unique maximal and minimal elements $\hat{1}$ and $\hat{0}$ respectively. Finally, we require a measure

$$K_{(x,y)}(x', y') = \mathbb{P}(\tilde{f}(y, U) = y' | \tilde{f}(x, U) = x'),$$

where $\tilde{f}(\cdot, U)$ is a monotone transition rule for $\tilde{P}$ such that we can sample from $K_{(x,y)}(x', \cdot)$ whenever $\tilde{P}(x, x') > 0$ and $x \leq_\Omega y$.

The algorithm is a simple 3-step procedure:

1. Run the chain X forward in time using P for N steps with $\hat{0}$ as the initial state to obtain a sample z of $P^N(\hat{0}, \cdot)$, recording the trajectory $(X_0 = \hat{0}, \ldots, X_N = z)$.

2. Reverse the direction of the trajectory obtained in step 1 to obtain a trajectory $(\tilde{X}_0 = z, \ldots, \tilde{X}_N = \hat{0})$ of $\tilde{P}$ conditioned on starting in state z and ending in state $\hat{0}$.

3. Simulate a second Markov chain Y using the measure $K_{(\tilde{X}_i, Y_i)}(\tilde{X}_{i+1}, \cdot)$ where the initial state Y_0 of Y is $\hat{1}$.

If the final state Y_N of Y is $\hat{0}$ then z is accepted as a draw from π. If not, double N and start again independently of the trajectories obtained in the previous run.

3.5.1 Proof of the validity of Fill's algorithm

Fill's algorithm works because $\mathbb{P}(Y_N = \hat{0} | \tilde{X}_0 = z, \tilde{X}_N = \hat{0}, Y_0 = \hat{1})$ is the acceptance probability for rejection sampling where (by comparison with equation (3.3)) $P^N(\hat{0}, \cdot)$ is the density p, z is the sample X and $\pi(\hat{0}) / \tilde{P}^N(\hat{1}, \hat{0})$ is the constant C.

The expression $\pi(\hat{0}) / \tilde{P}^N(\hat{1}, \hat{0})$ is a suitable upper bound for $\pi(z) / P^N(\hat{0}, z)$ because from the detailed balance criterion[7], $\tilde{P}$ is given by

$$\tilde{P}(x, y) = \frac{\pi(x) P(y, x)}{\pi(y)}. \tag{3.4}$$

[7] The reader should note that the detailed balance condition used in equation (3.4) is *not* the same as that given in equation (4.4) on page 57. Equation (4.4) deals with transition *rates*, whereas equation (3.4) deals with transition *probabilities*.

for x s.t. $\pi(x) > 0$ and thus

$$\frac{\pi(z)}{P^N(\hat{0}, z)} = \frac{\pi(\hat{0})}{\tilde{P}^N(z, \hat{0})} \leq \frac{\pi(\hat{0})}{\tilde{P}^N(\hat{1}, \hat{0})}$$

by monotonicity. Hence the acceptance probability is given by

$$\frac{\tilde{P}^N(\hat{1}, \hat{0})}{\pi(\hat{0})} \times \frac{\pi(z)}{P^N(\hat{0}, z)} = \frac{\tilde{P}^N(\hat{1}, \hat{0})}{\pi(\hat{0})} \times \frac{\pi(\hat{0})}{\tilde{P}^N(z, \hat{0})} = \frac{\tilde{P}^N(\hat{1}, \hat{0})}{\tilde{P}^N(z, \hat{0})}.$$

Now

$$\begin{aligned} \mathbb{P}(Y_N = \hat{0} | \tilde{X}_0 = z, \tilde{X}_N = \hat{0}, Y_0 = \hat{1}) &= \frac{\mathbb{P}(Y_N = \hat{0}, \tilde{X}_N = \hat{0} | \tilde{X}_0 = z, Y_0 = \hat{1})}{\mathbb{P}(\tilde{X}_N = \hat{0} | \tilde{X}_0 = z, Y_0 = \hat{1})} \\ &= \frac{\mathbb{P}(Y_N = \hat{0} | \tilde{X}_0 = z, Y_0 = \hat{1})}{\mathbb{P}(\tilde{X}_N = \hat{0} | \tilde{X}_0 = z)}, \end{aligned}$$

since the chain is monotone and X is independent of the starting point of Y. Furthermore,

$$\frac{\mathbb{P}(Y_N = \hat{0} | \tilde{X}_0 = z, Y_0 = \hat{1})}{\mathbb{P}(\tilde{X}_N = \hat{0} | \tilde{X}_0 = z)} = \frac{\tilde{P}^N(\hat{1}, \hat{0})}{\tilde{P}^N(z, \hat{0})},$$

the correct acceptance probability.

3.5.2 Applications and extensions of Fill's algorithm

Like CFTP, one of the drawbacks to the original version of Fill's algorithm is that it is only applicable to finite state spaces. This restriction has been partially overcome in two papers. Firstly, as mentioned in Section 3.4, Thönnes (1999) used Fill's algorithm to simulate the penetrable spheres mixture model using the work of Häggström et al. (1999) to obtain the required monotonicity. Using similar methods to Thönnes (1999), Møller and Schladitz (1999) applied Fill's algorithm to discrete but not necessarily finite models (for example lattice models) which are specified conditionally, i.e. the distribution of some random vector $X = (X_i)_{i \in I}$ is given by the conditional distributions $P(X_i | \{X_j : j \neq i; i, j \in I\})$. The obvious example of this type of model is the auto-Poisson model (see Cressie (1993), chapter 6), and indeed this is one of the simulation examples given in the paper.

A further restriction of Fill's algorithm is that is only works if the time-reversed chain is monotone. Fill et al. (2000) have extended the algorithm for general

chains, although the state space must still be finite. Huber (1999) uses these results to sample from the stationary distribution some non-monotone chains and develops some theory pertaining to the running time of Fill et al.'s algorithm in comparison to CFTP.

3.6 Other Perfect simulation algorithms

Coupling from the past and Fill's algorithm are not the only perfect simulation algorithms, merely the first and most widely used. Another is the method of Fernández et al. (1999) for spatial point processes and lattice models mentioned in Section 3.4, which uses the same spatial birth and death model as its basis as the dominated coupling from the past algorithm of Kendall (1998), but requires no monotonicity. Yet another is the method of Cai (1999), which makes use of auxiliary variables and uses a single run of a Markov chain, out of which exact samples are picked. Finally, Fill and Huber (2000) introduced a method called the Randomness Recycler, which generates perfect samples, but without using the traditional Markov chain framework.

3.7 Conclusions

During this chapter we have seen that Coupling From The Past shows great promise of solving the problem of stationarity in Markov Chain simulation. There is, however, much work remaining to be done if it is to truly fulfil this promise. As yet, it still only really works effectively when the state space is finite, or when simulating certain spatial point processes. In general when the state space is infinite there have been a number of nice ideas (which effectively reduce the infinite state space to a finite one by using a variety of tricks), but nothing truly revolutionary. So far the signs are that CFTP may fail to deliver on its promising beginning.

One of the main problems is the enormous amount of work, both computational and analytic, which goes into the generation of a single exact sample. The question

is, "Is it worth it?" Is the overhead of great analytic and computational effort which goes into generating a single exact sample so restrictive as to make the technique effectively worthless in the long run?

In the general Bayesian setting I am tempted to say "yes" to this last question, although in particular situations where there is a lot of natural stochastic monotonicity (as in the spatial work by Kendall and others) dominated CFTP still shows some promise (aided by the fact that in the spatial context it is often useful even to have just a small number of exact samples). This idea of stochastic monotonicity may, however, be the key. If it were possible to set up a method like the birth-death method (see Sections 4.1.3 and 4.1.4) in the spatial case which used easy-to-sample draws from a distribution with heavier tails than the desired distribution and used an accept/reject rule to generate a sample from the true distribution, then it is possible that dominated CFTP could prove more generally useful. Sadly, this has turned out to be more difficult than was initially hoped.

Chapter 4

Spatial Point Processes

Spatial point processes are all around us. From the distribution of people in a room or particles in a box, to the locations of cities across the world and even the relative positions of stars in the universe, an enormously wide spread of phenomena can be modelled as a random configuration of points in two or three dimensional space. In this chapter we begin by introducing the reader to the simplest spatial point process model, the Poisson process, before moving on to a more complex model, the area-interaction process. At each stage we continue the themes introduced in Chapter 3 by showing not only how to simulate these models, but how to simulate them perfectly. In the case of the area-interaction process, this requires an extension of coupling from the past.

After introducing these standard models, we then move to discuss methods for analysing spatial point patterns, before introducing a new spatial point process model. We show how to simulate this model perfectly using coupling from the past. Finally, we use the methods of analysis introduced earlier to check what kind of behaviour we can expect from spatial point processes of this form.

4.1 Spatial Point Processes

In this section we introduce the simplest type of spatial point process — the homogeneous Poisson process — before moving on to discuss the area-interaction process (which contains the Poisson process as a special case) in Section 4.1.2.

Firstly, however, we must mention some notational conventions which we have used below.

We follow the lead of Cressie (1993) by making the following distinction between points and events: let X be a point configuration in some locally compact metric space χ. Then we call an element $x \in X$ an *event*, and an element $\xi \in \chi$ a *point*. Using this definition, it might be more correct to begin to call spatial point processes "spatial event processes" and point configurations "event configurations", but we will retain the former names, as there does not seem to be any possibility of confusion here. We will sometimes refer to the element of χ where there is an event in X as the *location* of that event.

4.1.1 The Poisson Process

Let χ be some locally compact complete metric space and $\mathfrak{N}^f$ be the space of all possible configurations of points in χ. Let μ be a finite Borel regular measure on χ. Then N is a *homogeneous Poisson process with intensity* λ if

- The number of events $N(A)$ in any bounded region A of χ has a Poisson distribution with parameter $\lambda\mu(A)$.
- Given that there are n events in A, their locations are independent and form a random sample from a uniform distribution on A.

This definition immediately gives a method for simulating a Poisson process over a rectangle $R \subset \mathbb{R}^2$. If the length of R is given by L and the width by W then we first draw from a Poisson(LW) distribution to find the number of events in R. We then scatter the events uniformly within R by drawing two $U[0,1]$ numbers x and y for each event and placing the event at location (Lx, Wy), where the first coordinate signifies the distance along the side of length L and the second signifies the distance along the side of length W. In Section 4.1.3 we discuss some further methods of simulating the homogeneous Poisson process and then use those methods together with CFTP to enable us to simulate the area-interaction process (Baddeley and van Lieshout 1995).

In this thesis we often talk about *complete spatial randomness*. When we do so, we are talking about spatial point patterns whose behaviour is the same as that of the homogeneous Poisson process. A general or *inhomogeneous* Poisson process is one for which λ is not a constant, but varies over χ.

4.1.2 The Area-Interaction Point Process

The area-interaction point process was first proposed in 1995 by Baddeley and van Lieshout as a model which was suitable for producing "both moderately clustered and moderately ordered patterns". It is interesting precisely because it *does* produce clustered patterns, something which the more traditional Strauss process (see Strauss (1975)) fails to do, as was pointed out by Kelly and Ripley (1976). We consider the model in its most general setting, as we will need this generality when we come to discrete models in Section 5.1. We then go on to give the most typical example of this model.

Let χ be some locally compact complete metric space and $\mathfrak{N}^f$ be the space of all possible configurations of points in χ. Let ν be a finite Borel regular measure on χ and $Z : \chi \to \mathcal{K}$ be a myopically continuous function (where $\mathcal{K}$ is as usual the class of all compact subsets of χ). Then the probability density of the general area-interaction process is

$$p(X) = \alpha\lambda^{N(X)}\gamma^{-\nu(U(X))} \tag{4.1}$$

with respect to the unit rate Poisson process[1], where $X = \{x_1, \ldots, x_n\} \in \mathfrak{N}^f$, $U(X) = \bigcup_{i=1}^{n} Z(x_i)$ and $N(X)$ is the number of points in the configuration X.

The crucial thing to do now that we have made the definition is to show that it makes sense — in other words to show that the density (4.1) is measurable and integrable. Baddeley and van Lieshout prove this and for completeness we repeat

[1] The reader should note that spatial point process densities are almost always given with respect to the unit rate Poison process, and this statement refers to the fact that the expression in (4.1) is the Radon-Nikodým derivative with respect to the unit rate Poisson process rather than the standard Lebesgue measure since the Lebesgue measure is, for obvious reasons, not suited to measuring configurations! For a brief background on the Radon-Nikodým theorem see Sections 2.2 and 2.4.

their proof since it will help us when we come to extend the area-interaction process in Section 4.4.

Let $t > 0$ and consider $V = \{X \in \mathfrak{N}^f : \nu(U(X)) < t\}$. We will show that V is open in the weak topology with respect to the function U, and thus that $\nu(U(X))$ is weakly upper semicontinuous. Since we know that upper or lower semicontinuous functions are measurable it is then a short road to showing that (4.1) is measurable.

Pick $X \in V$. Then since ν is regular there is an open set $G \subset \chi$ containing $U(X)$ such that $\nu(G) < t$ as well. Now clearly X has no events in the set $H = \{x \in \chi : Z(x) \cap G^c \neq \phi\}$, and $Z(H) \subset J = \{K \in \mathcal{K} : K \cap G^c \neq \phi\}$, which is a closed set in the myopic topology. Clearly $Z^{-1}(J) = H$, and since Z is a myopically continuous function this shows that H is also closed[2]. It is now easy to see that $W = \{Y \in \mathfrak{N}^f : N(Y_H) = 0\}$ (where Y_H is the restriction of the configuration Y to the set H) is open in the weak topology with respect to the function U, and since V is the union of a collection of sets of the form W then V is also open.

This shows that $X \to \nu(U(X))$ is weakly upper semicontinuous. Thus the map $X \to \exp(-\nu(U(X)) \log \gamma)$ is weakly upper semicontinuous if $\gamma \in (0, 1)$ and weakly lower semicontinuous if $\gamma > 1$. Thus $X \to \exp(-\nu(U(X)) \log \gamma)$ is measurable. Since $\lambda^{N(X)}$ is clearly measurable this means that (4.1) is measurable.

To see that (4.1) is integrable note that

$$0 \leq \nu(U(X)) \leq \nu(\chi) < \infty. \tag{4.2}$$

Now the function $f(X) = \lambda^{N(X)}$ is integrable, since this is simply the Radon-Nikodým derivative of the Poisson process with rate λ with respect to the unit rate Poisson process. Hence (4.1) is dominated by an integrable function, and therefore integrable[3]. □

[2] It can be shown that the inverse image of a closed set with respect to a continuous function is closed.

[3] In fact this shows that the generalised area-interaction process measure is uniformly absolutely continuous with respect to the λ-rate Poisson process measure and so its Radon-Nikodým derivative is uniformly bounded.

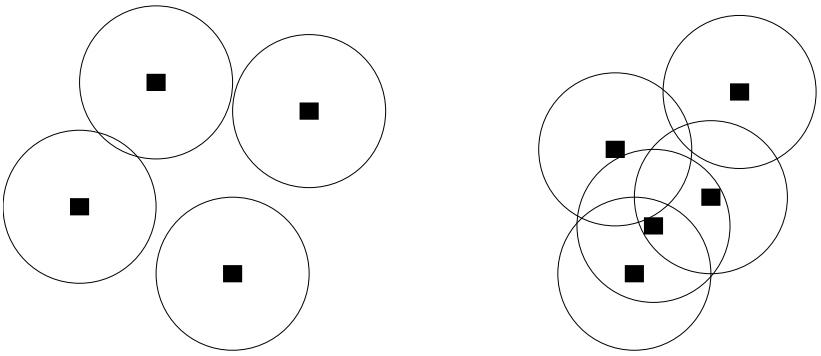

Figure 4.1: An example of some events together with circular "grains" G. The events in the above diagram would be the actual members of the process. The circles around them are to show what the set $X \oplus G$ would look like. If γ were large, the point configuration on the right would be favoured, whereas if γ were small, the configuration on the the left would be favoured.

As a specific example we consider the standard two dimensional case where the measure is Lebesgue and the myopically continuous function is Minkowski addition:

$$p(X) = \alpha\lambda^{N(X)}\gamma^{-m(X\oplus G)}, \tag{4.3}$$

where G is some compact subset of χ. Here $0 < \gamma < 1$ is the *repulsive* case, while $\gamma > 1$ is the *attractive* case. The case $\gamma = 1$ gives total spatial randomness.

As a further simplification consider the case where G is a disk of radius r. Then the set $X \oplus G$ is the set of all points within a distance r of an event in X. See Figure 4.1 for illustration. It is easy to see that in this simple case the area-interaction process is Markov of range $2r$.

We see that (4.3) reduces to the Poisson process with rate λ when $\gamma = 1$. The density (4.3) also looks superficially similar to the Strauss process, which has density

$$p(X) = \alpha\lambda^{N(X)}\gamma^{s(X)},$$

where $s(X)$ is the number of pairs of events within a distance r of one another, and also reduces to the Poisson process with rate λ when $\gamma = 1$. However, the area-interaction process is well-defined for $\gamma > 1$, while the Strauss process is non-integrable for these values of γ (Kelly and Ripley 1976). Thus the area-interaction

process can produce clustered point patterns, whereas the Strauss process cannot.

4.1.3 Simulation of the Poisson Processes

In this section we discuss two further methods for simulating the Poisson process. For a more general background in simulation see Ross (1990). The material here is important background to Section 4.1.4, where the simulation of the area interaction process is discussed. For simplicity we will only discuss simulation on a rectangular subset of $\mathbb{R}^2$ although the methods can be extended fairly naturally.

The first method is an almost trivial modification of the method discussed at the beginning of this section. Say we wish to simulate a Poisson process with rate λ on a rectangular box R having length L and width W. Then first we simply generate Exponential(λ) random variables[4] X_i until n is the smallest integer such that

$$S_n = \sum_{i=1}^{n} X_i > L \times W.$$

Second, we generate $(n-1)$ $U[0,1]$ random variables, U_i. Finally, the coordinates of the locations of the $(n-1)$ events in the realisation of the Poisson process with rate λ over R which we have just generated are $(S_i/W, WU_i)$, where the origin of this coordinate system is a corner of R and the first coordinate is the distance along the side of length L, the second along the side of length W.

To see why this method is equivalent to the method discussed earlier we need only consider the first coordinate, as the second coordinate is identical in both cases. In considering the inter-event distances we must consider the probability that there are no events in the interval $[a, b]$ for arbitrary a and b such that $a < b$. Now since the events are uniformly distributed it is clear that

$$P(p \in [a, b]) = \frac{b-a}{L}$$

for each event p in the configuration. Thus if we label the events $p_1, \ldots, p_k$ we see

[4] See Norris (1997) Section 2.4 for details of generating exponential random numbers.

that

$$P(p_i \neq [a,b]\ \forall i \in 1,\ldots,k | k = n) = \left(1 - \frac{b-a}{L}\right)^n.$$

Since $k \sim \text{Poisson}(\lambda)$ we have

$$\begin{aligned} P(p_i \neq [a,b]\ \forall i \in 1,\ldots,n) &= \sum_{n=0}^{\infty} \frac{e^{-\lambda}\lambda^n}{n!}\left(1 - \frac{b-a}{L}\right)^n \\ &= e^{-\lambda}\sum_{n=0}^{\infty} \frac{\left(\lambda\left(1 - \frac{b-a}{L}\right)\right)^n}{n!} \\ &= e^{-\lambda} \times \exp\left\{\lambda\left(1 - \frac{b-a}{L}\right)\right\} \\ &= e^{-\lambda(b-a)/L}. \end{aligned}$$

Thus inter-event distances are exponentially distributed with parameter λ/L just as in the method outlined above.

We now come to the second and more general method of simulating a Poisson process. This is a Markov process method utilising a *birth-death process*.

Define the *transition rate* of a Markov process X from state r to state s with $r \neq s$ to be

$$q(r,s) = \lim_{h \to 0} \frac{P(X(t+h) = s | X(t) = r)}{h}.$$

Let χ be some locally compact complete metric space (e.g. $(\mathbb{Z}, d)$ with $d(x,y) = |y - x|$) and $\mathfrak{R}^f$ be the space of all possible configurations of points in χ. Let $r, s \in \mathfrak{R}^f$ and let $N_r(p)$ be the number of events at $p \in \chi$ under configuration r. Then a Markov process Z is called a birth-death process if $q(r,s) = 0$ unless

- r and s differ at only a single location $p \in \chi$ and
- $|N_s(p) - N_r(p)| = 1$.

If these conditions hold and $N_s(p) - N_r(p) = 1$ then $q(r,s)$ is called the *birth rate* of Z at p and $q(s,r)$ is called the *death rate* of Z at p. In the case of the Poisson process these will be independent of the location p. The standard procedure is to have the death rate as unity and to have a more complicated rate as the birth rate. Let r and s be identical configurations except for one event, x, which is in s

and not in r. When the death rate is unity the detailed balance condition[5]

$$\pi(r)q(r,s) = \pi(s)q(s,r) \tag{4.4}$$

allows us to find the birth rate because equation (4.4) becomes simply

$$q(r,s) = \frac{\pi(s)}{\pi(r)}, \tag{4.5}$$

where $q(r,s)$ is the rate at which the birth-death process moves from state r to state s and is thus the birth rate. Noting that we may write $s = r \cup \{p\}$ we see that (4.5) becomes

$$q(r,s) = \frac{\pi(r \cup \{p\})}{\pi(r)},$$

and see that the birth rate is simply the Papangelou conditional intensity of a point process introduced in Section 2.3[6].

The density of a Poisson process of rate λ (with respect to a unit rate Poisson process) is

$$p(X) = \alpha\lambda^{N(X)},$$

where $N(X)$ is the number of events in the configuration X and α is a normalising constant. Thus the birth rate for a Poisson process with rate λ is simply

$$q(r,s) = \frac{\alpha\lambda^{N(s)}}{\alpha\lambda^{N(r)}} = \lambda,$$

since s has one more event in it than r. Clearly the Poisson process is one example of a birth-death process whose birth rate is independent of location.

This gives a very simple way to evolve a Poisson process through time. Starting from some initial configuration, generate an Exponential(1) random number for each event in the configuration. These are the death times of the events. Now generate a collection of Exponential(λ) random numbers and for each of these, an

[5] The detailed balance equation is important because whenever it holds for a Markov chain (or process) then that Markov chain (or process) is *reversible*, and reversibility is a sufficient condition for stationarity. See Kelly (1979) chapter 1 for further details.

[6] It is easy to see that if we define the birth rate to be unity the death rate must be the inverse of the Papangelou conditional intensity.

Exponential(1) random number so that we have a collection of pairs of random numbers (x_i, y_i). The cumulative partial sums of the x_i's are the birth times $b_i = \sum_{j=1}^{i} x_i$ of new events whose death times are given by $b_i + y_i$. When a death occurs we remove that event from the configuration. When a birth occurs a new event is put into the configuration at a random (Uniform) position.

Due to the detailed balance condition this process will tend asymptotically to a Poisson process of rate λ. Thus if the initial configuration is obtained from the one of the first two methods of simulating a Poisson process then the Markov process will *always* give a Poisson configuration. This fact will be used in the following section.

4.1.4 Perfect Simulation of the Area-Interaction Process

In this section, we review the method of Kendall (1998) for the perfect simulation of the area-interaction process. For simplicity, we first restrict attention to the attractive case of the area-interaction process (i.e. the case where $\gamma > 1$). Throughout this section we assume that we are generating an area-interaction process on the unit square in $\mathbb{R}^2$ and are using the standard two dimensional Lebesgue measure, although all of the results generalise to any kind of bounded compact window and most reasonable measures (to be precise, it must be a *totally finite, Borel-regular* measure).

As was mentioned in Section 3.4, the reason dominated CFTP got its name is that the perfect simulation of this process depends upon the *stochastic domination* of the process by the underlying Poisson process with rate λ, and depends upon the fact that *we can sample directly from the stationary distribution of this Poisson process*. We showed how this is done in Section 4.1.3.

The procedure is as follows: We obtain a sample of the Poisson process with rate λ and call this the state at time zero. We then evolve the process *backwards* until some fixed time $-T$, using a birth and death process with death rate equal to 1 using the method described at the end of the Section 4.1.3.

After this has been done, we next give all the events that exist in the sample in the interval $[-T, 0]$ marks generated from a $U[0, 1]$ distribution. These marks will be used for rejection sampling, and so are denoted $P(x)$, where x is the event, since they will be used for determining the outcome of events which occur with a certain probability. So far, except for the strange insistence upon generating the process backwards, this all looks like the standard process of simulating a spatial point process using birth-death models. We now apply monotone CFTP to make the simulation exact.

Recursively define two new processes, Y^{max} and Y^{min} as follows. The initial configurations at time $-T$ for the processes are

$$\begin{aligned} Y^{max}(-T, -T) &= \{x \ : x \in Z(-T)\} \\ Y^{min}(-T, -T) &= \left\{x \ : x \in Z(-T) \text{ and } P(x) \leq \gamma^{-m(G)}\right\} \end{aligned}$$

where $Z(-T)$ is the underlying Poisson process which we just simulated at time $-T$, $P(x)$ is the $U[0, 1]$ mark given to the event x and G is the compact set introduced in equation (4.3).

The processes are then generated *forwards* in time to time $t = 0$ in the following way:

At each time u in $[-T, 0]$ assume that the processes have been generated up to that time, and suppose that the next birth or death to occur happens at time t_i. If a **birth** happens next then we accept the birth for our Y processes if

$$P(x) \leq \gamma^{-m((x \oplus G) \setminus Y(-T,u) \oplus G)}, \tag{4.6}$$

where x is the event to be born and Y is either Y^{max} or Y^{min} depending upon which process we are generating. Figure 4.2 gives two examples of the construction $(x \oplus G) \setminus Y(-T, u) \oplus G$.

If, however, a **death** happens next then if the event is present in either of our processes we remove the dying event, setting

$$Y(-T, t_i) = Y(-T, u) \setminus \{x\}.$$

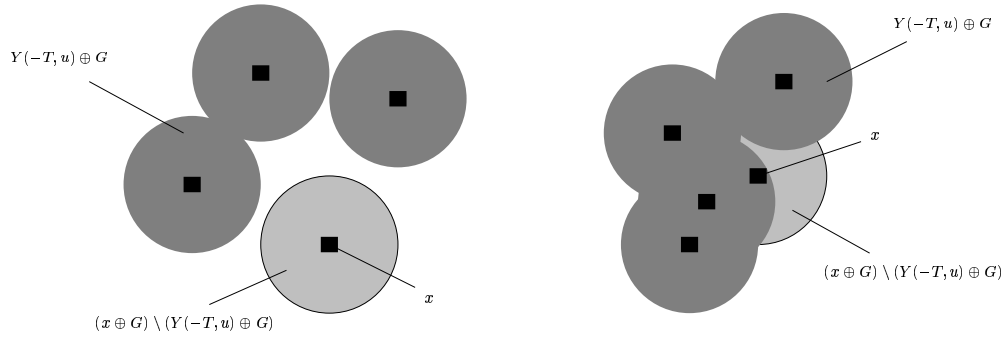

Figure 4.2: Another look at Figure 4.1 with some shading added to show the process of simulation. Dark shading shows $Y(-T,u)\oplus G$ where $Y(-T,u)$ is the state of the process immediately before we add the new event. Light shading shows the amount added if we accept the new event. In the configuration on the left, $x\oplus G=(x\oplus G)\setminus(Y(-T,u)\oplus G)$, so that if we are simulating an attractive process we are very unlikely to accept the new event. In the configuration on the right we are adding very little area to $(Y(-T,u)\oplus G)$ by adding the event, so we are far more likely to accept this event in an attractive process.

All that now remains is to define $Y(-T,u+\varepsilon)=Y(-T,u)$ for $u<u+\varepsilon<t_i$.

From equation (4.6) and the fact that we are accepting/rejecting events from a process with birth rate λ, we see that events are born at rate

$$\lambda\times\gamma^{-m((x\oplus G)\setminus Y(-T,u)\oplus G)},$$

which is exactly the Papangelou conditional intensity of the area-interaction process. It is clear that events die at a constant rate of 1. Thus detailed balance shows us that both Y^{max} and Y^{min} converge in distribution to an area interaction process.

If these two processes are identical at time zero (i.e. if $Y^{max}(-T,0)=Y^{min}(-T,0)$), then we have the required sample from the area-interaction process with rate parameter λ and attraction parameter γ. If not, we extend the underlying Poisson process back in time to time $-(T+S)$, generate additional $U[0,1]$ marks (keeping the ones already generated), and start again.

To see why the argument works imagine that the underlying Poisson process Z has been going since time $-\infty$. This 'thought experiment' works because we have generated Z in stationarity. The area-interaction process generated using

the same random events as Z will clearly have fewer members than Z, but those members it has *will be members of* Z. Thus our Y^{max} works as a substitute 'maximum' process (since there is no 'True' maximum process) and Y^{min} is a suitable 'minimum' process since $\gamma^{-m(G)}$ is a lower bound on the proportion of the events in Z which will be in the area-interaction process (since this is based on the quantity $m(X \oplus G)$ when there is only *one event* in X).

The repulsive case is handled using a few simple modifications. If $\gamma < 1$, then clearly $\lambda\gamma^{-m(G)} > \lambda$, so that λ is no longer an upper bound on the birth rate, but is actually a *lower* bound. This is easy to fix, however, as $\lambda\gamma^{-m(G)}$ *is* an upper bound. Thus for the repulsive case we use a Poisson process with rate $\lambda\gamma^{-m(G)}$ as our dominating process and change the definition of $Y^{min}(-T,-T)$ and the acceptance/rejection step. Change $Y^{min}(-T,-T)$ to

$$Y^{min}(-T,-T) = \left\{x \ : x \in Z(-T) \text{ and } P(x) \leq \gamma^{m(G)}\right\},$$

since $\lambda\gamma^{-m(G)} \times \gamma^{m(G)} = \lambda$. The acceptance/rejection step is a little more tricky, as the obvious modification of accepting an event when

$$P(x) \leq \gamma^{m(G)-m((x\oplus G)\setminus Y(-T,u)\oplus G)}$$

breaks the stochastic monotonicity, as we then accept events in the minimum process with higher probability than in the maximum process (assuming that there are fewer events in the minimum process than the maximum process). However this can be fixed by accepting events in the maximum process according to the acceptance probability for the minimum process and accepting events in the minimum process according to the acceptance probability for the maximum process as follows:

- Accept events in Y^{max} if

$$P(x) \leq \gamma^{m(G)-m((x\oplus G)\setminus Y^{min}(-T,u)\oplus G)}.$$

- Accept events in Y^{min} if

$$P(x) \leq \gamma^{m(G)-m((x\oplus G)\setminus Y^{max}(-T,u)\oplus G)}.$$

An interesting consequence of this last modification is that the minimum and maximum processes are no longer Markov processes when treated individually, though the joint process (Y^{max}, Y^{min}) is Markov.

4.2 Descriptive statistics

The purpose of this section is to give a brief overview of some of the descriptive statistics used to characterise spatial point processes. We present only the definition and the most naïve estimators. A more complete discussion is outside the scope of this thesis, and the reader is referred to Cressie (1993) for more in-depth coverage. In particular we do not discuss the often important area of edge correction, nor do we discuss any spaces other than $\mathbb{R}^n$.

4.2.1 Nearest neighbour measures

An important class of functions used in characterising spatial point processes are the nearest neighbour measures. These measures use point-to-event and event-to-event distances, and range from simple minimum inter-event distances to more complex ratios of inter-event and point-to-event distances.

4.2.1.1 Minimum inter-event distances

This simple statistic was introduced by Ripley and Silverman (1978) as a quick method to determine whether further investigation of spatial interaction was necessary. For the sake of brevity we consider only $\mathbb{R}^2$. Silverman and Brown (1978) show that in the case where n events are distributed independently with density f,

$$\left\{ n(n-1)\pi \int f^2 \right\} d_i \xrightarrow{\mathcal{D}} \chi^2_{2i} \quad \text{as} \quad n \to \infty$$

where d_i is the ith-smallest inter-event distance. In the usual case of a uniform density over some region of area A,

$$\int f^2 = 1/A.$$

4.2.1.2 The empty space distribution F

For a stationary point process, the empty space function $F(r)$ is simply the probability that there is an event within r of a randomly chosen point $\xi \in \mathbb{R}^n$:

$$F(r) = P\big[n\{b(\xi, r)\} > 0\big].$$

In other words, if we were to drop a circle of radius r onto a point process, $F(r)$ tells us the probability that there will be at least one event in the circle.

To estimate this, we essentially drop a large number of these circles onto our configuration and count the number of times the circle contains at least one event. To do this it is usual to consider either a regular grid or a collection of Uniformly generated points over the sampling window as an auxiliary collection of ξs. If we let d_i be the distance from the ith auxiliary point (whether grid-based or random) to the nearest event in the configuration of interest we may estimate F by

$$\widehat{F}(r) = \frac{1}{n}\sum_{i=1}^{n} I(d_i < r), \qquad \text{for } r > 0,$$

where n is the number of auxiliary points. If we are not sampling on a torus or other surface without boundary it is clearly necessary to perform edge correction of some sort.

Empirical estimates of F are normally compared against complete spatial randomness, for which theoretical values of F are available. In particular, in $\mathbb{R}^2$ we have

$$F(r) = 1 - \exp(-\lambda\pi r^2).$$

See Cressie (1993) for details.

4.2.1.3 The nearest neighbour distribution G

The nearest neighbour function $G(r)$ of a point process X is the probability that there is an event within r of a randomly chosen *event* $\xi \in \mathbb{R}^n$:

$$G(r) = \mathbb{P}^!_\xi\big[n\{b(\xi, r)\} > 0\big],$$

where the reduced Palm distribution $\mathbb{P}^!_\xi$ is as defined in Section 2.3 and $\xi \in X$ is arbitrary.

For the purposes of estimation let n be the number of events observed and d_i be the distance from the ith event to its nearest neighbour, where we have imposed some arbitrary labelling on the events. Then we may estimate G by

$$\widehat{G}(r) = \frac{1}{n}\sum_{i=1}^{n} I(d_i < r), \qquad \text{for } r > 0.$$

As with $\widehat{F}(r)$ above, edge correction may be necessary if we are observing on a surface with a boundary.

As with F, empirical estimates of G are normally compared against complete spatial randomness, for which theoretical values of G are available. In particular, in $\mathbb{R}^2$ we have

$$G(r) = 1 - \exp(-\lambda\pi r^2),$$

which is the same as we had for F above. See Cressie (1993) for details. This fact is the motivation behind the next test function.

4.2.1.4 The J function

The J function (van Lieshout and Baddeley 1996) is based on F and G above and is defined as

$$J(r) = \frac{1 - G(r)}{1 - F(r)}, \qquad \text{for } r > 0 \text{ and } F(r) < 1$$

and can be estimated by

$$\widehat{J}(r) = \frac{1 - \widehat{G}(r)}{1 - \widehat{F}(r)}, \qquad \text{for } r > 0 \text{ and } \widehat{F}(r) < 1.$$

Under complete spatial randomness (i.e. for the homogeneous Poisson process) $J(r) = 1$ for all r, while $J < 1$ indicates clustering and $J > 1$ indicates regularity. The J function has several very nice properties, one of which we discuss in Section 4.3.3.

4.2.2 The K function

Perhaps the most frequently used measure of spatial structure is the K function, whose use goes back to Bartlett (1964), though its popularity is mainly due to the work of Ripley (1976, 1977). It measures the expected number of events within r of an arbitrary event as follows:

$$K(r) = \frac{|B|}{\mathbb{E}\{n(X)\}} \mathbb{E}^!_\xi X\{b(\xi, r)\},$$

where $\mathbb{E}^!_\xi$ is expected value with respect to the reduced Palm distribution as defined in Section 2.3 and we are observing a bounded region $B \in \mathcal{B}_0$. It is important to note that this definition is only valid for *stationary* point processes.

Despite the K function's popularity it is important to realise that it is insufficient to examine this statistic alone if we are to accurately characterise the behaviour of a point pattern. Doing so would be analogous to assuming that the mean and variance of a real valued random variable were sufficient to characterise the distribution from which it was drawn. To exemplify this point Baddeley and Silverman (1984) develop a model whose K function matches that of the Poisson process, but which is clearly not completely spatially random. Interestingly Kerscher (1998) shows that the J function (Section 4.2.1.4) successfully distinguishes between these two models and Schladitz and Baddeley (2000) show that the T function (Section 4.2.3) is also capable of discriminating between the Poisson process and Baddeley and Silverman's model. Somewhat curiously, the J function seems to emphasise the regularity of Baddeley and Silverman's model, whereas the T function detects clustering.

The K function can be estimated by

$$\widehat{K}(r) = \frac{|B|}{n} \sum_{i \neq j} \frac{I\{d(e_i, e_j) < r\}}{n - 1}$$

where n is the number of observed events and $d(e_i, e_j)$ is the distance between the ith and jth events (where we have imposed an arbitrary labelling on the events).

Ripley (1976) gives an edge corrected estimator for the case where we are sampling on a region with a boundary.

As with F, G and J above, empirical estimates for K are normally compared against complete spatial randomness, for which theoretical values of K are available (see Ripley (1977)), and in fact come out rather nicely. In particular, in $\mathbb{R}^2$ we have

$$K(r) = \pi r^2.$$

Due to this result, Besag (1977b) suggests that we consider a variation, L, of the form

$$L(r) = \sqrt{K(r)/\pi},$$

since $L(r) = r$ for the homogeneous Poisson process. In fact, as Cressie (1993) suggests, it is even easier to see what is going on if we plot $L(r) - r$.

4.2.3 The T function

The T function (Schladitz and Baddeley 2000) can be considered as a natural extension of the K function introduced in the previous section. Whereas the K function considers the expected number of events within r of an event, the T function considers the number of *pairs* of events which are

- within r of a specific event, and
- within r of *each other*.

Formally, the T function is defined as

$$T(r) = \frac{|B|^2}{2\big[\mathbb{E}\{n(X)\}\big]^2}\mathbb{E}^!_\xi \sum_{x,y\in b(\xi,r)} I\big\{0 < d(x,y) \le r\big\}, \qquad \text{for } r \ge 0,$$

where we again observe a bounded region $B \in \mathcal{B}_0$ and $\mathbb{E}^!_\xi$ is again with respect to the reduced Palm distribution of the process.

Schladitz and Baddeley (2000) give various methods for estimating T. We give a version without edge correction:

$$\widehat{T}(r) = \frac{|B|^2}{2n(n-1)(n-2)} \sum_{x\in X} \sum_{y,z\in X} \mathrm{pa}_r(x,y)\mathrm{pa}_r(x,z)\mathrm{pa}_r(y,z),$$

where $\mathrm{pa}_r(x, y)$ is the indicator function that events x and y constitute an r-close pair, i.e.

$$\mathrm{pa}_r(x, y) = I\big\{0 < d(x, y) \leq r\big\}.$$

The reader is referred to Schladitz and Baddeley (2000) for a discussion of edge-corrected estimators for T.

The crucial property of the T function is that it is a measure of third order interaction in the process. This means that it can distinguish some point processes whose first and second moments are similar, whereas if we were only to use the K function and measure the intensity we would not be able to do so. In particular, as mentioned in Section 4.2.2, it is able to distinguish between the Poisson process and the cell process of Baddeley and Silverman (1984).

Under complete spatial randomness it is possible to calculate theoretical values for the T function. In particular in $\mathbb{R}^2$ we have

$$T(r) = \frac{1}{2}\pi(\pi - \frac{3}{4}\sqrt{3})r^4.$$

Similarly to Besag's suggestion for plotting the K function, Schladitz and Baddeley (2000) suggest plotting

$$\sqrt[4]{\frac{2\widehat{T}(r)}{\pi(\pi - \frac{3}{4}\sqrt{3})}} \; - \; r$$

rather than $\widehat{T}(r)$ directly.

4.3 Parameter estimation techniques

There is not much point having a model without any way to estimate the parameters. We briefly introduce various approaches to this problem in the following sections.

4.3.1 Maximum likelihood

For many spatial point process models, we only know the density to within a constant of proportionality, i.e.

$$p(x) = cg(x)$$

where the normalising constant c is often not analytically known. As a result, standard maximum likelihood estimation relies on Monte-Carlo estimates of the normalising constant of the likelihood. We do not discuss the method further, but refer the reader to Geyer and Thompson (1992) for further information.

4.3.2 Maximum Pseudo-likelihood

The method of maximum pseudo-likelihood goes back to Besag (1974, 1975, 1977a), though we base our treatment on the approach used by Jensen and Møller (1991). The *pseudo-likelihood* of a point process X with Papangelou conditional intensity $\lambda(u; X)$ over a subset $A \subset W$ is

$$\mathrm{PL}_A(\theta, X) = \left[\prod_{x_i \in A} \lambda(x_i; X)\right] \exp\left\{-\int_A \lambda(u; X) du\right\},$$

where we observe events x_i in a bounded window W. This is similar to the likelihood, which for independent events is the product of the density of those events. Here instead we use a product of conditional intensities at the events given the remainder of the configuration, since the likelihood itself is usually intractable.

As usual with likelihood equations we take logs, differentiate and set equal to zero. Under regularity conditions (so that we can interchange the order of integration and differentiation) this gives equations

$$\sum_{x_i \in A} \frac{\partial}{\partial \theta} \log \lambda(x_i; X) = \int_A \frac{\partial}{\partial \theta} \lambda(u; X) du,$$

for each parameter θ in the model. Jensen and Møller (1991) prove that for exponential family models $\mathrm{PL}_A(\theta, X)$ is concave and for Markov models of finite range it is also consistent.

Baddeley and Turner (2000) develop useful methods for estimating the pseudo-likelihood using existing statistical packages. A worked example is presented in Section 4.4.3.

4.3.3 Takacs-Fiksel estimation

The Takacs-Fiksel estimation method uses the identity

$$\mu \mathbb{E}_a^! f(X) = \mathbb{E}[\lambda(a; X) f(X)], \tag{4.7}$$

which holds for any bounded measurable non-negative function $f : \mathcal{N} \to \mathbb{R}^+$ and any stationary point process X on $\mathbb{R}^d$ with finite intensity μ. As usual, $\mathbb{E}_a^!$ is with respect to the reduced Palm distribution and $\lambda(a; X)$ is the Papangelou conditional intensity of X at a (see Section 2.3).

Estimation requires choosing a number of functions greater than or equal to the number of parameters in the model.

For the area-interaction process described in Section 4.1.2, Baddeley and van Lieshout (1995) invite us to consider the case where $f(X)$ in (4.7) is

$$f(X) = \frac{\mathbf{1}_{\{X \cap B(a,s) = \phi\}}}{\lambda(a; X)}.$$

This is the indicator function that there is not an event in X within s of a divided by the conditional intensity of X at a. In the case of the area-interaction process this becomes

$$f(X) = \frac{\mathbf{1}_{\{X \cap B(a,s) = \phi\}}}{\lambda \gamma^{-m((a \oplus G) \setminus (X \oplus G))}}.$$

An important special case of this is when $s > 2d$ where d is the maximum distance between a point in $a \oplus G$ and a ($d = r$ in the special case where G is a disk of radius r centred at a). For this special case we have

$$f(X) = \frac{\mathbf{1}_{\{X \cap B(a,s) = \phi\}}}{\lambda \gamma^{-m(G)}} = \mathbf{1}_{\{X \cap B(a,s) = \phi\}} \frac{\gamma^{m(G)}}{\lambda}. \tag{4.8}$$

We can now substitute this in to both sides of equation (4.7). The left hand

side is

$$\begin{aligned}\mu\mathbb{E}_a^! f(X) &= \mu\int \mathbf{1}_{\{X\cap B(a,s)=\phi\}}\frac{\gamma^{m(G)}}{\lambda}dP_a^!(X)\\ &= \mu\mathbb{P}_a^!\{X\cap B(a,s)=\phi\}\frac{\gamma^{m(G)}}{\lambda}\end{aligned}$$

and the right hand side is simply

$$\mathbb{E}[\lambda(a;X)f(X)] = \mathbb{P}\{X\cap B(a,s)=\phi\},$$

which follows directly from equation (4.8).

We see that the equation becomes

$$\mu\frac{1-G(s)}{1-F(s)} = \lambda\gamma^{-m(G)}. \tag{4.9}$$

where $G(s)$ is the nearest neighbour distribution function of Section 4.2.1.3 and $F(s)$ is the empty space distribution function of Section 4.2.1.2. Thus equation (4.9) becomes simply

$$\mu J(s) = \lambda\gamma^{-m(G)},$$

where $J(s)$ is the J function introduced in Section 4.2.1.4 and the equation is valid for $s > 2d$, as stated above.

We do not discuss this method further except to say that under many models pseudo-likelihood estimation is a special case of the Takacs-Fiksel method where the functions f are taken to be

$$f(X) = \frac{\partial}{\partial\theta}\lambda(u,X).$$

See Baddeley and van Lieshout (1995) and the references contained therein for further details.

4.4 An extension of the area-interaction process

The area-interaction process is a flexible model which allows for a good range of models — from regular through total spatial randomness to clustered. Unfortunately it does not allow for models whose behaviour changes at different resolutions, for example repulsion at small distances and attraction at large distances.

Some real-world examples of places where we see this sort of behaviour are the distribution of trees on a hillside, or the distribution of zebra in a patch of savannah. Another example of large scale attraction and small scale repulsion is the interaction between the strong nuclear force and the electro-magnetic force between two oppositely charged particles. The physical laws governing this behaviour are different from those governing the behaviour of the area-interaction class of models, though they may be sufficiently similar so as to provide a useful approximation.

In an attempt to develop a model with these characteristics we propose a model with the following density:

$$p(X) = \alpha \lambda^{N(X)} \gamma_1^{-m(X \oplus G_1)} \gamma_2^{-m(X \oplus G_2)}, \tag{4.10}$$

where α, λ, $N(X)$, m and $X \oplus G$ are as in equation (4.3), $\gamma_1 \in [1, \infty)$, $\gamma_2 \in (0, 1]$ and G_1 and G_2 are two grains of different sizes. Usually we will have $G_1 \supset G_2$, resulting in small scale repulsion and large scale attraction. Clearly in the simple case where G_1 and G_2 are balls of radius r_1 and r_2 then this point process is Markov of range $\max\{r_1, r_2\}$. For the sake of compactness, from here on we will refer to this new point process as the *attractive-repulsive process*.

A trivial extension of the proof that the area-interaction process density is both measurable and integrable (see pages 52–53) shows that this process is also both measurable and integrable. We outline the extension below.

The proof on pages 52–53 shows that the map $X \to \gamma^{-\nu(U(X))}$ is weakly u.s.c. if $\gamma \in (0, 1)$ and weakly l.s.c. if $\gamma > 1$, where ν is any finite Borel regular measure over some locally compact complete metric space χ, and $U(X) = \bigcup_{i=1}^{n} Z(x_i)$, where $Z(x_i)$ is any myopically continuous function $Z : \chi \to \mathcal{K}$. Clearly this is sufficient to show that the mapping $X \to \gamma_1^{-m(X \oplus G_1)}$ is weakly l.s.c. and the mapping $X \to \gamma_2^{-m(X \oplus G_2)}$ is weakly u.s.c. Thus since $X \to \lambda^{N(X)}$ is clearly measurable so is (4.10).

Equation (4.2) on page 53 is sufficient to show that (4.10) is integrable.

4.4.1 Simulation

Perfect simulation of the above process is possible by an extension of the method for simulating the standard area-interaction process. We begin by noting that the Poisson process with rate

$$\lambda \times \gamma_2^{-m(G_2)}$$

dominates (4.10). Thus we begin by simulating a Poisson process with rate $\lambda \times \gamma_2^{-m(G_2)}$ and use this as our configuration at time 0. We then simulate the process backwards until some time $-T$ using a birth-death process with death rate equal to 1. These two steps are performed using the methods of Section 4.1.3. For consistency of notation, again let $Z(t)$ denote the configuration of events in this process at time t. Just as in the standard area-interaction process algorithm we then give all the events that exist in the model in the interval $[-T, 0]$ marks generated from a $U[0, 1]$ distribution. Next we recursively define two new processes Y^{max} and Y^{min} just as in Section 4.1.4:

We begin by defining the configurations at time $-T$:

$$\begin{aligned} Y^{max}(-T,-T) &= \{x \ : x \in Z(-T)\} \\ Y^{min}(-T,-T) &= \left\{x \ : x \in Z(-T) \text{ and } P(x) \leq \gamma_1^{-m(G_1)}\gamma_2^{m(G_2)}\right\} \end{aligned}$$

where $P(x)$ is the $U[0, 1]$ mark given to the event x. The two parameters of the Y's refer to the time at which the process starts and the time at which we are examining its value respectively. For example $Y^{max}(-T, u)$ is the value at time u of the Y^{max} process which was started at time $-T$ and simulated according to the rules given below.

The Papangelou conditional intensity of our process is

$$\lambda(u, X) = \lambda\gamma_1^{-m((u\oplus G_1)\setminus X\oplus G_1)}\gamma_2^{-m((u\oplus G_2)\setminus X\oplus G_2)},$$

so we see from equation (4.4) on page 57 that births must occur at rate

$$q(X, X \cup \{x\}) = \lambda\gamma_1^{-m((x\oplus G_1)\setminus X\oplus G_1)}\gamma_2^{-m((x\oplus G_2)\setminus X\oplus G_2)},$$

where X is the state of the process prior to a birth and x is a new event being born. Since we are simulating this process with respect to a Poisson process with rate $\lambda \times \gamma_2^{-m(G_2)}$ this means that as we simulate the maximum and minimum processes forwards in time we should accept births if

$$P(x) \leq \gamma_1^{-m((x \oplus G_1) \setminus X \oplus G_1)} \gamma_2^{m(G_2) - m((x \oplus G_2) \setminus X \oplus G_2)}.$$

However this breaks the monotonicity of the process, as it sometimes results in higher acceptance probabilities for the minimum process than it does for the maximum process. This is easy to rectify, as the acceptance probability is a product of monotonic functions. We find that if we accept events in the maximum process whenever

$$P(x) \leq \gamma_1^{-m((x \oplus G_1) \setminus Y^{max} \oplus G_1)} \gamma_2^{m(G_2) - m((x \oplus G_2) \setminus Y^{min} \oplus G_2)} \tag{4.11}$$

and accept events in the minimum process whenever

$$P(x) \leq \gamma_1^{-m((x \oplus G_1) \setminus Y^{min} \oplus G_1)} \gamma_2^{m(G_2) - m((x \oplus G_2) \setminus Y^{max} \oplus G_2)} \tag{4.12}$$

then monotonicity is restored. It is also easy to see that the stationary distribution of this new process is as required, although the maximum and minimum processes are no longer individually Markov processes, though the combined process

$$(Y^{max}, Y^{min})$$

is Markov. This is similar to the repulsive case for the standard area-interaction process covered in Section 4.1.4.

This differs from the approach taken by Kendall and Møller (1999) in a couple of ways. Firstly, the initial configuration of the minimum process used by Kendall and Møller was the empty configuration, whereas we used rejection sampling on the dominating processes, rejecting those events whose marks were greater than $\gamma_1^{-m(G_1)} \gamma_2^{m(G_2)}$. Secondly, our acceptance probabilities (Equations (4.11) and (4.12)) are different. The advantages of our method are:

- The minimum process is initially either closer to the maximum process than Kendall and Møller's minimum process, or at least as close.

- Our acceptance probabilities are much easier to compute[7].

The disadvantage of our method is that the difference between the acceptance probability for our maximum process and the acceptance probability for our minimum process is greater than (or equal to) the difference between the acceptance probabilities used by Kendall and Møller. This means that although the acceptance probabilities used in our method are faster to compute, the minimum and maximum processes will take more steps to coalesce. Our feeling is that the advantages outweigh the disadvantage, though more work is needed to show whether this is indeed true.

Having discussed the birth-behaviour of our process in some detail, we proceed by spelling out the algorithm as a whole.

The processes are generated forwards in time to time $t = 0$ in the following way:

At each time u in $[-T, 0]$ assume that the processes have been generated up to that time, and suppose that the next birth or death to occur happens at time t_i. If the next event is a **birth**, then we accept the birth for Y^{max} if

$$P(x) \leq \gamma_1^{-m((x\oplus G_1)\setminus Y^{max}(-T,u)\oplus G_1)}\gamma_2^{m(G_2)-m((x\oplus G_2)\setminus Y^{min}(-T,u)\oplus G_2)}$$

where x is the event to be born. We accept the birth for Y^{min} if

$$P(x) \leq \gamma_1^{-m((x\oplus G_1)\setminus Y^{min}(-T,u)\oplus G_1)}\gamma_2^{m(G_2)-m((x\oplus G_2)\setminus Y^{max}(-T,u)\oplus G_2)}.$$

If, however, the next event is a **death**, then we remove the dying event from our processes, setting

$$Y(-T, t_i) = Y(-T, u) \setminus \{x\}.$$

[7] The acceptance probabilities used by Kendall and Møller were

$$\max\{\lambda(u, Y)/K : Y^{min} \subseteq Y \subseteq Y^{max}\} \quad \text{and}$$
$$\min\{\lambda(u, Y)/K : Y^{min} \subseteq Y \subseteq Y^{max}\}$$

for the maximum and minimum processes respectively, where $\lambda(u, Y)$ is the Papangelou conditional intensity of the process of interest and K is the rate of the dominating process. This clearly involves calculation of $\lambda(u, Y)$ for each configuration that is both a subset of Y^{max} and a superset of Y^{min}. Since calculation of $\lambda(u, Y)$ is typically expensive, this calculation may be very expensive.

All that now remains is to define $Y(-T, u+\varepsilon) = Y(-T, u)$ for $u < u+\varepsilon < t_i$.

If these two processes are identical at time zero (i.e. if $Y^{max}(-T, 0) = Y^{min}(-T, 0)$), then we have the required sample from the attractive-repulsive process. If not we extend the underlying Poisson process back in time to time $-(T+S)$ for some $S > 0$, generate a few more $U[0, 1]$ marks (keeping the ones already generated), and start again generating the processes forwards to time $t = 0$ again.

The same reasoning which lead us to see that the algorithm in Section 4.1.4 correctly simulated the desired distribution also applies to this new process, since the true process will again be sandwiched in between the maximum and minimum processes.

4.4.2 Descriptive Statistics

To give examples of the type of behaviour it is possible to model with an attractive-repulsive process, we simulated 1000 draws from the process for different parameter values and calculated average values of $\widehat{F}$, $\widehat{G}$, $\widehat{J}$, $\widehat{K}$ and $\widehat{T}$ (see Sections 4.2.1.2, 4.2.1.3, 4.2.1.4, 4.2.2 and 4.2.3), as well as statistics for minimum inter-event distances (see Section 4.2.1.1).

In all cases discussed below we set $\lambda = 100$, simulate on the unit square and let the measure, m be Lebesgue scaled by a factor of 10π. This was done because it was discovered that in order to get point configurations which were significantly different from complete spatial randomness, it was necessary to have a very large value of γ_1 and a very small value of γ_2. Scaling by a factor of 10π handles this effectively, as it is equivalent to raising γ_1 and γ_2 to the power of 10π. We choose G_1 and G_2 to be circles with radii r_1 and r_2 respectively. All processes were generated with toroidal boundary conditions and the test functions were estimated similarly.

As a base line, we set $\gamma_1 = \gamma_2 = 1$, which generates a Poisson process (the values of r_1 and r_2 are irrelevant). We also simulated small scale attraction with large scale repulsion using parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$, and small scale repulsion with large scale attraction with parameters $\gamma_1 = 10$,

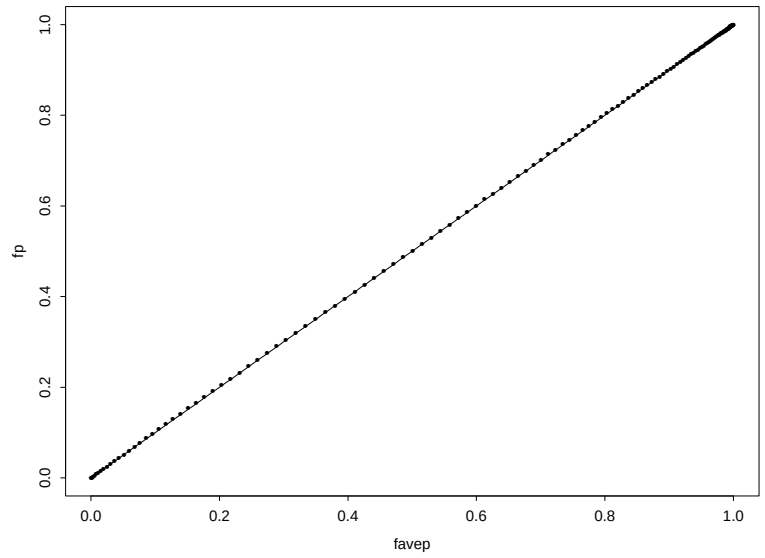

Figure 4.3: Probability plot of the $\widehat{F}$ function averaged over 1000 realisations of a Poisson process compared with the theoretical value for complete spatial randomness.

$\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$. Figures 4.3 to 4.13 show plots of $\widehat{F}$, $\widehat{G}$, $\widehat{J}$, $\widehat{K}$ and $\widehat{T}$ functions for these three processes[8], computed as an average over 1000 simulations. Simulation was performed using the perfect simulation techniques discussed in the previous section to produce independent draws. Discussion of this implementation is postponed until Chapter 7.

One traditional way to view $\widehat{F}$ and $\widehat{G}$ functions is to draw probability plots such as those in Figures 4.3 to 4.8, comparing empirical values against the theoretical values under complete spatial randomness. Inspired by Besag's suggestion for the $\widehat{K}$ function that we should transform the values based on the theoretical value under complete spatial randomness, in Figures 4.9 and 4.10 we plot

$$\widehat{E}(r) = \sqrt{-\frac{1}{\widehat{\lambda}\pi}\log(1-\widehat{F}(r))} \; - \; r \tag{4.13}$$

and

$$\widehat{H}(r) = \sqrt{-\frac{1}{\widehat{\lambda}\pi}\log(1-\widehat{G}(r))} \; - \; r. \tag{4.14}$$

[8] As suggested in Section 4.2.2, we plot $\widehat{L}(r) - r$ rather than $\widehat{K}$.

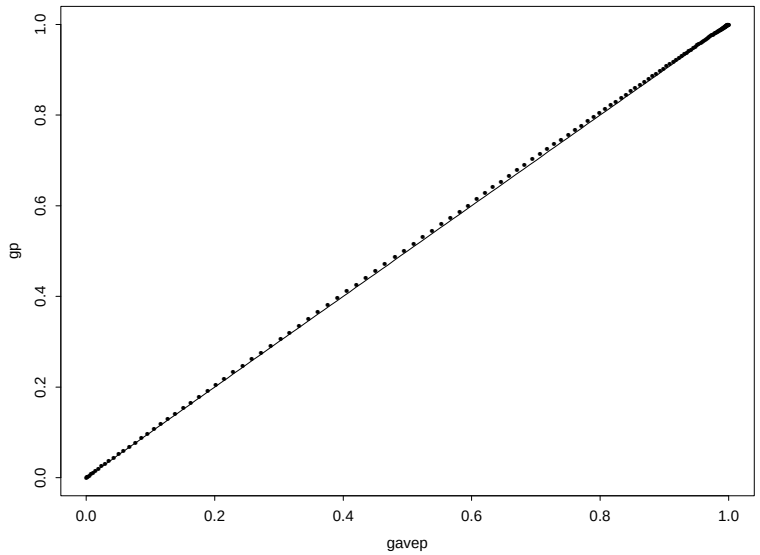

Figure 4.4: Probability plot of the $\widehat{G}$ function averaged over 1000 realisations of a Poisson process compared with the theoretical value for complete spatial randomness.

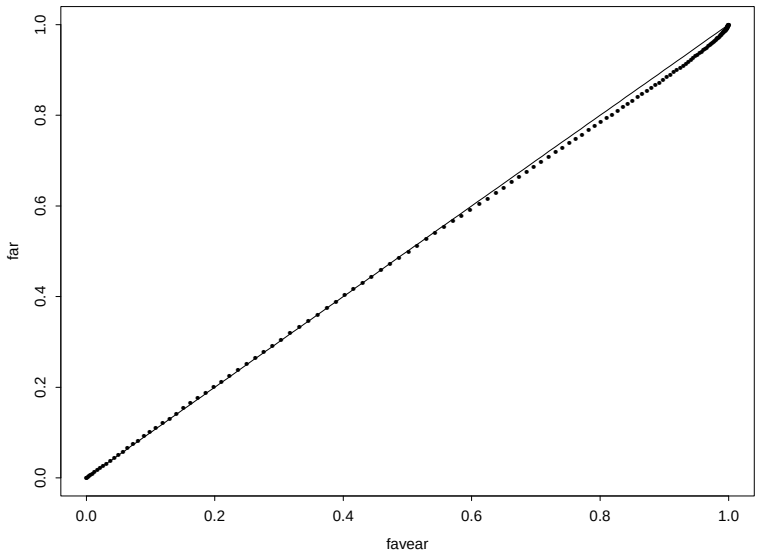

Figure 4.5: Probability plot of the $\widehat{F}$ function averaged over 1000 realisations of an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ compared with the theoretical value for complete spatial randomness.

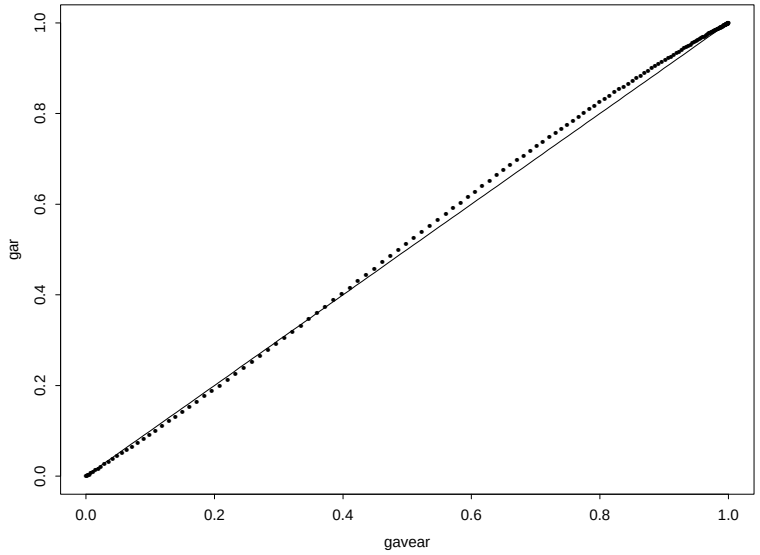

Figure 4.6: Probability plot of the $\widehat{G}$ function averaged over 1000 realisations of the attractive-repulsive process of Figure 4.5 compared with the theoretical value for complete spatial randomness.

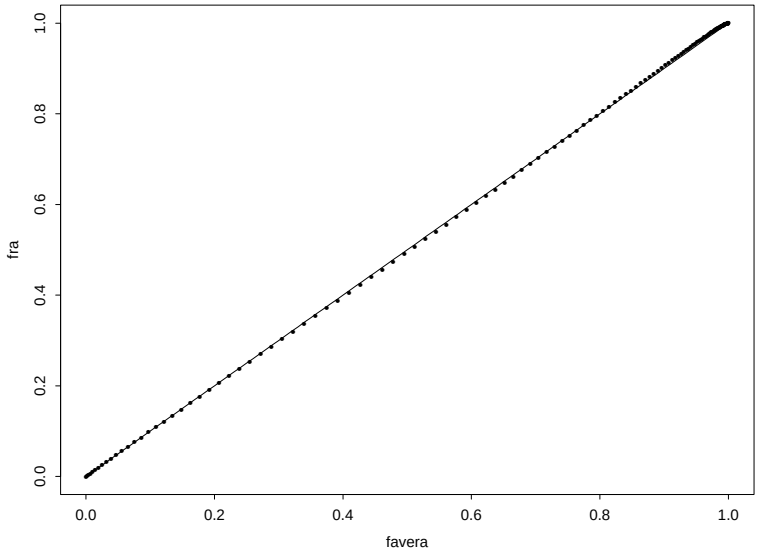

Figure 4.7: Probability plot of the $\widehat{F}$ function averaged over 1000 realisations of an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ compared with the theoretical value for complete spatial randomness.

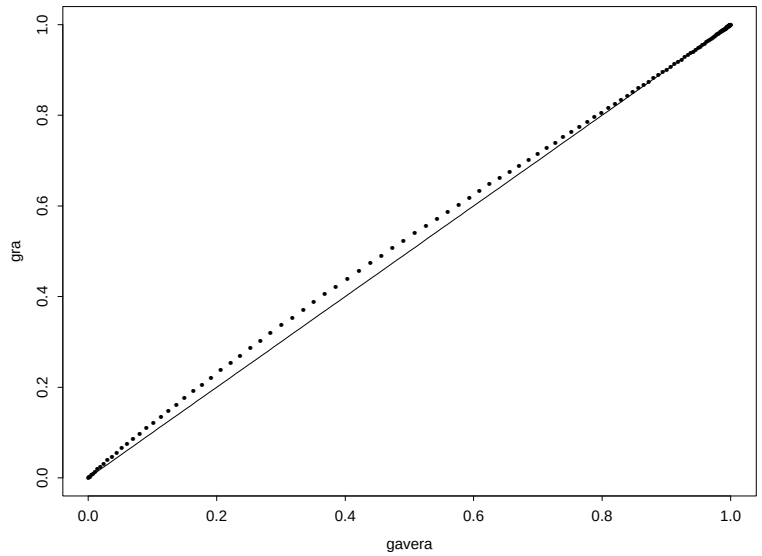

Figure 4.8: Probability plot of the $\widehat{G}$ function averaged over 1000 realisations of the attractive-repulsive process of Figure 4.7 compared with the theoretical value for complete spatial randomness.

Comparing these graphs with the probability plots in Figures 4.3 to 4.8 we see that it is far easier to see the behaviour of the different processes in the transformed plots. Another advantage is that we can compare $\widehat{F}$ or $\widehat{G}$ functions for several different processes on a single plot. Both $1 - \widehat{F}(t) = \varepsilon(t)$ and $1 - \widehat{G}(t) = \delta(t)$ are such that for $t \geq 0.2$ we have $\varepsilon(t) < \xi$ and $\delta < \xi$ where ξ is the precision of the software used to calculate $\widehat{F}(t)$ and $\widehat{G}(t)$. As a result, we have restricted attention to the range $[0, 0.18)$. Both $\widehat{F}$ and $\widehat{G}$ seem poor at picking up the short range interactions (though $\widehat{G}$ seems better than $\widehat{F}$), which is strange since they measure the distributions of nearest events.

Figure 4.11 shows a plot of the $\widehat{J}$ function for our three processes. A log scale seemed most natural for this plot, as values are in the range $(0, \infty)$, with the reference value for complete spatial randomness being one. Since $\widehat{F}$ and $\widehat{G}$ become one just below $r = 0.2$ we have again restricted attention to the range $[0, 0.18)$. Clearly $\widehat{J}$ is not very good at picking up the short range interactions, which is as

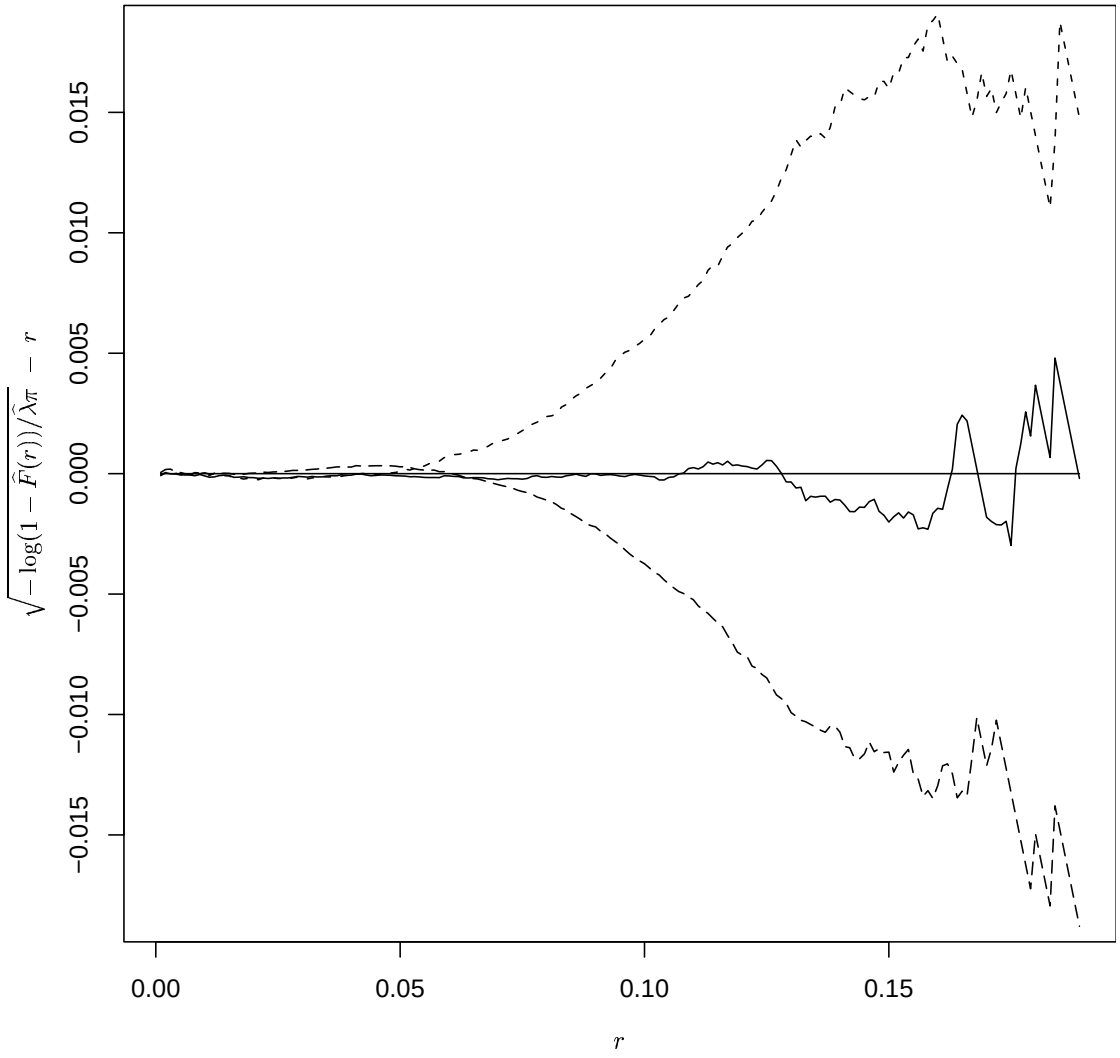

Figure 4.9: Plot of the transformed $\widehat{F}$ function given in equation (4.13) averaged over 1000 realisations of a Poisson process (solid line), an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ (short dashes) and an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ (long dashes).

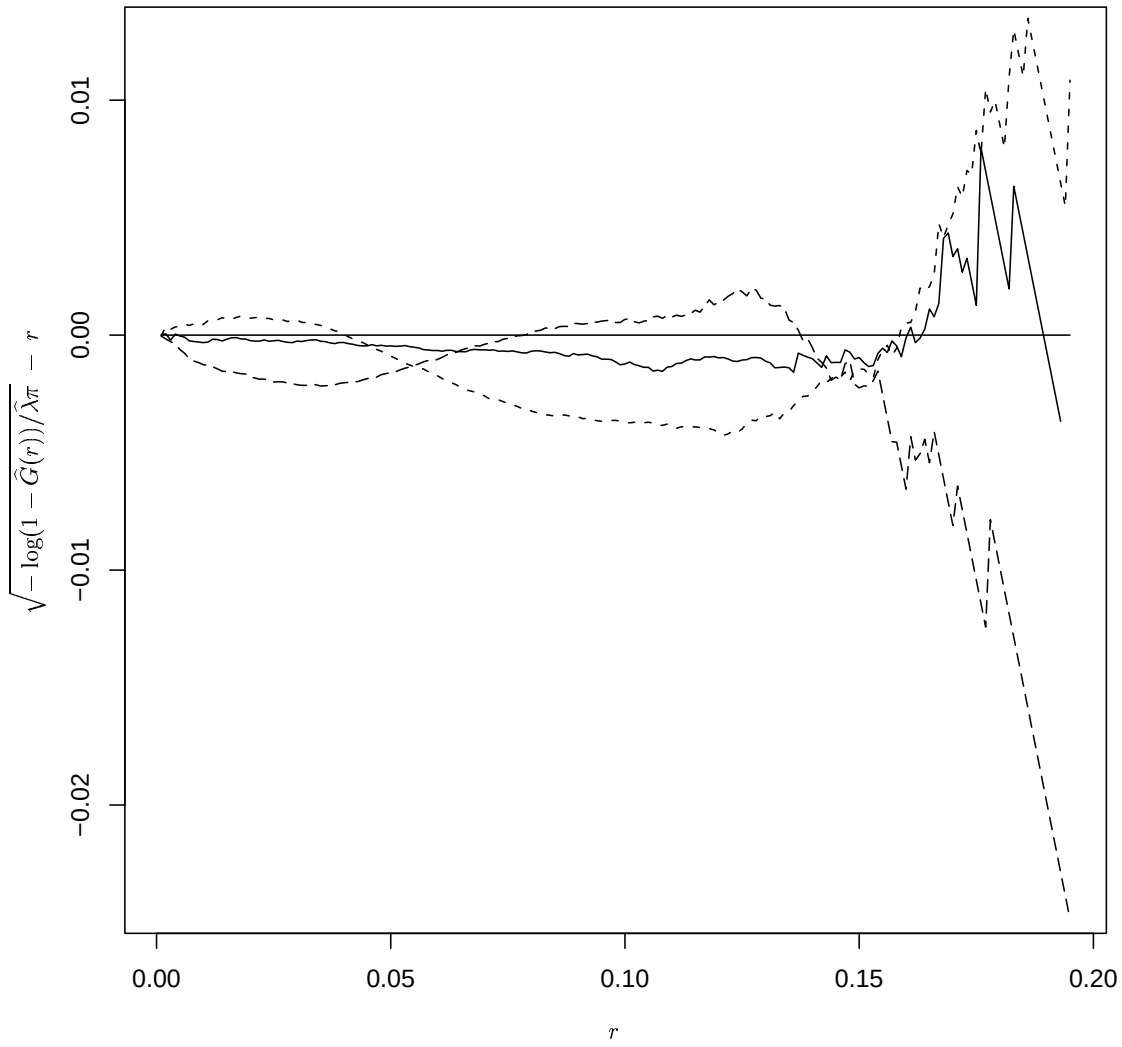

Figure 4.10: Plot of the transformed $\widehat{G}$ function given in equation (4.14) averaged over 1000 realisations of a Poisson process (solid line), an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ (short dashes) and an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ (long dashes).

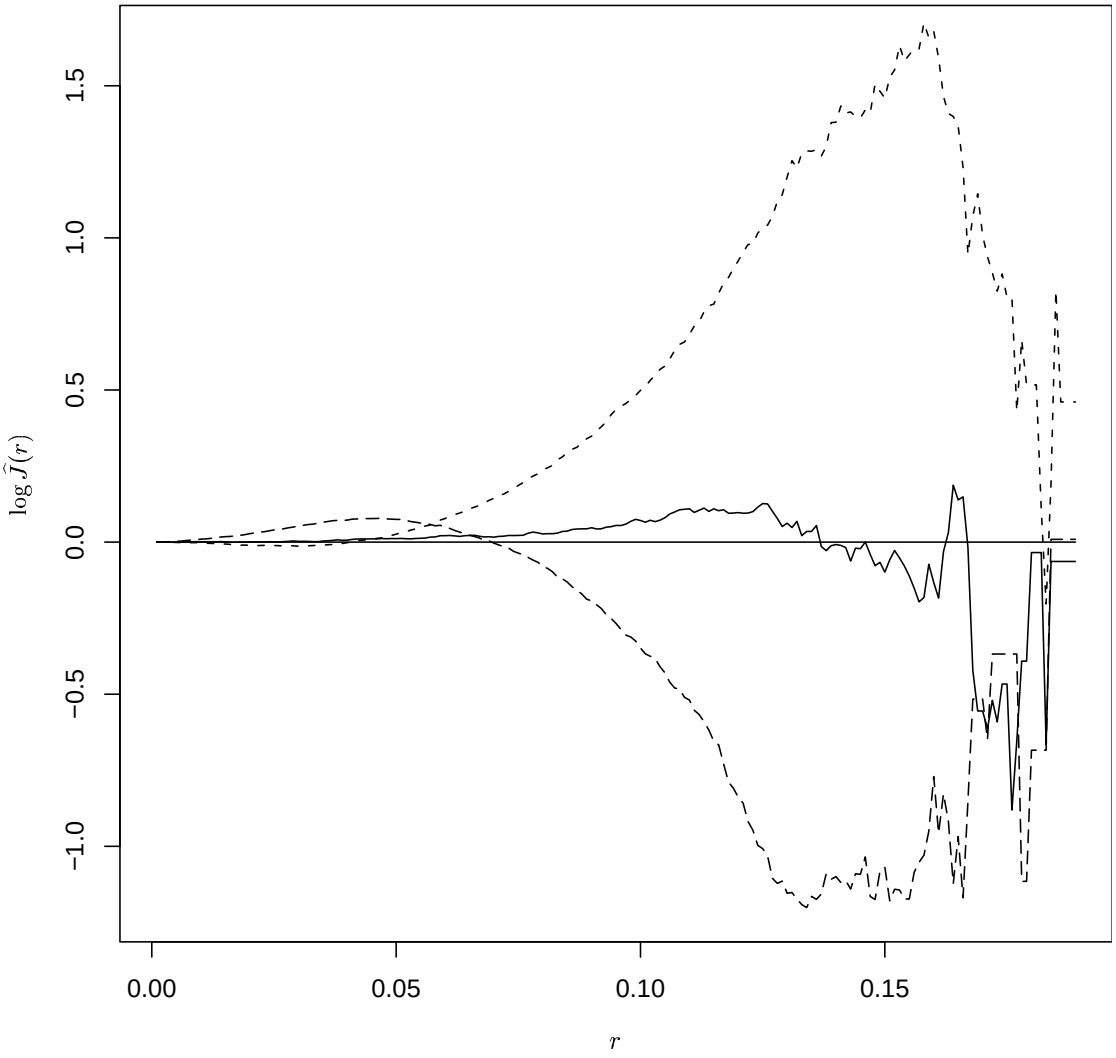

Figure 4.11: Plot of the natural logarithm of the $\widehat{J}$ function averaged over 1000 realisations of a Poisson process (solid line), an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ (short dashes) and an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ (long dashes).

we would expect since neither $\widehat{F}$ nor $\widehat{G}$ were very good at picking up interactions at this range and $\widehat{J}$ is merely a composite of the two.

The $\widehat{K}$ function, plotted in Figure 4.12, seems to be by far the best at picking up both long and short range interactions accurately, the peaks falling in the ranges $(r_1, 2r_1)$ and $(r_2, 2r_2)$ as we might hope. Plots of the $\widehat{K}$ function are therefore probably best for getting rough estimates for the interaction radii for the pseudo-likelihood estimation discussed in Section 4.4.3. We have also plotted approximate 95% pointwise confidence intervals for these curves.

The $\widehat{T}$ function, plotted in Figure 4.13, seems to pick up repulsion far more easily than attraction and also seems to be very susceptible to erratic behaviour for small values of the parameter r, though this may be being amplified by the transformation applied. It does, however, pick up both the clustered and regular parts of the model, though by examining the approximate pointwise confidence intervals we see that all values below around $r = 0.03$ should be taken with a pinch of salt.

In summary, plots 4.3—4.13 show that the attractive-repulsive model is indeed capable of exhibiting clustered behaviour at one scale and repulsive behaviour at another. A number of standard graphical summaries were considered. Among these, the most powerful test function for discriminating this type of behaviour appears to be the $\widehat{K}$ function.

Minimum inter-event distances were also computed for each of the three processes for each of the 1000 realisations. Table 4.1 shows the number of realisations for each process which would cause rejection of complete spatial randomness at the 5%, 1% and 0.1% levels. The table makes it clear that minimum inter-event distances are not an ideal way to distinguish between these models.

4.4.3 Estimation of parameters

Although a similar result to that of Section 4.3.3 applies to our attractive-repulsive process, Graphs 4.9 and 4.10 show us that the range of values of the argument of

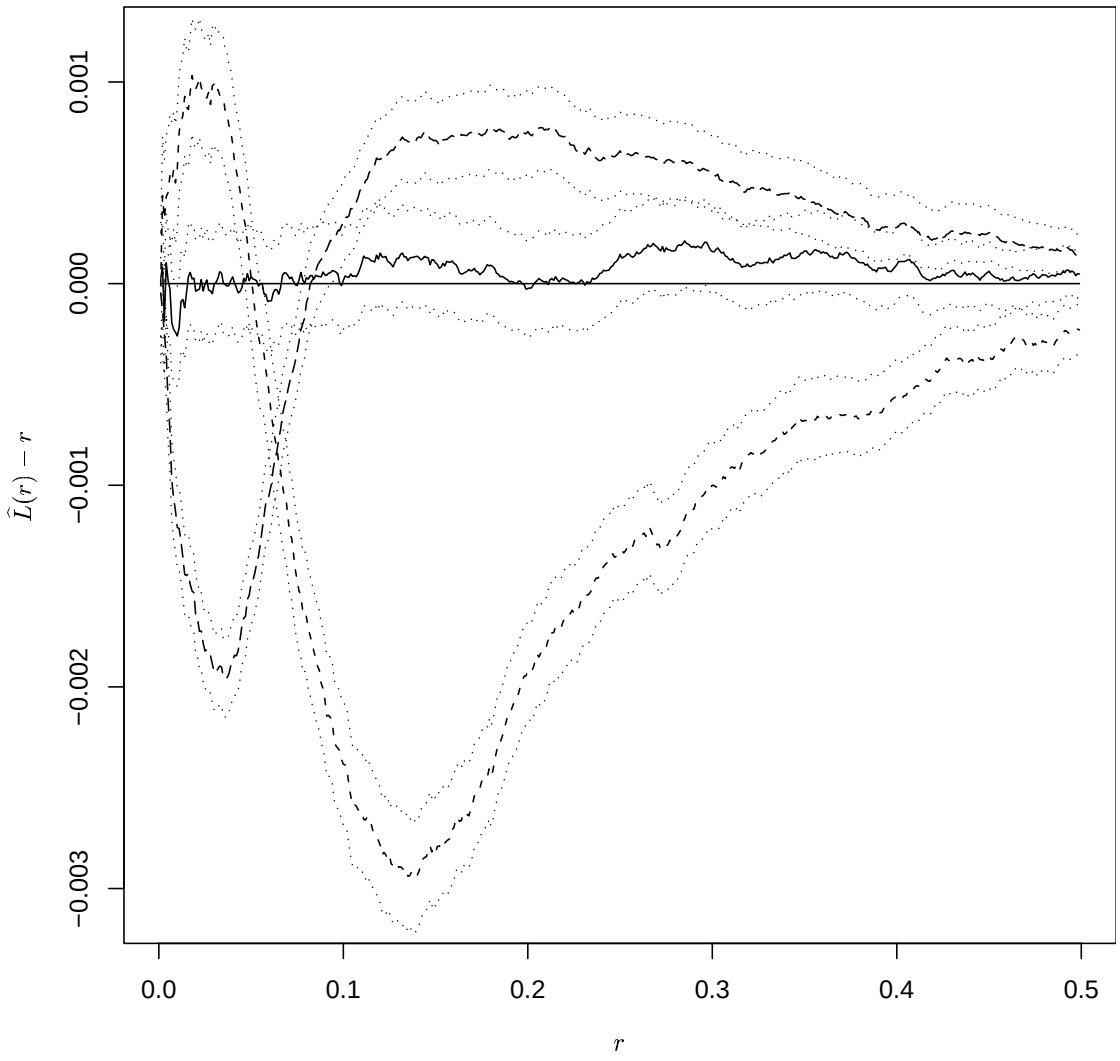

Figure 4.12: Plot of the L function averaged over 1000 realisations of a Poisson process (solid line), an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ (short dashes) and an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ (long dashes). For clarity we have plotted $\widehat{L}(r) - r$ versus r rather that $\widehat{L}(r)$ versus r. Dotted lines give approximate 95% pointwise confidence intervals for the three curves, calculated as ± 1.96 estimated standard errors.

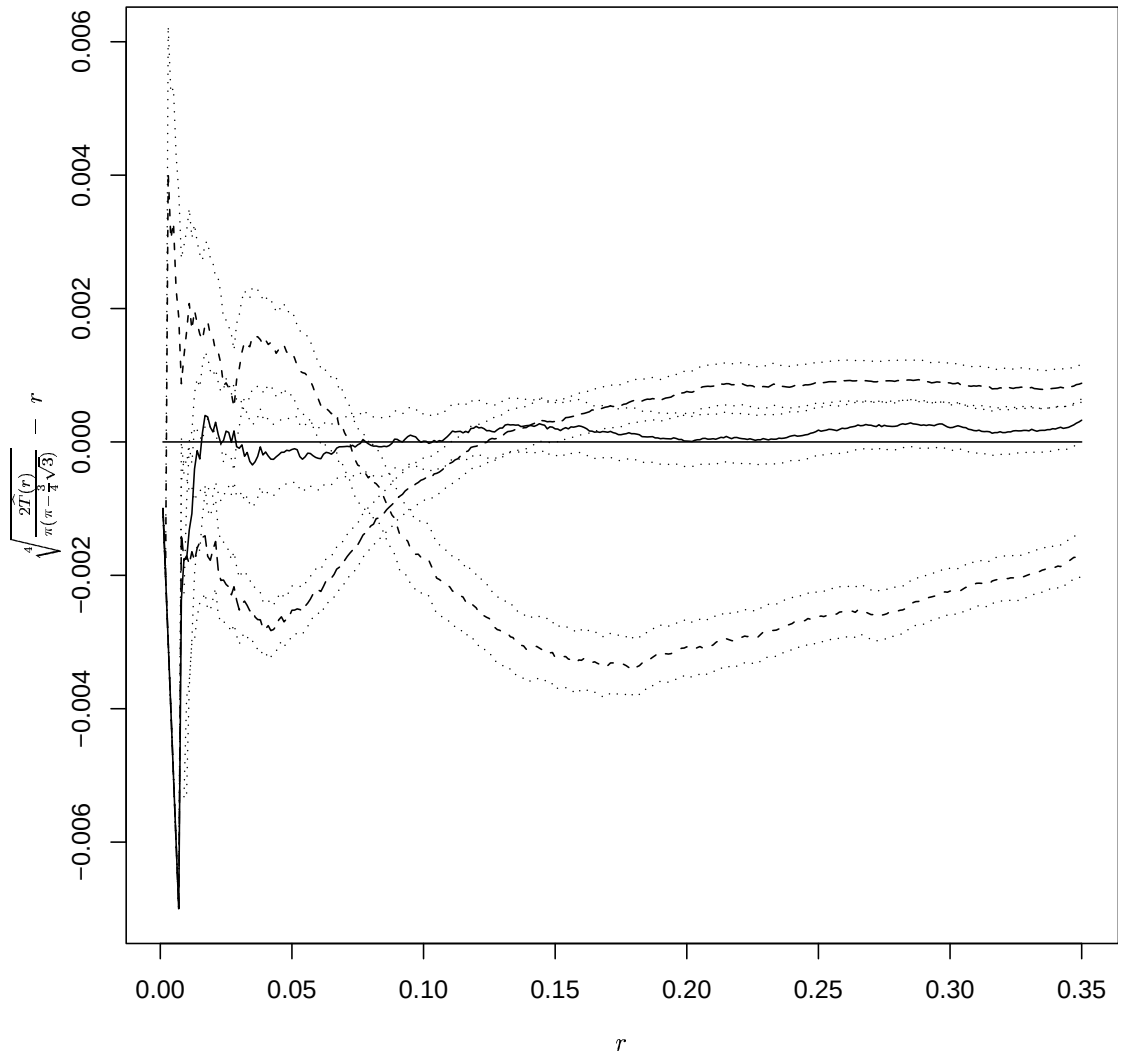

Figure 4.13: Plot of the $\widehat{T}$ function averaged over 1000 realisations of a Poisson process (solid line), an attractive-repulsive process with parameters $\gamma_1 = 100$, $\gamma_2 = 0.1$, $r_1 = 0.03$ and $r_2 = 0.1$ (short dashes) and an attractive-repulsive process with parameters $\gamma_1 = 10$, $\gamma_2 = 0.01$, $r_1 = 0.1$ and $r_2 = 0.03$ (long dashes). For clarity we have plotted $\sqrt[4]{\frac{2\widehat{T}(r)}{\pi(\pi-\frac{3}{4}\sqrt{3})}} - r$ versus r rather that $\widehat{T}(r)$ versus r. Dotted lines give approximate 95% pointwise confidence intervals for the three curves.

Statistic	Process	5%	1%	0.1%
	Poisson	55	9	0
d_1	Att-Rep	42	8	1
	Rep-Att	73	14	0
	Poisson	45	10	0
d_2	Att-Rep	46	9	1
	Rep-Att	72	17	3
	Poisson	44	12	0
d_3	Att-Rep	44	12	2
	Rep-Att	66	18	0
	Poisson	35	7	1
d_4	Att-Rep	49	6	2
	Rep-Att	67	19	0

Table 4.1: The table shows the number (out of 1000) of simulated realisations of the three processes discussed in Section 4.4.2 which cause a null hypothesis of complete spatial randomness to be rejected at the 5%, 1% and 0.1% levels. Att-Rep refers to the process which had small scale attraction and large scale repulsion, while Rep-Att refers to the process with small scale repulsion and large scale attraction.

$\widehat{F}$ and $\widehat{G}$ for which we are able to estimate the functions reliably only extend to about 0.15. Since this is less than $2r$, we are unable to make use of this result.

We may still, however, be able to make use of maximum pseudo-likelihood to estimate the parameters λ, γ_1 and γ_2, since the model is from an exponential family.

As we saw in Section 4.4.1, the Papangelou conditional intensity of our process is

$$\lambda(u; X) = \lambda \gamma_1^{-m((u \oplus G_1) \setminus X \oplus G_1)} \gamma_2^{-m((u \oplus G_2) \setminus X \oplus G_2)}.$$

Taking logs this gives

$$\begin{aligned} \log \lambda(u; X) &= \log \lambda - m((u \oplus G_1) \setminus X \oplus G_1) \log \gamma_1 - \\ &\qquad m((u \oplus G_2) \setminus X \oplus G_2) \log \gamma_2. \end{aligned}$$

Thus the pseudo-likelihood equations for this model are

$$\sum_{x_i \in A} \frac{1}{\lambda} = \int_A \gamma_1^{-m((u \oplus G_1) \setminus X \oplus G_1)} \gamma_2^{-m((u \oplus G_2) \setminus X \oplus G_2)} du, \quad (4.15)$$

$$\sum_{x_i \in A} \frac{m((x_i \oplus G_1) \setminus X \oplus G_1)}{\gamma_1} = \int_A m((u \oplus G_1) \setminus X \oplus G_1) \times \lambda \gamma_1^{-m((u \oplus G_1) \setminus X \oplus G_1)} \gamma_2^{-m((u \oplus G_2) \setminus X \oplus G_2)} du \quad (4.16)$$

and

$$\sum_{x_i \in A} \frac{m((x_i \oplus G_2) \setminus X \oplus G_2)}{\gamma_2} = \int_A m((u \oplus G_2) \setminus X \oplus G_2) \times \lambda \gamma_1^{-m((u \oplus G_1) \setminus X \oplus G_1)} \gamma_2^{-m((u \oplus G_2) \setminus X \oplus G_2)} du, \quad (4.17)$$

where we recall that A is an arbitrary subset of the window in which we observe the point process. Clearly the main difficulty is in estimating the integrals on the right hand side of equations (4.15) to (4.17). Baddeley and Turner (2000) tackle this problem directly by noting that the integral in the log pseudo-likelihood

$$\log \mathrm{PL}(\theta; X) = \sum_{x_i \in A} \log \lambda(x_i; X) - \int_A \lambda(u; X) du$$

can be approximated by

$$\int_A \lambda(u; X) du \simeq \sum_{j=1}^{m} \lambda(u_j; X) w_j,$$

where u_j are points in A and w_j are quadrature weights. Using and extending an observation made by Berman and Turner (1992), they note that if the set $\{u_j : j = 1, \ldots, m\}$ contains all the events $\{x_i : i = 1, \ldots, n(X)\}$, then the log pseudo-likelihood may be approximated by

$$\log \mathrm{PL}(\theta; X) \simeq \sum_{j=1}^{m} (y_j \log \lambda_j - \lambda_j) w_j, \quad (4.18)$$

where $\lambda_j = \lambda(u_j; X)$, $y_j = z_j / w_j$ and

$$z_j = \begin{cases} 1 & \text{if} \quad u_j \in \{x_i : i = 1, \ldots, n(X)\} \\ 0 & \text{if} \quad u_j \notin \{x_i : i = 1, \ldots, n(X)\}. \end{cases}$$

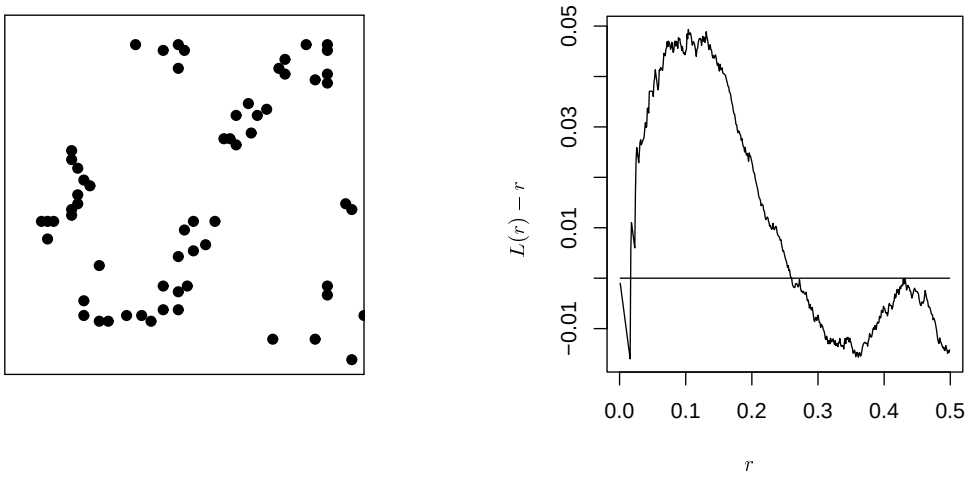

Figure 4.14: Redwood seedlings data. Left: The data, selected by Ripley (1977) from a larger data set analysed by Strauss (1975). Right: Plot of the L function for the redwood seedlings. There seems to be interaction at 3 different scales: (very) small scale repulsion followed by attraction at a moderate scale and then repulsion at larger scales.

For a fixed point pattern X the right hand side of (4.18) is equivalent to the log likelihood of independent Poisson variables $Y_k \sim \text{Poisson}(\lambda_k)$ taken with weights w_k, so (4.18) can therefore be maximised using standard software for fitting Generalised Linear Models, such as that in S-Plus.

In order to put the estimation procedure above into practice, we must have values for r_1 and r_2, the radii of G_1 and G_2 respectively. Baddeley and Turner (2000) suggest fitting the model for a variety of values of these "nuisance parameters" which do not fit into the exponential family model, and choosing the values which maximise the pseudo-likelihood. It may be wise to plot estimates of some standard functions such as K and G in order to narrow the search somewhat. See the previous section for more details.

4.5 Redwood seedlings

We take a brief look at a data-set which has been much-analysed in the literature, the Redwood seedlings data first introduced by Strauss (1975). We examine a

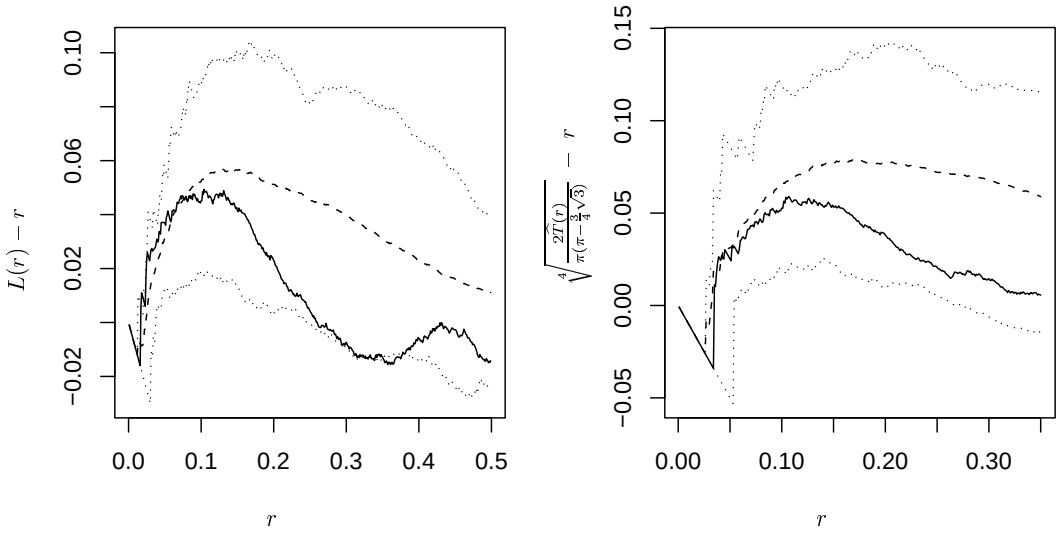

Figure 4.15: L and T function plots of the redwood seedlings data. Left: L-function plots of the data together with simulations of the attractive-repulsive model with parameters $R_1 = 0.07$, $R_2 = 0.013$, $\lambda = 0.118$, $\gamma_1 = 2000$ and $\gamma_2 = 10^{-200}$. Dotted lines give an envelope of 19 simulations of the model, the solid line is the redwood seedlings data and the dashed line is the average of the 19 simulations. Right: The same for the T function.

subset of the original data chosen by Ripley (1977) and later analysed by Diggle (1978) among others. The data are plotted in Figure 4.14. We wish to model this data using the attractive-repulsive model we have introduced. From an inspection of the estimated K-function (right pane in Figure 4.14) of the data using Ripley's edge correction scheme (Ripley 1977) we estimate values of R_1 and R_2 as 0.07 and 0.013 respectively, giving repulsion at small scales and attraction at moderate scales. It also seems that there is some repulsion at slightly larger scales, so it may be possible to use $R_2 = 0.2$ and to model the large scale interaction rather than the small scale interaction as we have chosen.

Fitting the remaining parameters by eye again, we chose values $\lambda = 0.118$, $\gamma_1 = 2000$ and $\gamma_2 = 10^{-200}$. The remarkably small value of γ_2 was necessary because the value of R_2 was also very small. It is clear from these numbers that it would be more natural to define γ_1 and γ_2 on a logarithmic scale. Figure 4.15 shows K and T function plots for 19 simulations from this model, providing

approximate 95% monte-carlo confidence envelopes for the values of the functions. It can be seen that on the basis of these functions, the model appears to fit the data reasonably well.

The plots show several things: Firstly that the model fits reasonably well, but that it is possible that we chose a value of R_1 which was slightly too large. Perhaps $R_1 = 0.06$ would have been better. Secondly, it seems that the large scale repulsion may an important factor which should not be ignored. Thirdly, in this case we have gained little new information by plotting the T function — the third order behaviour of the data seems to be similar to the second order structure.

4.6 Conclusions

We have developed an extension of the area-interaction process which incorporates both repulsion and attraction and given a method for perfect simulation of this process. We have shown using standard descriptive functions that the properties of this process are as we have claimed, and demonstrated a method for parametric inference for model fitting. We have also given a small application to the redwood seedlings data introduced by Strauss (1975).

It would be interesting to see how well our model would fit if we were to choose to model the large scale interactions in the redwood seedlings data rather than the small scale interactions. It would also be interesting to see how the automatic method of parametric inference discussed in Section 4.4.3 performed on this data set.

Chapter 5

Lattice Processes

Lattice processes are the discrete cousin of spatial point processes. These are interesting mainly because of the fact that in many cases real-world data is only sampled on a discrete grid, rather than a true continuous space. An example of lattice data would be the number of instances of some disease in each county in England. In contrast to spatial point processes, where we usually allow only single events at each location, with lattice data we usually have either multiple events at a single location, or a measurement of a continuous variable at each location. An example of where this is not true is in image analysis, where black and white images can be thought of as a lattice model with 'white' being represented by 'one event' and 'black' as 'no events'. For a more detailed treatment of lattice models see Cressie (1993). In this chapter we are particularly interested in extending the spatial processes of Baddeley and van Lieshout (1995) discussed in Chapter 4 to discrete spaces.

5.1 Area-interaction processes on discrete space

Before discussing these models we introduce some notation which will allow us to discuss these models with less confusion. The notation used in this chapter is not consistent with that used in Chapter 4. This is due to the fact that the material in this chapter is more concerned with simulation issues. As a result, words like 'event', 'point' and 'location' have natural uses which are distinct from their uses

in the previous chapter.

- A *location* is a vertex of the grid (or non-directed graph) upon which we are sampling.
- A *point* is an incident at a specific location in the grid. There may be multiple 'points' at one location at any given instant in time, or there may be none. For some models we will specify that the maximum number of points which may exist at a single location is one.

Following the work of Møller and Waagepetersen (1998) we refer to those models where the density can be written as a product of terms pertaining to each connected component within the set of possible locations and where there is only allowed to be a maximum of one point per location as *Markov connected-component fields*. The components are connected by the presence of points at neighbouring locations.

Let Z be the grid we are concerned with, for example Z might be a rectangular lattice, or a (undirected) binary tree, or any other non-directed graph. Also let $\chi = \{0,1\}^Z$ be the state space of possible configurations of points within the grid. Now let $\sim$ be a reflexive and symmetric relation on Z and $\mathcal{K}$ be the set of all possible subsets of Z which are *connected* by $\sim$, i.e. $K \in \mathcal{K} \iff \forall i,j \in K\ \exists x_1,\ldots,x_n$ such that $i = x_1 \sim x_2 \cdots x_{n-1} \sim x_n = j$. Now let $X \in \chi$ be a non-empty set of points on the grid Z (a realisation of some random process). We define a reflexive and symmetric relation $\sim_X$ based on $\sim$ which is the connected components relation on X by

$$i \sim_X j \iff \exists x_1,\ldots,x_n \in Z \text{ s.t. } i = x_1 \sim \cdots \sim x_n = j$$

and

$$\forall k \in 1,\ldots,n-1,\ (x_k \in X \text{ or } x_{k+1} \in X) \text{ (or both)}.$$

Then X is a *Markov connected-component field* if

$$p(X) = \alpha \prod_{K \in \mathcal{K}(X)} \Phi_K(X_K),$$

where X_K is X restricted to the set K rather than Z and $\mathcal{K}(X)$ is the set of maximally connected components in X, i.e. if $i \in K \in \mathcal{K}(X)$ then $i \sim_x j \implies j \in K$.

In the case of the area-interaction process we return to the notation used for the general area-interaction process given in equation (4.1) on page 52 and define the density by

$$p(X) = \alpha \lambda^{N(X)} \gamma^{-m(U(X))}, \tag{5.1}$$

where as in Section 4.1.2 we have $U(X) = \bigcup_{i=1}^{n} K(x_i)$. We define $K(x)$ to be x together with those locations which are connected to x by an edge[1]. If m is the counting measure we see that this model fits neatly into the definition of a Markov connected component field.

5.1.1 An attempt at perfect simulation

Since the state space of the model we have introduced in the previous section is finite it would be possible to simply write down the probabilities of each configuration and then invent a measure-preserving map from $[0, 1]$ to our new distribution, thus enabling us to generate draws from our distribution using $U[0, 1]$ random numbers, which we can simulate easily. This is often not practical, because although the state space is finite, it is often extremely large (if the grid χ upon which we are sampling has n locations then the state space has 2^n different configurations). Clearly it would be convenient if we could extend the perfect simulation of the continuous space area-interaction process (Kendall 1998) to the discrete model we have just proposed, since this would overcome the problems associated with the large state space. By natural extension we propose a binomial process with rate λ for the dominating process. We now discuss a suitable method for simulating this process.

Let r be some configuration of points on our grid of locations and let s be a

[1] This only transfers the issue to a different part of the modelling procedure, as we can often choose which locations to link by edges

configuration which differs from r at a single location x only. Let s have a point at x and r have no point there. Next, define $q(y, z)$ as the rate at which we move from state y to state z in a time-evolving process. If we now view r and s as two states in a time-evolving binomial process then $q(r, s)$ can be viewed as the birth rate at x and $q(s, r)$ can be seen as the death rate at this location. Since we wish to simulate a binomial process with rate λ it is clear that we must have

$$\frac{q(r,s)}{q(s,r)} = \lambda. \tag{5.2}$$

It is easy see that acceptable choices of the birth and death rates would be λ and 1 respectively, and indeed we always work with a death rate of 1 since this allows us to concentrate on the birth rate, just as in the continuous case. From the detailed balance equation

$$\pi(s)q(s,r) = q(r,s)\pi(r),$$

we see that equation (5.2) gives

$$\frac{\pi(s)}{\pi(r)} = \lambda. \tag{5.3}$$

Now let the probability that there is a point at a given location be ν, so that the probability that there is no point there is $(1 - \nu)$. Since we are considering a binomial process this value is the same for all locations in the grid. If we label the locations $x_1, \ldots, x_n$ then by independence

$$\pi(s) = \prod_{i=1}^{n} p(x_i = s_i),$$

where s_i is the state of location x_i under configuration s. Also

$$\pi(r) = \prod_{i=1}^{n} p(x_i = r_i),$$

where again r_i is the state of location x_i under configuration r. By our initial definitions of r and s we have $r_i = s_i \; \forall i$ s.t. $x_i \neq x$ and $p(\text{point at } x) = \nu$, so that we have

$$\pi(s) = \nu \prod_{x_i \neq x} p(x_i = s_i)$$

and

$$\pi(r) = (1-\nu)\prod_{x_i \neq x} p(x_i = s_i)$$

so that

$$\frac{\pi(s)}{\pi(r)} = \frac{\nu \prod_{x_i \neq x} p(x_i = s_i)}{(1-\nu)\prod_{x_i \neq x} p(x_i = s_i)} = \frac{\nu}{1-\nu}.$$

Thus from (5.3) we have

$$\frac{\nu}{1-\nu} = \lambda$$

and so

$$\nu = \frac{\lambda}{1+\lambda}.$$

From this information it is easy to see that we can modify the algorithm in Section 4.1.3 for simulating a time-evolving Poisson process to simulate a time-evolving binomial process in the following way.

We begin by generating an initial configuration by allowing there to be a point at each location with probability $\lambda/(1+\lambda)$. To do this we generate a $U[0,1]$ number for each location in turn. If this is less than $\lambda/(1+\lambda)$ then there is a point at that location (and if not then there is no point there).

Starting from this initial configuration we then simulate the process through time using parallel birth-death processes at each location. To do this we generate an exponential(1) number for each location which has a point in the initial configuration. These are the death times of these points. We then generate an exponential(λ) number for each of the locations which does not have a point in the initial configuration. These are the birth times of new points which will be at these locations.

We then begin evolving the process through time. When we reach a time when there is an event (either a birth or a death) we add a point to the location where the event is happening if we have reached a birth time and remove a point from that location if we have reached a death time. If it was a birth then we generate an exponential(1) number, add this to the current time and record this as the death time of the point. If it was a death then we generate an exponential(λ) number,

add this to the current time and record this as the birth time of the point. We then continue on to the next event and repeat the above procedure. At any time we may stop and we will have a draw from a binomial process with rate λ.

Now that we have a method for simulating a time-evolving binomial process perfectly we can attempt to use this to dominate a discrete area-interaction process. Before discussing this we introduce some notation and terminology which enables us to express this algorithm and the algorithms of the following sections more clearly.

- Recalling the notation on page 92, a *point* is an incident at a specific location in a birth-death process which is simulated on the grid. This 'point' has a birth time and a death time.

- M is a positive number (usually an integer, though this is not a requirement). As is usual in dominated CFTP we initially pick M and simulate a dominating process backwards in time from time 0 to time $-M$. We then simulate candidate maximum and minimum processes (which may or may not be true maximum and minimum processes) from time $-M$ forwards to time 0 using the transitions of the dominating process (i.e. the candidate maximum, candidate minimum and dominating processes are *coupled*). If the candidate maximum and candidate minimum processes are equal at time 0 then our work is done. Otherwise we double M and repeat the process, keeping the transitions of the dominating process which we have already generated.

- The *next event* at a given location is the next time (moving forwards in time) that there is a positive probability that the state of the candidate maximum process will change at that location, given the transitions of the dominating process. The next event after time $-M$ is also called the *first event*.

As we have just discussed, we begin by simulating a binomial process with rate λ (as detailed above) backwards in time from time $t = 0$ to time $t = -M$. Each

point which is alive at any time during $[-M, 0]$ is given a $U[0, 1]$ mark. We then simulate our candidate maximum and candidate minimum processes forwards in time according to the following rules.

The state of the candidate maximum process at time $t = -M$ is the same as the state of the dominating binomial process. The candidate minimum process is like the maximum process except that we reject every point whose mark is greater than $\gamma^{-M(\chi)}$, where

$$M(\chi) = \max_{\chi}(m(K(x))) \tag{5.4}$$

and χ is the grid of locations upon which we are simulating the process. More formally, the two processes can be defined at time $t = -M$ by

$$\begin{aligned} Y^{max}(-M, -M) &= \{x \ : x \in Z(-M)\} \text{ and} \\ Y^{min}(-M, -M) &= \left\{x \ : x \in Z(-M) \text{ and } P(x) \leq \gamma^{-M(\chi)}\right\}, \end{aligned}$$

where $Z(t)$ is the dominating time-evolving binomial process and $Y(t_1, t_2)$ is the candidate process which started at time t_1 and is currently at time t_2.

We then evolve the two new processes using the same method as was applied for the continuous area-interaction process, namely we proceed forwards in time towards time $t = 0$ using the following algorithm:

At each time u in $[-T, 0]$ assume that the processes have been generated up to that time, and suppose that the next birth or death to occur happens at time t_i. Consider the candidate maximum process $Y^{max}(-M, u)$ and the candidate minimum process $Y^{min}(-M, u)$ in turn and use the following rules to update both of them, where $Y(-M, u)$ represents whichever process we are currently considering.

If the next event is a **birth** in the dominating process then we accept the birth if

$$P(x) \leq \gamma^{-m(K(x) \setminus U(Y(-M,u))}$$

where x is the point to be born.

If, however, the next event is a **death** in the dominating process then we remove

the dying point from our processes if it was there in the first place, setting

$$Y(-M, t_i) = Y(-M, u) \setminus \{x\}.$$

All that now remains is to define $Y(-M, u + \varepsilon) = Y(-M, u)$ for all ε such that $u < u + \varepsilon < t_i$.

If $Y^{max}(-M, 0) = Y^{min}(-M, 0)$ then we are finished. If not, we must double M and try again, keeping the transitions of the dominating process which we have already generated.

Intuitively appealing as this process is, it fails to generate draws from the desired distribution. The generation of the dominating time-evolving binomial process is correct — it does indeed produce draws from the desired distribution, but the acceptance/rejection step to get from this to the area-interaction process is wrong. To see the reason for this we look at a simple two point example.

5.1.2 Example: A two point model

Here we consider a two location example with parameters $\lambda = 1$ and $\gamma = 2$. The two locations are labelled '1' and '2'. We also assign $K(x) = \{0, 0\}$ if x is either location and there is no point there and $K(x) = \{1, 1\}$ if x is either location and there is a point there (thus $U(X) = \{0, 0\}$ if there is a point at neither location and $U(X) = \{1, 1\}$ if there is a point at either location, or if there are points at both locations). Let m be the counting measure. By simply plugging these values into equation (5.1) we find that $\alpha = 4/7$. Thus the probability distribution of this process should be

$$p(\{0,0\}) = \frac{4}{7}; \quad p(\{1,0\}) = \frac{1}{7}; \quad p(\{0,1\}) = \frac{1}{7}; \quad p(\{1,1\}) = \frac{1}{7}.$$

Since the process is generated from a binomial process we consider a model with the nine states[2] $x_1, \ldots, x_9$ shown in Table 5.1.

[2] Since there are in effect four locations (two in the dominating process and two in the dominated process), and each location can be in one of two possible states we actually have $2^4 = 16$ different states. However due to the nature of the model there are seven states

		State of Dominating process		State of Dominated process	
		1	2	1	2
State	x_1	0	0	0	0
	x_2	1	0	0	0
	x_3	0	1	0	0
	x_4	1	1	0	0
	x_5	1	0	1	0
	x_6	1	1	1	0
	x_7	0	1	0	1
	x_8	1	1	0	1
	x_9	1	1	1	1

Table 5.1: Possible states of our example process

Denote equilibrium probabilities

$$p(x_i) = p_i.$$

Since the birth and death rates are equal in this example, the equilibrium distribution of our Markov process will be equal to the equilibrium distribution of the jump chain of our process. This means that we can calculate that equilibrium distribution using transition probabilities rather than transition rates, which makes the calculations conceptually slightly easier. From detailed balance we know that

$$p_1 \times p(x_1, x_2) = p(x_2, x_1) \times p_2,$$

where $p(x_1, x_2)$ and $p(x_2, x_1)$ are the transition probabilities of the jump chain of our Markov process. Now $p(x_1, x_2) = \frac{1}{2} \times (1 - \gamma^{-m(K(x) \setminus U(Y(-T,u)))})$ and $p(x_2, x_1) = \frac{1}{2}$, so we have

$$\frac{3}{8} p_1 = \frac{1}{2} p_2.$$

Similar analysis leads to equations relating the other states, which can be solved (together with the equation $\sum_{i=1}^{9} p_i = 1$) to give the following probability distribution:

(such as there being no points in the dominating process but points at both locations in the dominated process) which have by definition zero probability of happening. Due to this fact we only considered the other nine states.

State	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Probability	$\frac{16}{67}$	$\frac{12}{67}$	$\frac{12}{67}$	$\frac{9}{67}$	$\frac{4}{67}$	$\frac{3}{67}$	$\frac{4}{67}$	$\frac{3}{67}$	$\frac{4}{67}$.

By summing components this gives

$$p(\{0,0\}) = \frac{49}{67}; \quad p(\{1,0\}) = \frac{7}{67}; \quad p(\{0,1\}) = \frac{7}{67}; \quad p(\{1,1\}) = \frac{4}{67},$$

which is clearly not the desired distribution.

This raises the obvious question of *why* the methodology does not work when discretized in this way. Although it has been easier to show that the equilibrium probabilities are incorrect using transition *probabilities*, it is easier to see why they are incorrect by once more considering transition *rates*. Consider state x_2, where the dominating process has a point at location 1 but the dominated process does not. Consider also that the transition rate of the dominated process should be the same as that of the true process (as it should in all states). Thus we should have

$$q_{desired}(\{0,0\},\{1,0\}) = \lambda\gamma^{-m(K(x)\setminus U(Y(-M,u)))}.$$

However, since the dominating process is in state $\{1,0\}$, the transition rate is actually

$$q_{actual}(\{0,0\},\{1,0\}) = (1+\lambda)\gamma^{-m(K(x)\setminus U(Y(-M,u)))},$$

since we can only move to state x_5 via state x_1. This obviously leads to smaller probabilities that there are points at the locations on the grid, as we saw above.

In an attempt to overcome the difficulties encountered by attempting to dominate the discretized area-interaction process with a binomial process, we now define a new process which may have multiple points at each location.

5.2 A Poisson process on a finite discrete space

Let χ be a finite grid (or lattice) of locations. The discrete Poisson process with rate λ is simply a random configuration of points over χ such that each location has a Poisson(λ) number of points, independently of the other locations. This is

a simplified version of what is more commonly known as the auto-Poisson model. See Cressie (1993) for details. We call this model a discrete Poisson process because it can be viewed as a 'windowing' of a continuous Poisson process, with the number of points at a given location equal to the number of points in the window corresponding to that location. It should be noted that since we now allow multiple points at each location this is no longer a Markov connected component field.

If we are to use this in dominated CFTP we must have a method for evolving the process through time. Using the notation of Section 5.1.1, we again let r and s be two different configurations of points, this time with the possibility of having multiple points at single locations. Again, s has one more point in it than r at some location x and is otherwise identical. Clearly

$$\pi(s) = \prod_{i=1}^{n} p(s_{x_i})$$

and

$$\pi(r) = \prod_{i=1}^{n} p(r_{x_i}),$$

where again $x_1, \ldots, x_n$ are the locations in χ, and s_{x_i} and r_{x_i} are the states of location x_i under s and r respectively. Our initial definitions of r and s again enable us to simplify the above equations to

$$\pi(s) = e^{-\lambda} \frac{\lambda^{N_s(x)}}{N_s(x)!} \prod_{x_i \neq x} p(s_{x_i})$$

and

$$\pi(r) = e^{-\lambda} \frac{\lambda^{N_s(x)-1}}{(N_s(x)-1)!} \prod_{x_i \neq x} p(s_{x_i}),$$

where $N_s(x)$ is the number of points at location x under configuration s. Appealing once more to the detailed balance equation (4.4), we see once more that

$$\frac{q(r,s)}{q(s,r)} = \frac{e^{-\lambda} \frac{\lambda^{N_s(x)}}{N_s(x)!} \prod_{x_i \neq x} p(s_{x_i})}{e^{-\lambda} \frac{\lambda^{N_s(x)-1}}{(N_s(x)-1)!} \prod_{x_i \neq x} p(s_{x_i})} = \frac{\lambda}{N_s(x)}.$$

Thus by basic theory it is possible to simulate a time-evolving version of this process using a birth-death process with each *location* having a birth rate of λ and

each *point* having a death rate of one. Readers familiar with queueing theory will recognise this as a collection of independent (M, M, ∞) queues.

We now extend the attractive-repulsive process defined in Section 4.4 to discrete space and use the discrete Poisson process to enable us to perfectly simulate the new model.

5.3 The attractive-repulsive process on discrete space

Let $d(X)$ be the number of locations in a configuration X with at least one point, and $N(X)$ be the number of points in X including multiplicity. For this model let

$$p(X) = \alpha\lambda^{N(X)}\gamma_1^{-m(U(X))}\gamma_2^{-m(X\oplus G)} \tag{5.5}$$

with respect to the unit rate discrete Poisson process[3], where we take G sufficiently small so that $(x \oplus G)$ is simply $\{x\}$ in this discrete setting. We return once more to the notation $U(X)$ for the exponent of γ_1 as we did when describing the discrete area-interaction process on page 93, and note that if there are multiple points at a given location then $U(X)$ is no different than if there were only a single point there. As before m is counting measure. With G as above we see that

$$\gamma_2^{-m(X\oplus G)} = \gamma_2^{-d(X)},$$

and letting $N^*(X) = N(X) - d(X)$, equation (5.5) becomes

$$p(X) = \alpha\left(\frac{\lambda}{\gamma_2}\right)^{N(X)}\gamma_1^{-m(U(X))}\gamma_2^{N^*(X)}. \tag{5.6}$$

[3] This is a slight departure from the way we have previously defined discrete distributions. In the past all definitions have been probability mass functions — i.e. they have been stated with respect to some suitable counting measure (see Sections 2.2 and 2.4 for a brief discussion of what it means to define a probability measure with respect to some other measure). Equation (5.5) can also be written in this form as follows:

$$p(X) = \alpha\left(\prod_{i=1}^{n}\frac{\lambda^{N(x_i)}}{N(x_i)!}\right)\gamma_1^{-m(U(X))}\gamma_2^{-m(X\oplus G)} = \alpha\lambda^{N(X)}\gamma_1^{-m(U(X))}\gamma_2^{-m(X\oplus G)}\prod_{i=1}^{n}\frac{1}{N(x_i)!}.$$

The reader should note how little clarity is added by doing so.

This model is not a Markov connected component field as it allows multiple instances of a single point.

5.3.1 Simulation

Perfect simulation of the above process is actually simpler than the method for simulating the point process version of the attractive-repulsive process. We begin by noting that the discrete Poisson process with rate

$$\frac{\lambda}{\gamma_2}$$

dominates (5.6). Thus we begin by simulating a discrete Poisson process with rate λ/γ_2 as discussed in Section 5.2 and use this as our configuration at time 0. We then continue with the methods discussed in that section to evolve the process backwards until some time $-M$ using a birth-death process with death rate equal to 1. For consistency of notation, again let $Z(t)$ denote the configuration of points in this process at time t. Just as in the standard area-interaction process algorithm we then give all the points that exist in the model in the interval $[-M, 0]$ marks generated from a $U[0,1]$ distribution. Next we recursively define two new processes Y^{max} and Y^{min} just as in Section 4.4.1 but with a few minor adjustments:

We begin by defining the configurations at time $-M$:

$$\begin{aligned} Y^{max}(-M,-M) &= \{x \ : x \in Z(-M)\} \\ Y^{min}(-M,-M) &= \left\{x \ : x \in Z(-M) \text{ and } P(x) \leq \gamma_1^{-M(\chi)}\gamma_2\right\} \end{aligned} \tag{5.7}$$

where $P(x)$ is the $U[0,1]$ mark given to the point x and

$$M(\chi) = \max_{\chi}(m(K(x)))$$

as on page 97 in Section 5.1.1.

By examining the detailed balance equation (4.4) we see that that the birth rate of our process should be

$$q(X, X \cup \{x\}) = \frac{\lambda}{\gamma_2}\gamma_1^{-m(K(x)\setminus U(X))}\gamma_2^{N^*(X\cup\{x\})-N^*(X)},$$

where X is the state of the process prior to a birth and x is a new point being born. Since we are simulating this process with respect to a Poisson process with rate λ/γ_2 this means that as we simulate the maximum and minimum processes forwards in time we should accept births if

$$P(x) \leq \gamma_1^{-m(K(x)\setminus U(X))}\gamma_2^{N^*(X\cup\{x\})-N^*(X)}. \tag{5.8}$$

We may now see that the bound on $P(x)$ given in (5.7) is correct, as it is the minimum of equation (5.8). Unlike the point process version of the attractive-repulsive process, using (5.8) as the acceptance probability does not break the monotonicity of the process. Having discussed the birth-behaviour of our process in some detail, we proceed by spelling out the algorithm as a whole.

The processes are generated forwards in time to time $t = 0$ in the following way:

At each time u in $[-M, 0]$ assume that the processes have been generated up to that time, and suppose that the next birth or death to occur happens at time t_i. If the next event is a **birth**, then we accept the birth for our Y processes if

$$P(x) \leq \gamma^{-m(K(x)\setminus U(Y(-M,u)))}\gamma_2^{N^*(Y(-M,u)\cup\{x\})-N^*(Y(-M,u))}$$

where x is the point to be born and Y is either Y^{max} or Y^{min} depending upon which process we are generating.

If, however, the next event is a **death**, then we remove the dying point from our processes (if it was actually in either process), setting

$$Y(-M, t_i) = Y(-M, u) \setminus \{x\}.$$

All that now remains is to define $Y(-M, u + \varepsilon) = Y(-M, u)$ for $u < u + \varepsilon < t_i$.

If these two processes are identical at time zero (i.e. if $Y^{max}(-M, 0) = Y^{min}(-M, 0)$), then we have the required sample from the discretized attractive-repulsive process. If not we extend the underlying Poisson process back in time to time $-(M + S)$, generate a few more $U[0, 1]$ marks (keeping the ones already generated), and start again generating the processes forwards to time $t = 0$ again.

The same reasoning which led us to see that the algorithm in Section 4.1.4 correctly simulated the desired distribution also applies to this new process, since the true process will again be sandwiched in between the maximum and minimum processes.

In the next section we show how an extension of this model leads to another attempt at perfectly simulating the discrete area-interaction process.

5.4 Perfect simulation of a discrete area-interaction process revisited

Here we present a second attempt at perfectly simulating a discretized area-interaction process which exemplifies the second way in which perfect simulation can fail when using dominated CFTP.

We begin by modifying the attractive/repulsive process so that the density is

$$p(X) = \alpha \lambda_1^{N(X)} \gamma^{-m(U(X))} \gamma_2^{-m_2(X)} \tag{5.9}$$

with respect to the unit rate discrete Poisson process, where

$$\begin{aligned} m_2(X) &= N(X) \quad \text{if } N(X) = d(X) \\ &= \infty \qquad \text{if } N(X) > d(X) \end{aligned}$$

and $\gamma_2 \in (1, \infty)$. We see that m_2 is a measure since it is clearly additive and we also see that it is dominated by the discrete Poisson process with density

$$p(X) = \alpha \left(\frac{\lambda_1}{\gamma_2} \right)^{N(X)} \tag{5.10}$$

with respect to the unit rate discrete Poisson process.

Clearly the density (5.9) is equivalent to the discretized area-interaction process, since it gives zero probability to configurations with more than one instance of a given point and is otherwise identical (except for the rate being λ_1/γ_2, which can be corrected by scaling λ_1 prior to simulation). It may also be amenable to perfect simulation, since the discrete Poisson process (5.10) allowing multiple instances of a point may be used as a dominating process.

To see this we examine the detailed balance equation

$$\pi(r)q(r,s) = \pi(s)q(s,r)$$

with $s = r \cup \{p\}$ for p a point at some location x on our grid. These values for r and s mean that $q(r,s)$ is the birth rate at location x and $q(s,r)$ is the death rate $(= N_s(x))$ there. Thus we have

$$\begin{aligned}
\alpha\lambda_1^{N(r)}\gamma^{-m(U(r))}\gamma_2^{-m_2(r)}q(r,s) &= q(s,r)\alpha\lambda_1^{N(s)}\gamma^{-m(U(s))}\gamma_2^{-m_2(s)} \\
\implies \quad \lambda_1^{N(r)}\gamma^{-m(U(r))}\gamma_2^{-N(r)}q(r,s) &= N_s(x))\lambda_1^{N(r)+1}\gamma^{-m(U(s))}\gamma_2^{-m_2(s)} \\
\implies \quad q(r,s) &= N_s(x)\lambda_1\gamma^{-m(K(p)\setminus U(r))}\frac{\gamma_2^{-m_2(s)}}{\gamma_2^{-N(r)}}.
\end{aligned}$$

Now if $N_r(x) = 0$ then $\gamma_2^{-m_2(s)} = \gamma_2^{-(N(r)+1)}$ and $N_s(x) = 1$, so

$$q(r,s) = \frac{\lambda_1}{\gamma_2}\gamma^{-m(K(p)\setminus U(r))}.$$

If $N_r(x) > 0$ then $\gamma_2^{-m_2(s)} = \gamma_2^{-\infty} = \gamma_2^{-(N(r)+1)}\gamma_2^{-\infty}$ and so

$$q(r,s) = N_s(x)\frac{\lambda_1}{\gamma_2}\gamma^{-m(K(p)\setminus U(r))}\gamma_2^{-\infty} = 0.$$

Thus

$$q(r,s) = \frac{\lambda_1}{\gamma_2}\gamma^{-m(K(p)\setminus U(r))}\gamma_2^{-m_3(\{p\}\setminus r)}$$

where

$$\begin{aligned}
m_3(X) &= 0 \quad \text{if } X \neq \phi \\
&= \infty \quad \text{if } X = \phi.
\end{aligned}$$

Making continued use of the terminology discussed on page 96, our next attempt at perfect simulation of the discretized area-interaction process is as follows:

1. For each location i in turn:
   (a) Generate a Poisson(λ) random variable. This is the number of points in the dominating process which are alive at location i at time 0.
   (b) For each of these points generate an Exponential(1) random variable. This is the lifetime of the point. If any of these values is greater than M then there is zero probability of coalescence, so return to the start (point 1), double M and try again.

(c) Beginning at time $t = 0$ generate:

i. An Exponential(λ) random variable and add this to t. This is -1 times the death time of a point.

ii. An Exponential(1) random variable. Subtracting this from the death time gives the birth time of the point whose death time we just found.

This should be repeated until either $t > M$ or the birth time of a point is less than $-M$.

(d) If we stopped because of a birth time being less than $-M$ then the death time of this point should be stored as the first event to happen at location i and the candidate maximum process should be given a value of 1 (alive) at that location at time $-M$.

(e) If we stopped because $t > M$ then the smallest birth time after $-M$ is the first event to happen at location i. The fact that $t > M$ means that the point which we have just considered died before time $-M$ and is thus not part of the process we are simulating. If there is no birth time after $-M$ then use zero as the next birth time. The candidate maximum process should be given the value '0' (dead) at location i at time $-M$.

(f) Set the candidate minimum process to 0 (dead) at location i at time $-M$.

2. Initialise $U(X)$ for the candidate maximum process using the values of the candidate maximum process at time $-M$ and set it to the empty set for the candidate minimum process.

3. Sort the 'next event' times for each location (at this stage these are obviously the 'first event' times).

4. Repeat:

(a) If the next event is a death at location j, set both processes to 0 (dead) at that location, update $U(X)$ for both processes and find the next birth (if there is one) which happens at location j. Set that as the next event to happen at that location.

(b) Otherwise (a birth) calculate $\gamma^{-m(K(p)\setminus U(X))}$ for each process (where p is the point being born) and use the point's mark to decide whether to allow the birth. If we reject the birth from the max process then let the next event at that location be the next birth which happens there (if there is one). If we accept the point in the max process then let the next event at that location be the death of that point.

(c) Insert the next event time we have just calculated in the correct place in the sorted list of next events.

Until we have dealt with all events at all locations.

5. If max and min processes are the same we have our perfect draw. Otherwise double M and return to the start (step 1).

5.4.1 Example revisited: The two point model

Returning to the example of Section 5.1.2 we treat the general case this time with the minor alteration that we let λ be the ratio of λ_1/γ_2 due to our new simulation method. We see that the single-step transition probabilities are now independent of the state of the dominating process, as points in that process can be born regardless of whether there is a point there already. From the definition of the process we see that

$$\begin{aligned} p(\{0,0\}) &= \alpha\lambda^0\gamma^{-0} = \alpha, \\ p(\{1,0\}) &= \alpha\lambda^1\gamma^{-2}, \\ p(\{0,1\}) &= \alpha\lambda^1\gamma^{-2} \text{ and} \\ p(\{1,1\}) &= \alpha\lambda^2\gamma^{-2}, \end{aligned}$$

which combined with the normalisation equation $p(\{0,0\})+p(\{1,0\})+p(\{0,1\})+p(\{1,1\})=1$ give

$$\alpha = \frac{\gamma^2}{\lambda^2+2\lambda+\gamma^2}.$$

Therefore the equilibrium probabilities are

$$p(\{0,0\}) = \frac{\gamma^2}{\lambda^2+2\lambda+\gamma^2}, \qquad p(\{1,0\}) = \frac{\lambda}{\lambda^2+2\lambda+\gamma^2},$$
$$p(\{0,1\}) = \frac{\lambda}{\lambda^2+2\lambda+\gamma^2} \qquad p(\{1,1\}) = \frac{\lambda^2}{\lambda^2+2\lambda+\gamma^2}.$$

It is easy to see that the at a given instant in time the probability that the next event to occur in the dominating process is a birth is given by

$$p(\text{next event is birth}) = \frac{\text{birth rate}}{\text{birth rate } + \text{ death rate}} = \frac{\lambda}{1+\lambda}.$$

This clearly gives

$$p(\text{next event is birth at point } p) = \frac{\lambda}{1+\lambda} \times \frac{1}{\#\text{ of points in grid}}$$

and since our model has two points this reduces to

$$p(\text{next event is birth at point } p) = \frac{\lambda}{1+\lambda} \times \frac{1}{2}.$$

An analogous argument shows that

$$p(\text{next event is death at point } p) = \frac{1}{1+\lambda} \times \frac{1}{2}.$$

From these equations and detailed balance it is easy to see that

$$\begin{aligned} p(\{0,0\}) \cdot \lambda \cdot \frac{1}{\gamma^2} &= p(\{1,0\}), \\ p(\{0,0\}) \cdot \lambda \cdot \frac{1}{\gamma^2} &= p(\{0,1\}) \text{ and} \\ p(\{1,0\}) \cdot \lambda \cdot 1 &= p(\{1,1\}). \end{aligned}$$

Combining these equations with the normalisation equation gives

$$p(\{0,0\}) = \frac{\gamma^2}{\lambda^2+2\lambda+\gamma^2}, \qquad p(\{1,0\}) = \frac{\lambda}{\lambda^2+2\lambda+\gamma^2},$$
$$p)\{0,1\}) = \frac{\lambda}{\lambda^2+2\lambda+\gamma^2}, \qquad p(\{1,1\}) = \frac{\lambda^2}{\lambda^2+2\lambda+\gamma^2}$$

as required. Thus the stationary distribution of the candidate maximum and candidate minimum processes is indeed the discretized area-interaction process.

5.4.2 What goes wrong

This now looks very promising, as we have a method for generating candidate minimum and candidate maximum processes whose stationary distribution is the discretized area-interaction process. Unfortunately this algorithm is also flawed. The reason for this comes back to the idea of *stochastic domination*, and is the same reason that the algorithm for sampling the attractive-repulsive process needed to be modified.

Recall from Section 4.1.4 that the reason that coupling from the past could be applied was that the maximum process dominated the true process and the minimum process was dominated by the true process. This meant that when the maximum and minimum processes were equal they were also equal to the true process. This is what fails for the candidate maximum and candidate minimum processes in our second algorithm, and we can see this most clearly by looking once again at our example.

5.4.3 Example revisited again

To see this we examine the case where the dominating process is in state $\{1, 1\}$ at time $-M$ (or in any state with at least one point at each location). Then the candidate maximum process will be in state $\{1, 1\}$ at time $-M$ and the candidate minimum process will be in state $\{0, 0\}$. Now the 'real' process (the discretized area-interaction process) could be in any of the four possible states at this point. Now suppose that the 'real' process was originally in state $\{0, 0\}$ and that there is a birth between time $-M$ and the first death time of any of the points which are alive in the dominating process at time $-M$. Then there is a non-zero probability that the 'real' process will no longer be between the candidate minimum and candidate maximum processes before the first transition of either process has even taken place! This is because there will be a point alive in the 'real' process (the one which has just been born) which is not alive in the candidate maximum process.

This clearly breaks the stochastic domination.

We now introduce a method which overcomes the difficulties highlighted in Sections 5.1 and 5.4.

5.5 A correct algorithm

A careful examination of the algorithm for simulating draws from the attractive-repulsive process (see Section 4.4.1) reveals a solution to the problems encountered above. First, we must allow multiple points to exist at a single location in the candidate maximum process and let the configuration of this process at time $-M$ be equal to the configuration of the dominating process. Second, we must allow points to be born in the candidate maximum and candidate minimum processes at any time where there is a birth in the dominating process. Third, the acceptance probability for births in the candidate maximum process should be

$$P(x) \leq \gamma^{-m(K(x)\setminus U(X^{max}))}\gamma_2^{-m_3(\{x\}\setminus X^{min})},$$

while the acceptance probability for births in the candidate minimum process should be

$$P(x) \leq \gamma^{-m(K(x)\setminus U(X^{min}))}\gamma_2^{-m_3(\{x\}\setminus X^{max})}.$$

Making these three changes restores the monotonicity at the expense of making the convergence times extremely long — it makes the probability of accepting a birth in the candidate minimum process zero whenever there is a point alive in the candidate maximum process. This means that a lower bound on the coalescence time is the time taken for the number of points in the candidate maximum process to reach zero at every location (although not necessarily at the same time). An upper bound is the time taken for there to be no live points anywhere in the candidate maximum process.

5.6 Conclusions

We have introduced some discrete analogues of the spatial point process models of Chapter 4 and shown how to simulate these models using dominated coupling from the past. We have also highlighted some of the important issues which require consideration when using dominated coupling from the past, giving examples to show how things go wrong if these issues are not considered.

In the following chapter, we use a discrete area-interaction process to model the distribution of significant wavelet coefficients. In this application we decided to drop the requirement that there should be a maximum of one point per location. Instead, we use the model whose density is given by

$$p(X) = \alpha\lambda^{N(X)}\gamma^{-m(U(X))}$$

with respect to the discrete Poisson process introduced in Section 5.2. This gives both faster convergence times and more flexible models, as it enables us to use the number of points at a given location as a scaling parameter, as we shall see in Chapter 6. It is easy to see how a simplification of the methods described in Section 5.3.1 leads to perfect simulation of the above process.

Chapter 6

An application to wavelet thresholding

6.1 Introduction

In this chapter we discuss wavelets and introduce an extension of Abramovich et al. (1998)'s wavelet-based procedure for reconstructing a signal when we observe a noisy version. We begin by introducing the *continuous wavelet transform*, before moving on to the *discrete wavelet transform*, which will be used extensively in this chapter. Much of this material comes from lecture notes taken from a course given by Guy Nason at the University of Bristol in 1999 and from Abramovich et al. (1998), who give an excellent introduction in their paper. Throughout this chapter we retain the notation of Abramovich et al. (1998) wherever appropriate.

The continuous wavelet transform maps a function $f \in L^2(\mathbb{R})$ onto its *wavelet series representation*. This is a way of expressing f in terms of a sum of scaled and shifted versions of a single function, ψ, called the *mother wavelet*:

$$f(t) = \sum_j \sum_k w_{jk} \sqrt{2^j} \psi(2^j t - k), \tag{6.1}$$

where $j \in \mathbb{Z}$ is the scale and $k \in \mathbb{Z}$ is the shift. It is clear that not every choice of ψ will generate an orthonormal basis for $L^2(\mathbb{R})$ when scaled and shifted in this way. It is, however, possible to choose ψ such that $\psi_{jk}(t) = \sqrt{2^j}\psi(2^j t - k)$ does indeed form an orthonormal basis of $L^2(\mathbb{R})$. The examples which we make use of in this chapter are Daubechies's least asymmetric wavelets (Daubechies 1992) and

the Haar wavelets. The numbers w_{jk} in equation (6.1) above are called the *wavelet coefficients* of f and are given by $\langle f, \psi_{jk} \rangle$:

$$w_{jk} = \int_{\mathbb{R}} f(t)\psi_{jk}(t)dt.$$

In practice we do not collect data on a continuum of points, but on a discrete grid (whose points, for the purposes of this chapter, we shall assume are equally spaced). We also do not observe perfectly, but with noise:

$$y_i = g(t_i) + \varepsilon_i. \tag{6.2}$$

Suppose that we observe a vector (sequence) $\mathbf{y}$ which has been sampled from (6.2) at $2^m = n$ points, where we assume that $g \in L^2(\mathbb{R})$. The *discrete wavelet transform* (DWT) is best introduced in terms of low- and high-pass filters, though we shall see that this leads to a discrete approximation of the continuous wavelet transform.

A *filter*, $\mathcal{F}$, is defined by a sequence $\{f_j\}$. Given our observed sequence $\mathbf{y}$, then

$$\mathcal{F}(y_i) = \sum_j f_j y_{i+j}.$$

Mallat (1989) developed an algorithm based on applying two filters and a decimation step recursively. In this context, *decimation* means keeping only the even numbered coefficients. The two filters, called "low"- and "high"-pass, recursively separate the sequence $\mathbf{y}$ into smooth and detailed parts at $m-1$ levels. This process corresponds to a discretized wavelet transform of $\mathbf{y}$. The relationship between the filter coefficients and the wavelets can be found in Section 5.6 of Daubechies (1992). The algorithm works as follows:

1. Set $j = 1$.
2. Repeat:
   (a) Apply the low- and high- pass filters to the sequence $\mathbf{y}$, obtaining two new sequences, $\mathbf{y}^d$ and $\mathbf{y}^c$ respectively. Intuitively, $\mathbf{y}^d$ will consist of how much the sequence changes at each sampling point and $\mathbf{y}_c$ will consist of how much it stays the same (d =difference; c =constant).

(b) Decimate both $\mathbf{y}^d$ and $\mathbf{y}^c$, obtaining two new sequences, $\mathbf{d}^{m-j}$ and $\mathbf{c}^{m-j}$ respectively, each of length 2^{m-j}.

(c) Keep $\mathbf{d}^{m-j}$. These are the discrete wavelet coefficients at level $m-j$.

(d) Replace $\mathbf{y}$ by $\mathbf{c}^{m-1}$ and add one to j.

Until $j = m$.

This procedure is known as a *multiresolution analysis*, since the sequence is recursively analysed at ever-coarser scales as the algorithm progresses. A key feature of the algorithm presented by Mallat (1989) is that it is fast — for a signal of length n, it takes only $O(n)$ computations to perform.

Daubechies (1992) gives the filter coefficients necessary to perform this analysis for the wavelets we used. The coefficients for the Haar wavelets are the simplest example, and are $-g_1 = g_2 = 1/\sqrt{2}$ for the high-pass filter and $h_1 = h_2 = 1/\sqrt{2}$ for the low-pass filter.

How does this relate to the continuous wavelet transform? If we express the filtering as a matrix, $\mathcal{W}$, and concatenate the discrete wavelet coefficients into a vector, $\widehat{\mathbf{d}}$, then we may express the DWT as follows:

$$\widehat{\mathbf{d}} = \mathcal{W}\mathbf{y}.$$

The reason for the hat ($\widehat{\ }$) is because $\mathbf{y}$ is observed with noise (equation (6.2)). The coefficients will then be an *estimate* of the "true", or *population* discrete wavelet coefficients, $\mathbf{d}$, which we would have obtained if we could have transformed the sequence of function values, $g(t_i)$. Returning to the question of how this corresponds to the continuous wavelet transform, the (jk, i)th entry of $\mathcal{W}$ is approximately

$$W_{jk,i}\sqrt{n} \approx \psi_{jk}(i/n) = \sqrt{2^j}\psi(2^j i/n - k).$$

It is important to note that the population discrete wavelet coefficients are not the same as the wavelet coefficients of the function, $g(t)$, but are approximately related by $d_{jk} \approx w_{jk}\sqrt{n}$.

A large class of functions can be represented sparsely by taking the wavelet transform of the given function with respect to the Daubechies or Haar bases mentioned above. This results in the majority of the signal being concentrated in a small number of the wavelet coefficients. In these situations a reasonable approach to removing noise from a signal would be to simply throw away all of the small coefficients. For Gaussian (white) noise, this scheme works because the transform is orthogonal. A property of orthogonal transformations is that white noise in the observations is transformed to white noise in the coefficients of the transform. Thus, since most of the 'true' signal is concentrated in a few large coefficients we will be throwing away mainly noise. This procedure of throwing away the small values is known as *thresholding*, and the most simple method is to pick a number, t, and to set all coefficients whose absolute value is smaller than t to zero. This is known as 'hard' thresholding. An alternative to hard thresholding, 'soft' thresholding, also shrinks towards zero the coefficients which are above the threshold, using the formula

$$T_{\text{SOFT}}(d,t) = \text{sgn}(d)(|d| - t)_+,$$

where $(x)_+ = xI(x > 0)$ is the positive part of x. Many methods exist for choosing the value of t for both hard and soft thresholding. We now briefly describe some of these methods.

SureShrink (Donoho and Johnstone 1995) is a method for soft thresholding which minimises Stein's unbiased estimate of risk (Stein 1981). A different threshold is chosen for each level of the transform. The authors prove SureShrink is near-optimal in the way that it adapts to the smoothness of the underlying function.

Cross-validation is a general method which has been used in a number of areas of statistics. The principle is to split the data set into two pieces, a test set and a training set. The training set is then used to to fit a model (in this case a function).

The test set is then used to assess the performance of the method. Nason (1996) suggests splitting the data into the odd- and even-numbered observations (we shall call them the 'odds' and the 'evens'). First the evens are used to get an estimator for the function (using some threshold t) and the sum of squared errors(SSE) between the estimate and the odds is calculated. Secondly, the odds are used to get an estimator of the function using the same threshold t, and the SSE between the new estimate and the evens is calculated. Finally, the combined SSE is then minimised numerically over values of t.

False discovery rates (Benjamini and Hochberg 1995) were originally introduced in the field of multiple hypothesis testing, and control the expected proportion of false-positives. (Abramovich and Benjamini 1996) use this methodology to control the expected number of coefficients which are not thresholded but should have been.

BayesThresh (Abramovich et al. 1998) uses a Bayesian hierarchical model, assuming independent $N(0, \sigma^2)$ noise. They use a mixture of a point mass at 0 and a $N(0, \tau^2)$ density as their prior on the population wavelet coefficients. The marginal posterior median of the population wavelet coefficient is then used as their estimate of it. This gives a thresholding rule, since the point mass at 0 in the prior gives non-zero probability that the population wavelet coefficient will be zero. We now discuss an extension of this method.

6.2 An Extension of Bayesian Wavelet Thresholding

We describe a novel thresholding procedure which uses the discretized area-interaction process introduced in Chapter 5 to model the correlation between neighbouring coefficients in the wavelet transform.

The principle behind our method is to model the discrete wavelet transform as a

marked lattice process. The 'lattice' is the natural binary tree which is commonly used to represent the coefficients. A discretized area-interaction process is used as a prior on the distribution of non-zero coefficients. We also make use of the extra information gained by allowing multiple points to exist at a single location, using the number of points as a shrinkage factor. This is different from Abramovich et al. (1998), where the implicit assumption was that the configuration was Binomial (i.e. a totally random configuration of non-zero coefficients). The reason for thinking that the discretized area-interaction process would make a better prior is that the wavelet transform provides time-frequency localisation. This means that the effect of, for example, a discontinuity in the signal or in one of the first few derivatives of the signal will produce significant coefficients of the wavelet transform of the signal only in the coefficients close to the location at which the discontinuity occurs. This fact means that the wavelet transform will have all its coefficients clustered around a few locations, thus leading to a clustered rather than uniformly random distribution of coefficients. This can be seen clearly in Figure 6.1 on page 119, which shows the discrete wavelet transform of several common test functions represented in the natural binary tree configuration.

More formally, we begin by allowing for the presence of noise by assuming that the true wavelet coefficients are corrupted by Gaussian noise with zero mean and some variance σ^2. This gives the following likelihood:

$$\widehat{d}_{jk}|d_{jk} \sim N(d_{jk}, \sigma^2),$$

where $\widehat{d}_{jk}$ is the value of the noisy wavelet coefficient (the "data") and d_{jk} is the value of the "true" coefficient. We then place a prior on the value of the wavelet coefficients:

$$d_{jk}|\mathbf{J} \sim N(0, \tau^2 J_{jk}), \tag{6.3}$$

where τ^2 is a constant and J_{jk} is the number of points at location (j,k) of a certain lattice process J which exists on the natural binary tree commonly used to represent wavelet coefficients. Thus the more points at a given location, the larger

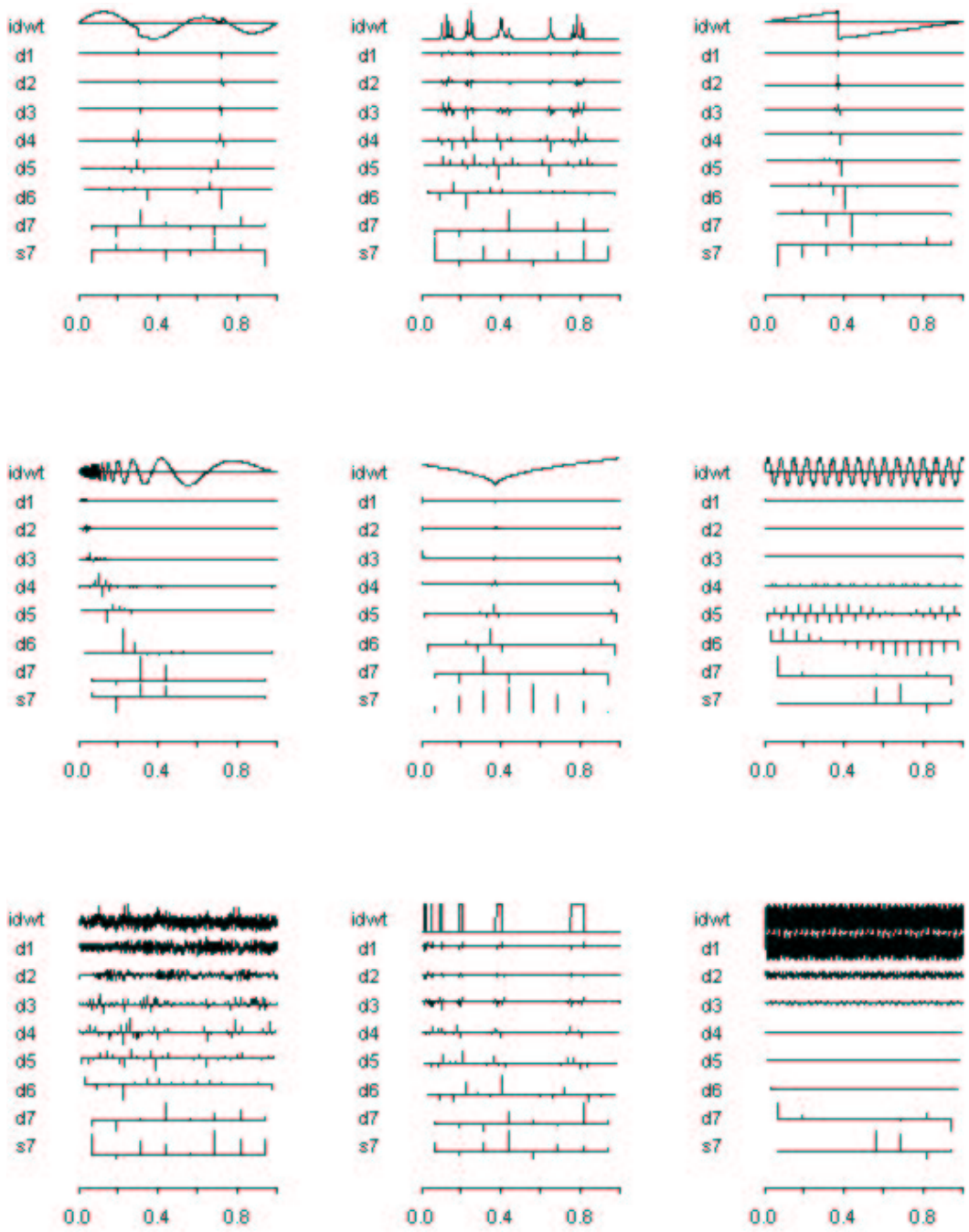

Figure 6.1: Examples of the discrete wavelet transform of some test functions. There is clear evidence of clustering in most of the graphs. The original functions are shown above their discrete wavelet transform each time.

the variance of the prior on d_{jk}, resulting in a higher probability of large values of d_{jk}. Finally, we place a hyperprior on this lattice process:

$$P(\mathbf{J}) = \alpha\lambda^{N(\mathbf{J})}\gamma^{-m(U(\mathbf{J}))} \tag{6.4}$$

with respect to the unit rate discrete Poisson process introduced in Section 5.2, where $\mathbf{J} = (J_{jk})$ is the configuration. If we take a value of γ greater than one this gives a clustered configuration. Thus we would expect to see clusters of large values of d_{jk} if this were a reasonable model — which is exactly what we do see in Figure 6.1.

This is similar to Abramovich et al. (1998), who assume that the true wavelet coefficients are distributed as a mixture of a Normal distribution with zero mean and variance dependent on the level of the coefficient, and a point mass at zero as follows:

$$d_{jk} \sim \zeta_{jk} N(0, \tau_j^2) + (1 - \zeta_{jk})\delta(0),$$

where d_{jk} is the value of the kth coefficient at level j of the discrete wavelet transform and τ_j is a positive constant. Notice that (6.3) includes a point mass at zero when $J_{jk} = 0$ (i.e. when there are no points alive at that location). Abramovich et al. (1998) also assume that there is $N(0, \sigma^2)$ noise added to the true coefficients. This is equivalent to our likelihood $\widehat{d}_{jk}|d_{jk} \sim N(d_{jk}, \sigma^2)$.

Clearly a suitable interpretation of $U(\mathbf{J}) = \bigcup_{(j,k)} Z(J_{jk})$ in equation (6.4) is required. Several possibilities spring to mind. Organising the wavelet coefficients into the traditional binary tree layout, one possibility would be to use the parent, children and immediate sibling and cousin of a coefficient as $Z(x)$. Another would be to use a variation on this taking into account the length of support of the wavelet used. Figure 6.2 shows the scheme used in the program which we used to implement the model (described in Section 7.1). There, we decided to use the parent, the coefficient on the parent's level of the transform which is next-nearest to x, the two adjacent coefficients on the level of x, the two children and the coefficients adjacent to them, making a total of nine coefficients (including x itself).

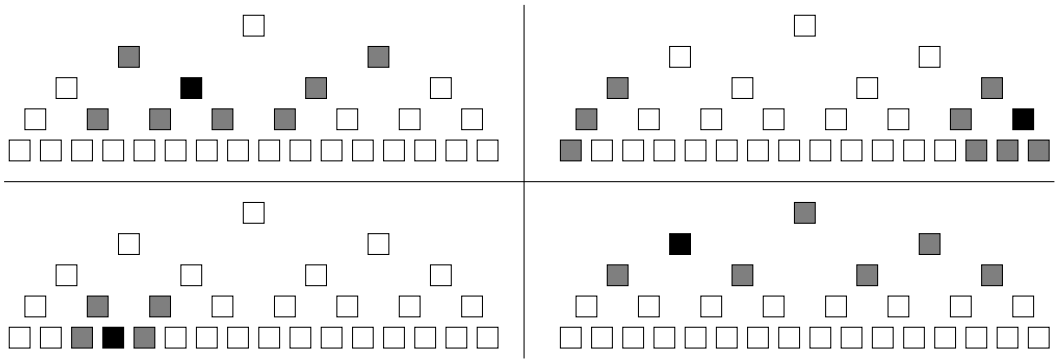

Figure 6.2: The four plots give examples of what we used as $Z(\cdot)$ for four different example locations showing how we dealt with boundaries. Grey boxes are $Z(x) \setminus \{x\}$ for each example location x, while x itself is shown as black.

Figure 6.2 also shows how we dealt with boundaries: we assumed that the signal we are examining is periodic, making it natural to have periodic boundary conditions horizontally. If $Z(x)$ overlaps with a vertical boundary we simply discard those parts which have no locations associated with them. The simple counting measure used has $m(K(x)) = 9$ unless x is in the bottom or one of the top two rows.

As usual in the Bayesian setting, we wish to know the posterior probability of $\mathbf{J}$ given a set of data $\widehat{\mathbf{d}} = \{\widehat{d}_{jk}\}$:

$$f(\mathbf{J}|\widehat{\mathbf{d}}) = k \times f(\mathbf{J}) f(\widehat{\mathbf{d}}|\mathbf{J})$$

where k is a constant of proportionality.

Switching to log-likelihoods for ease of notation this becomes

$$\log f(\mathbf{J}|\widehat{\mathbf{d}}) = \log \alpha + N(\mathbf{J}) \log \lambda - m(U(\mathbf{J})) \log \gamma + \log f(\widehat{\mathbf{d}}|\mathbf{J}),$$

where we have absorbed the constant of proportionality into $\log \alpha$. We now examine $f(\widehat{\mathbf{d}}|\mathbf{J})$.

Let $f_i(x)$ be the normal density with zero mean and variance $i\tau^2$. Then

$$f(\widehat{\mathbf{d}}|\mathbf{J}) = \prod_{i=0}^{\infty} \prod_{J_{jk}=i} f_i(\widehat{d}_{jk}).$$

Hence the log posterior probability is

$$\log f(\mathbf{J}|\widehat{\mathbf{d}}) = \log\alpha + N(\mathbf{J})\log\lambda - m(U(\mathbf{J}))\log\gamma + \sum_{i=0}^{\infty}\sum_{J_{jk}=i}\log f_i(\widehat{d}_{jk}). \qquad (6.5)$$

Clearly this is not an ordinary area-interaction process and we must once more extend our simulation techniques to deal with this fact.

6.3 Sampling from the posterior

Although the expression in equation (6.5) may look like a rather complicated density it turns out that it can be simulated perfectly using a rather simple extension of the procedure for simulating the discretized area-interaction process covered in Chapter 5.

As usual, we use a birth-death process with unit death rate. In this case the detailed balance equation (4.4) on page 57 becomes

$$q(\mathbf{J},\mathbf{K}) = \frac{\pi(\mathbf{K})}{\pi(\mathbf{J})},$$

where $\mathbf{J}$ is a configuration identical to $\mathbf{K}$ but with one less point. Let x denote that point. Thus from equation (6.5) we see that

$$\begin{aligned} q(\mathbf{J},\mathbf{K}) &= \frac{\alpha\lambda^{N(\mathbf{K})}\gamma^{-m(U(\mathbf{K}))}\prod_{i=0}^{\infty}\prod_{K_{jk}=i}f_i(\widehat{d}_{jk})}{\alpha\lambda^{N(\mathbf{J})}\gamma^{-m(U(\mathbf{J}))}\prod_{i=0}^{\infty}\prod_{J_{jk}=i}f_i(\widehat{d}_{jk})} \\ &= \lambda\gamma^{-m(K(x)\setminus U(\mathbf{J}))}\frac{f_{K_x}(\widehat{d}_x)}{f_{J_x}(\widehat{d}_x)} \\ &= \lambda\gamma^{-m(K(x)\setminus U(\mathbf{J}))} \\ &\quad\times\sqrt{\frac{\tau^2J_x+\sigma^2}{\tau^2(J_x+1)+\sigma^2}}\exp\left(\frac{\widehat{d}_x^2\tau^2}{2(\tau^2J_x+\sigma^2)(\tau^2(J_x+1)+\sigma^2)}\right). \end{aligned}$$

Since the dominating process is Poisson we can use the method introduced in Section 5.2 to simulate the dominating process. Since we generate independent processes at each location this enables us to use different birth rates at each location. A suitable birth rate is then clearly

$$\lambda_{jk}^{dom} = \lambda e^{\widehat{d}_{jk}^2\tau^2/2\sigma^2(\tau^2+\sigma^2)} \qquad (6.6)$$

at each location (j,k) on the grid. The probability of accepting a birth from this process is then

$$\begin{aligned} p(x) &= \gamma^{-m(K(x)\setminus U(\mathbf{J}))}\sqrt{\frac{\tau^2 J_x+\sigma^2}{\tau^2(J_x+1)+\sigma^2}} \\ &\times \exp\left(\frac{\widehat{d}_x^2\tau^2}{2(\tau^2 J_x+\sigma^2)(\tau^2(J_x+1)+\sigma^2)}-\frac{\widehat{d}_x^2\tau^2}{2\sigma^2(\tau^2+\sigma^2)}\right), \end{aligned}$$

which after some algebra becomes

$$\begin{aligned} p(x) &= \gamma^{-m(K(x)\setminus U(\mathbf{J}))}\sqrt{\frac{\tau^2 J_x+\sigma^2}{\tau^2(J_x+1)+\sigma^2}} \\ &\times \exp\left(-\frac{\widehat{d}_x^2\tau^2}{2}\frac{\tau^2 J_x(\tau^2(J_x+1)+2\sigma^2)}{\sigma^2(\tau^2+\sigma^2)(\tau^2 J_x+\sigma^2)(\tau^2(J_x+1)+\sigma^2)}\right). \end{aligned} \quad (6.7)$$

We recall from the work of Sections 5.1 to 5.4 that there are two important things to consider when using dominated coupling from the past in this way:

1. Whether the birth rates of the candidate minimum and the candidate maximum processes are independent of the state of the dominating process.

2. Whether stochastic monotonicity holds.

Since we are using a discrete Poisson process as the dominating process we clearly need not worry about point 1. The only problem, then, is point 2. Fortunately we may again use the methods developed in Section 4.4.1, since (6.7) is a product of monotonic functions. This enables us to restore stochastic monotonicity by using different acceptance rules for the candidate maximum and candidate minimum processes as we did for the attractive-repulsive process in Section 4.4.1. The required probabilities are then

$$\begin{aligned} P(x) &\leq \gamma^{-m(K(x)\setminus U(\mathbf{J}^{max}))}\sqrt{\frac{\tau^2 J_x^{max}+\sigma^2}{\tau^2(J_x^{max}+1)+\sigma^2}} \\ &\times \exp\left(-\frac{\widehat{d}_x^2\tau^2}{2}\frac{\tau^2 J_x^{min}(\tau^2(J_x^{min}+1)+2\sigma^2)}{\sigma^2(\tau^2+\sigma^2)(\tau^2 J_x^{min}+\sigma^2)(\tau^2(J_x^{min}+1)+\sigma^2)}\right) \end{aligned}$$

for the candidate maximum process and

$$\begin{aligned} P(x) &\leq \gamma^{-m(K(x)\setminus U(\mathbf{J}^{max}))}\sqrt{\frac{\tau^2 J_x^{min}+\sigma^2}{\tau^2(J_x^{min}+1)+\sigma^2}} \\ &\times \exp\left(-\frac{\widehat{d}_x^2\tau^2}{2}\frac{\tau^2 J_x^{max}(\tau^2(J_x^{max}+1)+2\sigma^2)}{\sigma^2(\tau^2+\sigma^2)(\tau^2 J_x^{max}+\sigma^2)(\tau^2(J_x^{max}+1)+\sigma^2)}\right) \end{aligned}$$

for the candidate minimum process. The remainder of the algorithm carries over in the obvious way, with the initial configuration of the minimum process being composed of those points whose marks are less than

$$P(x) = \gamma^{-M(\chi)}\sqrt{\frac{\sigma^2}{\tau^2+\sigma^2}} \times \exp\left(-\frac{\widehat{d}_x^2\tau^2}{2\sigma^2(\tau^2+\sigma^2)}\right)$$

where, as in Section 5.3, $M(\chi) = \max_\chi(m(K(x)))$. Since we have taken care to observe stochastic monotonicity and independence from the state of the dominating process the candidate maximum and candidate minimum processes truly are maximum and minimum processes and thus the dominated coupling from the past algorithm goes through without a hitch.

6.4 Using the Generated Samples

Having simulated realisations of $\mathbf{J}|\widehat{\mathbf{d}}$ we must then simulate $\mathbf{d}|\mathbf{J},\widehat{\mathbf{d}}$ for each realisation of $\mathbf{J}$ generated in the first step. Taking the sample median of this distribution gives an estimate for $\mathbf{d}$, as required. The median is used instead of the mean as this gives a thresholding rule, as discussed in Abramovich et al. (1998).

We calculate $f(\mathbf{d}|\mathbf{J},\widehat{\mathbf{d}})$ using logarithms for ease of notation. Assuming that

$J_{jk} \neq 0$ we find

$$\begin{aligned}
\log f(d_{jk}|\widehat{d}_{jk}, J_{jk} \neq 0) &= \log f(d_{jk}) + \log f(\widehat{d}_{jk}|d_{jk}, (j,k) \in \mathbf{J}) + C \\
&= \frac{-d_{jk}^2}{2\tau^2 J_{jk}} + \frac{-(\widehat{d}_{jk} - d_{jk})^2}{2\sigma^2} + C_1 \\
&= \frac{-d_{jk}^2\sigma^2 - (\widehat{d}_{jk} - d_{jk})^2\tau^2 J_{jk}}{2\sigma^2\tau^2 J_{jk}} + C_1 \\
&= -\frac{d_{jk}^2(\sigma^2 + \tau^2 J_{jk}) - 2\tau^2 J_{jk}\widehat{d}_{jk} d_{jk} - \tau^2 J_{jk}\widehat{d}_{jk}}{2\sigma^2\tau^2 J_{jk}} + C_1 \\
&= -\frac{(\sigma^2 + \tau^2 J_{jk})\left(d_{jk} - \frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2 + \tau^2 J_{jk}}\right)^2}{2\sigma^2\tau^2 J_{jk}} + C_2
\end{aligned}$$

where C, C_1 and C_2 are constants. Thus

$$f(d_{jk}|\widehat{d}_{jk}, J_{jk} \neq 0) \sim N\left(\frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2 + \tau^2 J_{jk}}, \frac{\sigma^2\tau^2 J_{jk}}{\sigma^2 + \tau^2 J_{jk}}\right).$$

When $J_{jk} = 0$ we clearly have $f(d_{jk}|J_{jk}, \widehat{d}_{jk}) = 0$.

6.5 Examples

In this section we present some results from using the program detailed in Chapter 7 to de-noise two standard test signals commonly called "jumpsine" and "heavisine". These are both sine waves with discontinuities at two points, resulting in a piece of the sine curve being either above or below the level of the rest of the curve. The test functions were generated using the statistics package S-Plus Wavelets.

6.5.1 Jumpsine

Figure 6.3 shows the jumpsine data set. The top picture is of the dataset with no noise at all. In the middle picture, the dataset has been corrupted with white noise with $\sigma = 0.1\sigma_0$, where σ_0 is the standard deviation of the signal. The bottom picture shows the result of processing the noisy signal shown in the middle picture using the program detailed in Chapter 7. Figure 6.4 shows the discrete wavelet transform of the same three signals.

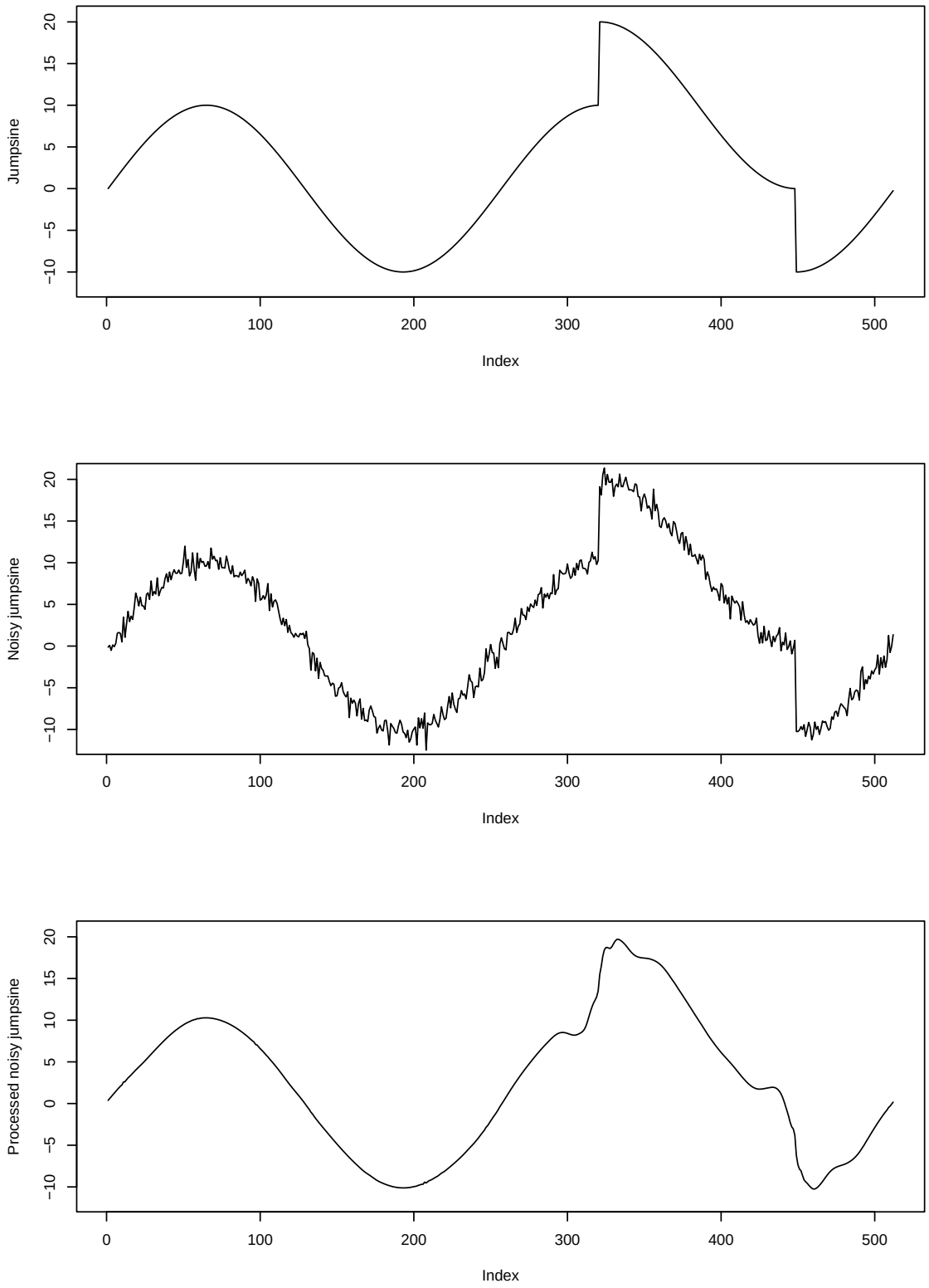

Figure 6.3: From top to bottom, the jumpsine dataset with no noise, white noise with $\sigma = 0.1\sigma_0$ and having been processed using our model.

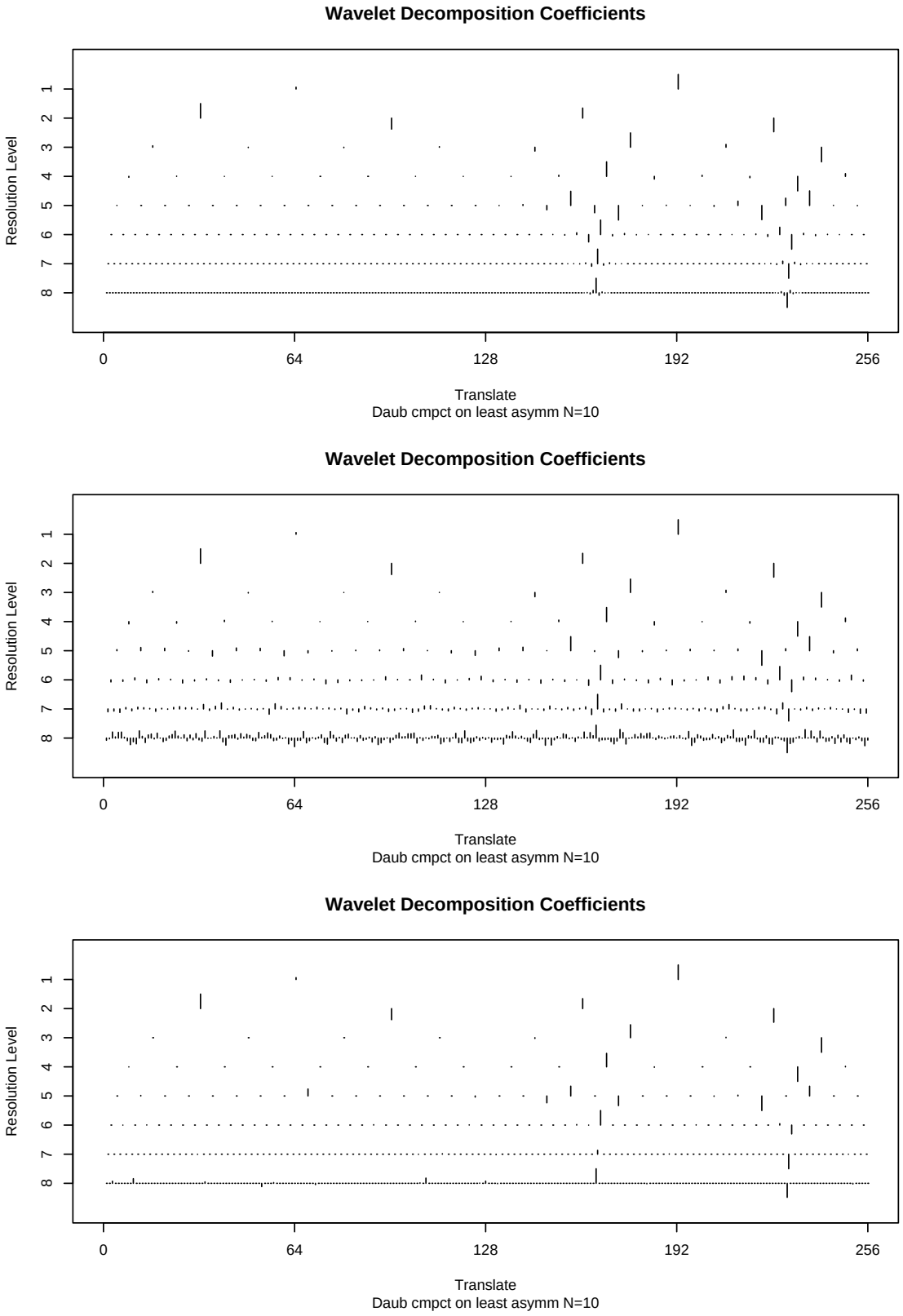

Figure 6.4: From top to bottom, the discrete wavelet transform of the jumpsine dataset with no noise, white noise with $\sigma = 0.1\sigma_0$ and having been processed using our model. Daubechies least asymmetric wavelets with n=10 were used.

The program described in chapter 7 was used to generate 1000 realisations of the posterior. These simulations were then used as discussed in Section 6.4. The parameters γ and λ were set to 3.0 and 1.1 respectively and τ was set to 0.05. The value of σ was set to the standard deviation of the noise that was added. As can be seen the de-noising process was fairly effective, but the discontinuities have not been dealt with as well as perhaps we could have hoped. I suspect that this problem may be due to the approximations it was necessary to make when dealing with the problems discussed in Section 7.1.5.

6.5.2 Heavisine

Figure 6.5 shows the heavisine data set in a similar configuration to that used for the jumpsine dataset in Figure 6.3. The top picture is of the dataset with no noise at all. In the middle picture, the dataset has been corrupted with white noise with $\sigma = 0.1\sigma_0$, where σ_0 is the standard deviation of the signal. The bottom picture shows the result of processing the noisy signal shown in the middle picture using the program detailed in Chapter 7. Figure 6.6 shows the discrete wavelet transform of the same three signals.

As in the case of the jumpsine dataset, the program described in chapter 7 was used to generate 1000 realisations of the posterior. These simulations were then used as discussed in Section 6.4. The parameters γ and λ were set to 3.0 and 0.7 respectively and τ was set to 0.15. σ was again set to the standard deviation of the noise that was added. As can be seen the de-noising process was fairly effective, with only a few 'mistakes'. I feel that these were due mainly to the problems discussed in Section 7.1.5, as it can be seen by comparing Figures 6.5 and 6.6 that the wavelet coefficients corresponding to these parts of the signal were unusually large, and so were probably included in the 'always on' list.

Figure 6.7 compares the results gained by reducing the clustering parameter γ to 2.0. Clearly the results gained by using a larger value are superior, giving good evidence for the merits of the added layer of complexity which our model uses over

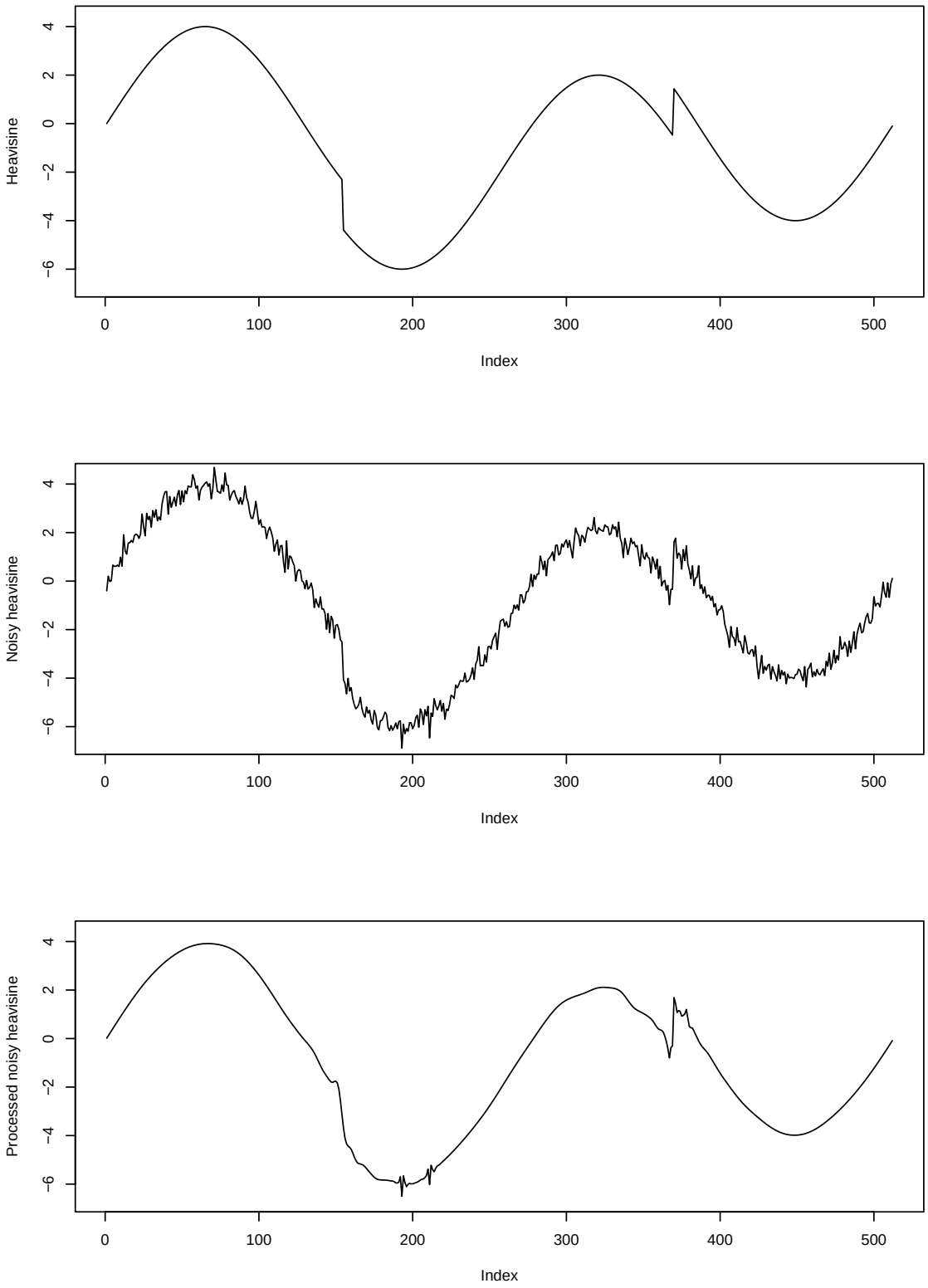

Figure 6.5: From top to bottom, the heavisine dataset with no noise, white noise with $\sigma = 0.1\sigma_0$ and having been processed using our model.

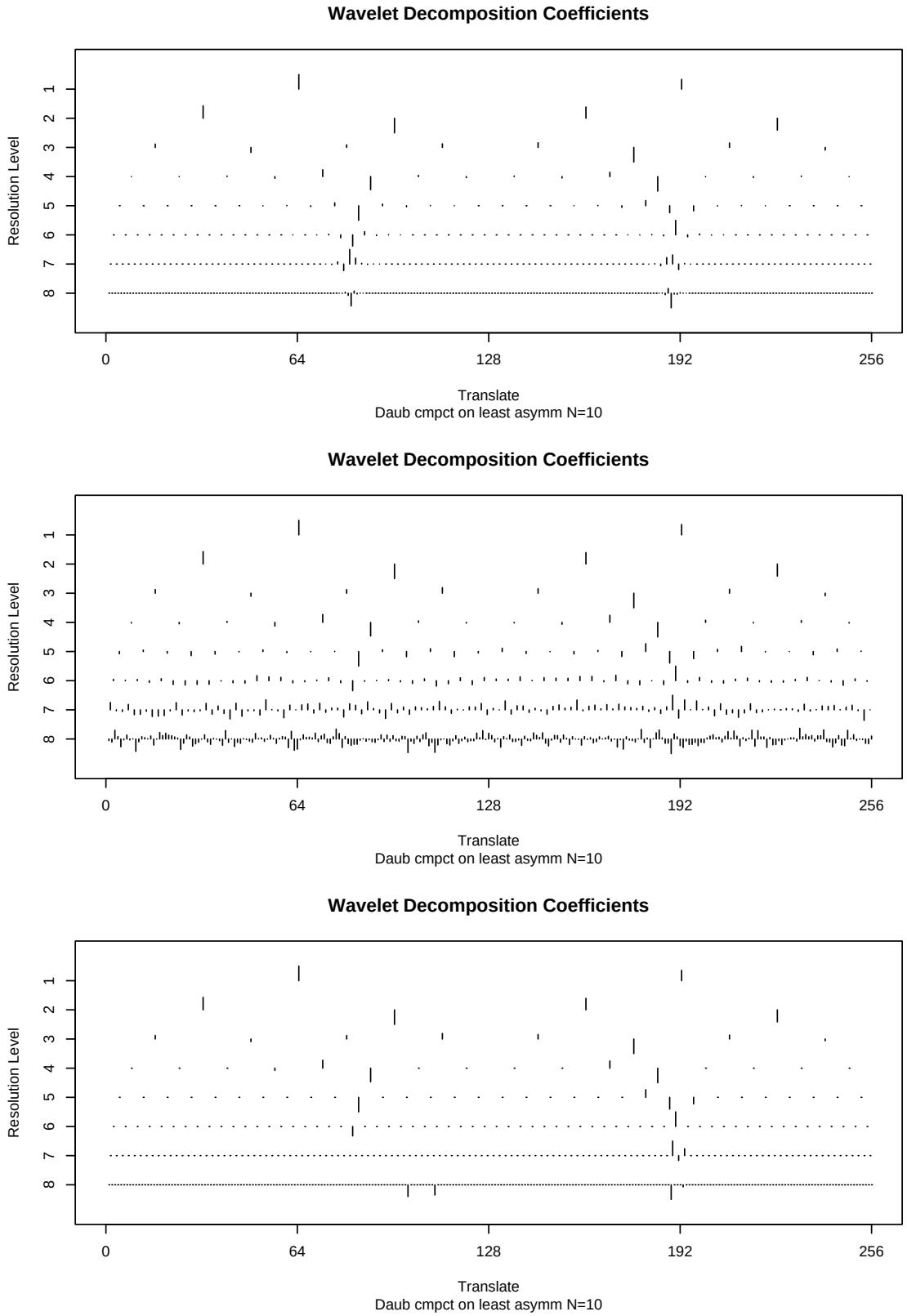

Figure 6.6: From top to bottom, the discrete wavelet transform of the heavisine dataset with no noise, white noise with $\sigma = 0.1\sigma_0$ and having been processed using our model. Daubechies least asymmetric wavelets with n=10 were used.

traditional Bayesian wavelet thresholding.

6.6 Simulation Study

We now present a more careful simulation study of the performance of our estimator relative to several established Wavelet-based estimators. Similar to the study of Abramovich et al. (1998), we investigate the performance of our method on the four standard test functions of Donoho and Johnstone (1994, 1995), namely "Blocks", "Bumps", "Doppler" and "Heavisine". These test functions are used because they exhibit different kinds behaviour typical of signals arising in a variety of applications.

The test functions were simulated at 256 points equally spaced on the unit interval. The test signals were centred and scaled so as to have mean value 0 and standard deviation 1. We then added independent $N(0, \sigma^2)$ noise to each of the functions, where σ was taken as 1/10, 1/7 and 1/3. The noise levels then correspond to root signal-to-noise ratios (RSNR) of 10, 7 and 3 respectively. We performed 25 replications. For our method, we simulated 25 independent draws from the posterior distribution of the d_{jk}'s and used the sample median as our estimate, as this gives a thresholding rule. For each of the runs, σ was set to the standard deviation of the noise we added, τ was set to 1.0, λ was set to 0.05 and γ was set to 3.0.

The values of parameters σ and τ were set to the true values of the standard deviation of the noise and the signal, respectively. In practice it would be necessary to develop some method for estimating these values. The value of λ was chosen to be 0.05 because it was felt that not many of the coefficients would be significant. The value of γ was chosen based on the small trials for the heavisine and jumpsine datasets presented in Section 6.5.

We compare our method with several established wavelet-based estimators for reconstructing noisy signals: ordinary BayesThresh (Abramovich et al. 1998),

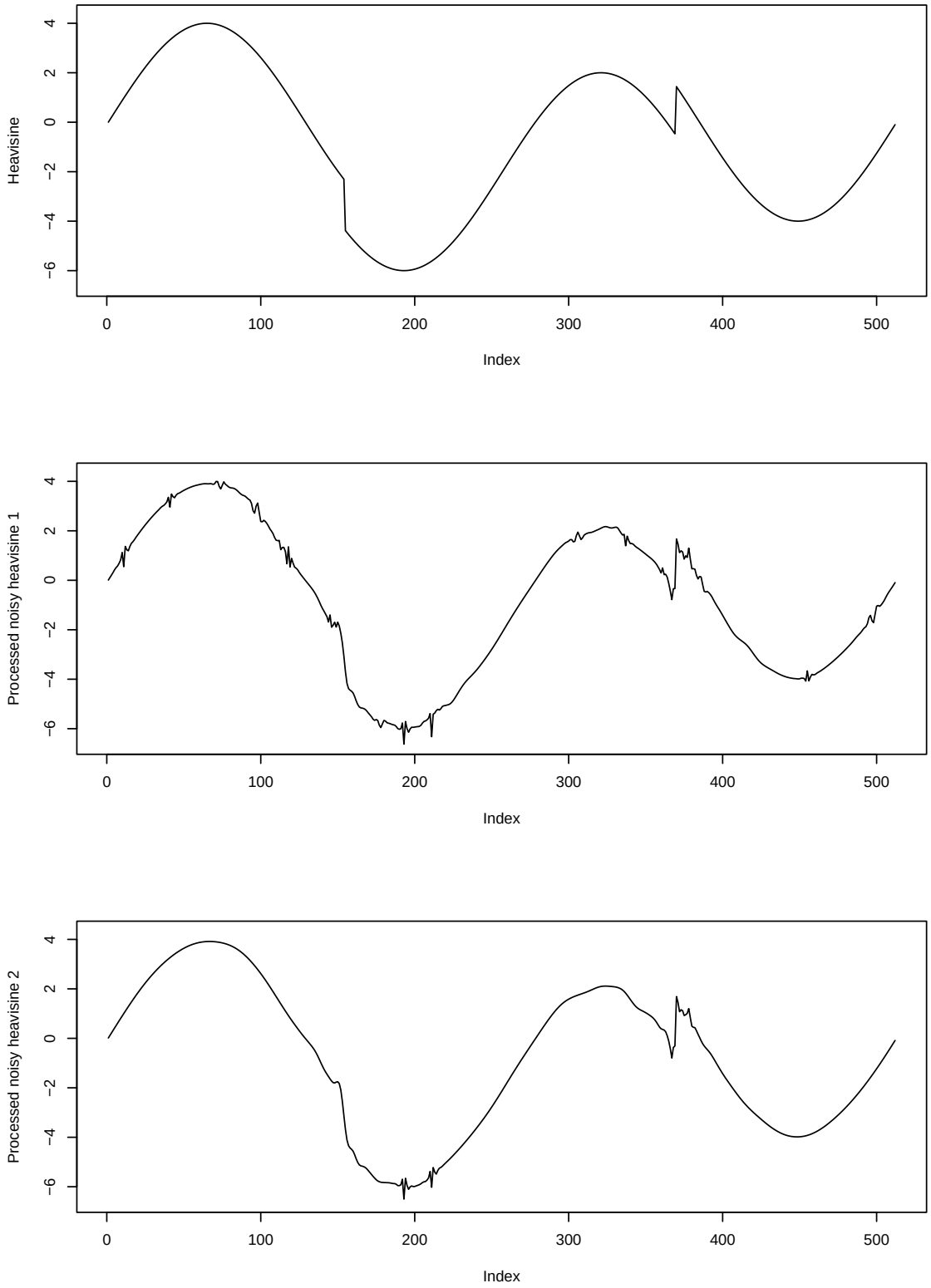

Figure 6.7: The heavisine dataset. At the top we have the original signal, in the middle we have performed our simulations with $\gamma = 2.0$ and at the bottom, with $\gamma = 3.0$. Clearly the latter is a superior result.

Method	RSNR	AMSEs for the following test functions:			
		Blocks	Bumps	Doppler	Heavisine
LatticeBayesThresh	10	0.0025	0.0084	0.0049	0.0032
	7	0.0056	0.0185	0.0087	0.0052
	3	0.0534	0.1023	0.0448	0.0149
BayesThresh	10	0.0344	0.1651	0.0167	0.0035
	7	0.0414	0.1716	0.0225	0.0057
	3	0.0860	0.2015	0.0448	0.0140
Cross-validation	10	0.0055	0.0392	0.0112	0.0030
	7	0.0096	0.0441	0.0135	0.0054
	3	0.0452	0.0914	0.0375	0.0057
SureShrink	10	0.0049	0.0131	0.0054	0.0065
	7	0.0098	0.0253	0.0099	0.0093
	3	0.0482	0.0973	0.0399	0.0147
False discovery rate	10	0.0159	0.0449	0.0144	0.0064
	7	0.0294	0.0758	0.0253	0.0093
	3	0.1230	0.2324	0.0861	0.0148

Table 6.1: Average mean-square errors for our estimator (labelled LatticeBayesThresh), ordinary BayesThresh, cross-validation, SureShrink and false discovery rate estimators for four test functions for two values of the root signal-to-noise ratio. Averages are based on 25 replicates.

SureShrink (Donoho and Johnstone 1994), cross-validation (Nason 1996) and the false discovery rate (Abramovich and Benjamini 1996). For test signals "Bumps", "Doppler" and "Heavisine" we used Daubechies least asymmetric wavelet of order 10 (Daubechies 1992). For "Blocks" we used the Haar wavelet, as the original signal was piecewise constant. The analysis was carried out using the *R* statistical package. Guy Nason's WaveThresh package was used to perform the discrete wavelet transform and also to compute the BayesThresh, SureShrink, cross-validation and false discovery rate estimators.

The goodness of fit of each estimator was measured by its average mean-square error (AMSE) over the 25 replications. Table 6.1 presents the results. It is clear that our estimator struggles when there is a small signal-to-noise ratio but performs extremely well with respect to the other estimators when the signal-to-noise ratio is larger. I feel that the results would have been even better if it had not been

necessary to make the approximations which are described in Section 7.1.5 of Chapter 7.

6.7 Future Work

We have introduced a procedure for Bayesian wavelet thresholding which uses the naturally clustered nature of the wavelet transform when deciding how much weight to give coefficient values. We have demonstrated at least in principle that this procedure works, though the implementation suffered from some problems which made exact computation infeasible. The performance seems good for moderate and low noise levels, though it was a little disappointing for higher noise levels.

One possible area for future work would be to replace equation (6.3) with

$$d_{jk}|\mathbf{J} \sim N(0, \tau^2 (J_{jk})^z),$$

where z would be a further parameter. This would modify the number of points which are likely to be alive at any given location and thus also modify the tail behaviour of the prior. The idea behind this suggestion is that when we know that the behaviour of the data is either heavy or light tailed, we could adjust z to compensate. This could possibly also help speed up convergence by reducing the number of points at locations with large values of d_{jk}. As inclusion of this extra parameter requires only minor modifications, the implementation discussed in Section 7.1 actually includes this option. The results presented in Sections 6.5 and 6.6 were generated by simply setting $z = 1$.

A second possible area for future work would be to develop some automatic methods for choosing the parameter values, perhaps using the method of maximum pseudo-likelihood introduced for the spatial case in Section 4.3.2, which was in fact originally developed for lattice models.

Chapter 7

Implementational issues

The simulations outlined in Sections 6.3 and 4.4.1 were carried out in the C programming language under Linux on a dual Pentium III 1000MHz computer. The reason C was chosen rather than a statistical language such as S was due to the computational effort involved in these simulations. The algorithm described in Section 6.3 was initially written in S for a simple case, but even in this simple case it took around 30 minutes to generate a single draw. Having now implemented the general case in C it now takes around 30 minutes to generate around 500 independent draws for a process with 1023 locations.[1]

Several important issues arose during implementation. In this chapter we discuss those issues. The program itself is discussed more fully in appendix A.

In some ways, the implementation of the algorithm given in Section 6.3 is simpler than the implementation of the algorithm given in Section 4.4.1. For this reason we begin by discussing those issues which arose while implementing the former. We then move on to the implementation of the algorithm given in Section 4.4.1. Due to the similarities between the underlying algorithms for these two models, many of the issues which arose while implementing the first algorithm came up again while implementing the second. Thus the material in Section 7.2 relies heavily on the material of Section 7.1, and should not be read without it.

[1] The term 'location' in its present context is defined in Section 5.1. Throughout Section 7.1 we use the notation introduced in Sections 5.1 and 5.4 on pages 92 and 96 respectively. Throughout Section 7.2 we use the notation introduced in Section 4.1 on page 50.

7.1 Bayesian Wavelet Thresholding

We begin by discussing those issues concerning the tracking of points in the dominating process and related to that, the use of suitable sequences of random numbers. We then discuss some approximations which were made to ensure that the program converged in a reasonable amount of time.

7.1.1 Random Number Generators

During implementation three separate pseudo random number generators were used to ensure that the simulations were correct.[2] These three sources were:

1. Seeds for individual independent draws.

2. Seeds for each location in the process.

3. Birth and death times and marks.

All of the generators used in the implementation are *linear congruential pseudo random number generators.* These generate a sequence of numbers $X_1, \ldots, X_n$ using the recurrence relation

$$X_{m+1} = (AX_m + C) \mod p. \tag{7.1}$$

To generate sequence of $U[0,1]$ random numbers, $U_1, \ldots, U_n$, we take

$$U_m = \frac{X_m}{p}.$$

For a complete review of these and many other types of pseudo random number generators see Knuth (1998a). Briefly, there are various important criteria for a good linear congruential pseudo random number generator which help to determine what the values of A, C and p should be, including the fact that A and p should be coprime and the rather obvious requirement that p be fairly large.

[2] It was not strictly necessary to use three separate random number generators, but this was the method of ensuring correctness which seemed most obvious to the author. Other solutions to the same problems also exist.

The most important fact about these pseudo random number generators from our point of view is that given a starting value (*seed*) X_0 the sequence generated by a particular choice of A, C and p is *entirely deterministic*. This means that as long as we store the value of the seed, we can generate the exact same sequence again if we need to. As we will see later on, this feature is crucial to the implementation of our algorithm.

We now return to discussion of the specific pseudo random number generators used in our program and the reasons they were necessary.

7.1.2 Seeds for different draws

Each time the program is run it requires several inputs. These are:

1. The data.
2. The parameter values.
3. Two seeds.

Here we discuss the third.

The reason we must have a separate pseudo random number generator for this is as follows: If we were to use one of the generators which uses the seeds passed to the program to generate the seeds themselves then (for example) the third random number generated in the first implementation of the program would be exactly the same as the first generated in the second implementation which would be exactly the same as the first seed passed to the third implementation. In general, the n^{th} number generated in the first implementation would be the same as the $(n - 2(m-1))^{\text{th}}$ generated in the m^{th} implementation. This obviously violates the independence of the draws. Thus when we are looking for a series of independent draws we must use a different random number generator to generate seeds for these draws than the one which uses these seeds within the program.

The linear congruential pseudo random number generator used here has parameters $A = 48271$, $C = 0$ and $p = 2147483647$ $(= 2^{31} - 1)$.

7.1.3 Seeds for each location

From the example in Section 3.2.2 we recall that if coalescence has not occurred at time 0 then we double the value of $-M$ and try again *keeping the transitions already generated*. It is easy to see that if we did not do this we would be introducing bias into our sample. In order to do this in our simulation we have two options. The first is to actually store the random numbers generated at each stage. This would require a large amount of memory and the program would be continually reading in from and writing out to memory (which is a very time consuming process). The other alternative is to simply store the values of the seeds used to generate these random numbers and generate the entire sequence again whenever necessary. This is also a very time consuming process and would require great care to ensure each time that things were generated in the right order so as to maintain the integrity of the whole thing.

In an attempt to lessen the overheads from both of these options a compromise was implemented. Using a second pseudo random number generator two seeds were generated for each location — one which was used to generate the marks when points were born (which we shall refer to as the markseeds) and one to generate the sequence of birth/death times (which we shall refer to as the timeseeds). These seeds were stored so as to reduce the overheads both from storing the entire sequence of births, deaths and marks *and* to reduce the overheads from having to generate the *entire* sequence of random numbers each time we want to find out what happens next. In addition to these the death times of each point currently alive (in the maximum or minimum processes) were stored, as was the birthtime, deathtime and mark of the next point to be born (in the dominating process — it may not actually be born in either the maximum or minimum process) at each location. This data means that each time a point dies we need not re-generate any events. Each time a point is born we need only re-generate the sequence of events at that single location to find the birthtime, deathtime and mark of the next point

to be born (in the dominating process) at that location.

The reason a second pseudo random number generator was used is similar to the reasons given in the previous section. If we used the same one as was used for the events (see Section 7.1.4) then the random number generated for the first event of the first location would be the same as that generated for the seed of the second location and so on. Reasons for not using the same generator as was used for the seeds input to the program are given in the previous section. The seeds used by this generator were those passed to the program (again, see previous section). The first was used to generate the markseeds and the second was used to generate the timeseeds.

The linear congruential pseudo random number generator used here has parameters $A = 1664525$, $C = 1013904223$ and $p = 4294967296$ $(= 2^{32})$. This is a faster pseudo random number generator than that used for different draws, as the modulus used is the length of an unsigned long integer. Thus we need only do `x=a*x+c` and the modulus takes care of itself.

7.1.4 Birth and death times and marks

From the timeseed of each location a third random number generator is used to determine the birth and death times for each location. It is also used to generate a mark from a location's markseed each time that a point is born. These are the marks discussed in Section 4.1.4.

The linear congruential pseudo random number generator used here has parameters $A = 2650845021$, $C = 0$ and $p = 4294967296$ $(= 2^{32})$ and is even faster than the last one, as we do not need to add a constant after multiplication. The value of A was found in an exhaustive search for the best multiplier to use with the modulus 2^{32} by L.C. Killingbeck[3].

The first and third pseudo random number generators above were taken from

[3] Cited in errata to Knuth (1998a) available from `http://www-cs-faculty.stanford.edu/~knuth/taocp.html`

those recommended in Knuth (1998a). The second was taken from Press et al. (1993).

7.1.5 Dealing with large and small rates

The second problem we encountered when attempting to implement the algorithm in Section 6.3 was that of extremely high birth rates. Recall from Equation 6.6 that if the maximum data value d_{jk} is twenty times larger in magnitude than the standard deviation of the noise (a not uncommon event for reasonable noise levels) then we have

$$\begin{aligned} \lambda_{dom} &= \lambda e^{400\sigma^2\tau^2/2\sigma^2(\tau^2+\sigma^2)} \\ &= \lambda e^{200\tau^2/(\tau^2+\sigma^2)}. \end{aligned}$$

Now unless τ is significantly smaller than σ, this will result in enormous birth rates. We are clearly not going to be able to simulate this efficiently.

To get around this problem we reasoned that the chances of there being no live points at a location whose data value is large (resulting in a value of λ_{dom} larger than e^4) is sufficiently small that for the purposes of calculating $m((x \oplus G) \setminus (Y(-M,u) \oplus G))$ for nearby locations it could be assumed that the number of points alive was strictly positive. This allows us to simulate the process accurately for the locations of interest and provide a reasonable level of discrimination in a more reasonable time frame.

Unfortunately, the problems do not stop there. Recall from Section 6.4 that

$$d_{jk}|J_{jk}, \widehat{d}_{jk} \sim N\left(\frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2+\tau^2 J_{jk}}, \frac{\sigma^2\tau^2 J_{jk}}{\sigma^2+\tau^2 J_{jk}}\right)$$

so that we need values of J_{jk} for each location (j,k) in the configuration. Unfortunately, we no longer know the value of J_{jk} for those locations which have large values of d_{jk}.

To get around this problem we first notice that

$$\frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2+\tau^2 J_{jk}} \xrightarrow[J_{jk}\to\infty]{} \widehat{d}_{jk}$$

monotonically from below, and that

$$\frac{\tau^2 J_{jk}\sigma^2}{\sigma^2+\tau^2 J_{jk}} \xrightarrow[J_{jk}\to\infty]{} \sigma^2,$$

also monotonically from below. Since σ is typically small, convergence is very fast indeed. Taking $\tau = \sigma$ as an example we see that even when $J_{jk} = 5$ we have

$$\frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2+\tau^2 J_{jk}} = \frac{5}{6}\widehat{d}_{jk}$$

and

$$\frac{\tau^2 J_{jk}\sigma^2}{\sigma^2+\tau^2 J_{jk}} = \frac{5}{6}\sigma^2.$$

We see that we are already within $\frac{1}{6}$ of the limit. Convergence is even faster for larger values of τ.

We also recall that the dominating process gives an upper bound for the value of J_{jk} at every location. Thus a good estimate for d_{jk} would be gained by taking the value of J_{jk} in the dominating process for those points where we do not know the exact value. This is a good solution but is unnecessary in some cases, as sometimes the value of λ_{dom} is so large that there is little advantage in using this value. Thus for exceptionally large values of λ_{dom} we simply use $N(\widehat{d}_{jk}, \sigma^2)$ numbers as our estimate of d_{jk}.

7.2 The Attractive-Repulsive Process

As with the issues concerning the implementation of the Bayesian wavelet thresholding algorithm, the issues which arose while simulating the attractive-repulsive process fall into two categories. Firstly, tracking the events in the dominating process and secondly, some approximations which were made.

7.2.1 Random Number Generators

As with the implementation in the previous section, three different random number generators were used to ensure that the simulations were correct. These three random number generators were used to generate:

	A	C	p
Independent draws	48271	0	$2^{31}-1$
Initial Configuration	16807	0	$2^{31}-1$
Times, marks and positions	2650845021	0	2^{32}

Table 7.1: Values of the parameters of the three linear congruential generators used during the implementation of the attractive-repulsive process.

1. Seeds for individual independent draws.

2. The initial configuration of events in the dominating process.

3. Birth and death times, marks and positions of events.

The generators used were all linear congruential pseudo random number generators as described in the previous section and the reasons for needing several sources of randomness are as outlined there. The values of the parameters of the recurrence relation given in equation (7.1) (which is used to generate the random numbers) are given in Table 7.1. These were all taken from Knuth[4] (1998a)

7.2.2 Threaded Binary Trees

It is clear that at every stage of the dominated coupling from the past algorithm it is necessary to store various pieces of information about each of the events that are currently alive. Since the list of live events is constantly changing, a dynamic allocation scheme may be best. Linked lists were initially considered for this task, but it seemed that trees might be better, due to the fact that inserting a new node should on average be faster. We made use of the 'threaded' binary trees proposed by Holt[5] (1963). Figure 7.1 contains a picture of a doubly-threaded binary tree

[4] The third random number generator was actually taken from a list of updates to this book since publication, which was downloaded from Knuth's web site (`http://www-cs-faculty.stanford.edu/~knuth/taocp.html`). It was found during an exhaustive search for the best value of A when $p = 2^{32}$

[5] The idea of threading was actually originally proposed by Perlis and Thornton (1960), but the idea of doubly-threaded binary trees was discovered independently and developed fully by Holt (1963).

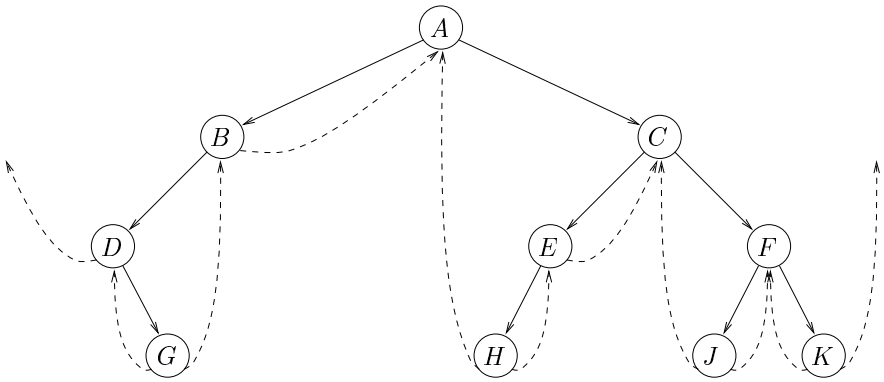

Figure 7.1: A doubly-threaded binary tree. Solid arrows represent children and dashed lines represent threads. As can be seen a link either points to a child or the predecessor/successor of the node.

like the ones used. For a full discussion of the use of trees in programming see Knuth (1997). The material below is a brief survey of some of the topics covered there.

The basic idea of using a tree structure for storing the information about the events is that each event should be represented by a `struct` of the following form:

```
struct event {
  short min_life;
  double birth;
  double death;
  double mark;
  coord pos;
  event *left;
  event *right;
  short tag;
};
```

The first five entries in the structure contain information about the event. The next two are pointers, which point to other events. The last is a tag which tells us whether the pointers point to children or to another part of the tree. It is these last three which enable us to implement a tree structure. Each time a new event is born we allocate space for a new event and 'insert' it in the tree of existing events according to either the new event's birth time or death time. Whether we use the new event's birth time or death time depends upon whether we are simulating the

dominating process or the maximum/minimum process. The insertion process is simple. Starting with the root node and proceeding recursively:

1. If the value of the node is less than the value of the new event:
   (a) If the node's `tag` indicates that the `right` pointer is a thread, go to step 3.
   (b) Otherwise, set `node=node->right` and go to step 1.
2. Otherwise (value of the node is greater than the value of the new event):
   (a) If the node's `tag` indicates that the `left` pointer is a thread, go to step 4.
   (b) Otherwise, set `node=node->left` and go to step 1.
3. (From step 1(a)) Set the new event's `right` pointer to be the value of the current node's `right` pointer, the event's `left` pointer to be the address of the current node, the node's `right` pointer to be the address of the new event, and set the current node's `tag` to show that `node->right` is no longer a thread.
4. (From step 2(a)) Set the new event's `left` pointer to be the value of the current node's `left` pointer, the event's `right` pointer to be the address of the current node, the node's `left` pointer to be the address of the new event, and set the current node's `tag` to show that `node->left` is no longer a thread.

The algorithm given is used to insert new points into the tree while generating the dominating process backwards to time $-M$. Replacing 'less than' by 'greater than' in step 1, and 'greater than' by 'less than' in step 2 gives the algorithm used to insert new points into the tree while simulating the minimum and maximum processes forwards to time 0. The 'value' of a node is determined by either its birth time or its death time, depending upon whether we are simulating the dominating process or the maximum/minimum process. Before beginning the insertion process

the new event's `tag` is initialised to show that both `left` and `right` pointers are threads.

It is often important to be able to traverse the tree of events in order to, for example, print the locations of the events, or count the events. Inorder traversal (i.e. traversal subject to the ordering that was used for insertion) is made easier by the threads we have mentioned earlier. Threads make it possible to traverse a tree inorder without using a stack. For details, see Knuth (1997).

Since traversal was necessary for several different reasons, two general purpose functions were written, `applyinorder` and `testinorder` to traverse the tree. These two functions took other functions as parameters (more precisely, they took pointers to functions as parameters). They would then traverse the tree, performing the function at each node. The difference between the two is that `testinorder` would stop as soon as the function failed on a node, whereas `applyinorder` applied the function at each node without regard for a return value. The function `testinorder` was used to test coalescence. Stopping after failure was the desired behaviour since a single event which was in the maximum process but not in the minimum process meant non-coalescence. The function `applyinorder` also took a third parameter (the first was the root node of the tree, the second the function). An example of this parameter's use was as a format string together with the function `printcoord`. See Section A.2.2 for further examples.

7.2.3 Calculating overlap

The main difficulty encountered while simulating this model was what to do about the calculations involving acceptance or rejection of new events. During this step it is necessary to calculate

$$m\big\{(x \oplus G_1) \setminus X \oplus G_1\big\}$$

and

$$m\big\{(x \oplus G_2) \setminus X \oplus G_2\big\}$$

for both the minimum and the maximum processes. Since calculating this overlap exactly is a non-trivial exercise it was decided to simplify matters by instead centring a rectangular grid of points at x with side-length $2r$ (where r is the radius of the disk G) and counting the number of points in this grid which were in $x \oplus G$ but not in $X \oplus G$. We chose a grid side length of 20 points, giving a total of 400 points, over 300 of which will be in $x \oplus G$.

Appendix A

Implementational details

The source code for all of the programs I have written is available on the web from my home page:

`http://www.maths.bris.ac.uk/~magka/`

Here we discuss that implementation in some detail, making heavy reference to the source.

A.1 Bayesian Wavelet Thresholding

The algorithm for the Bayesian wavelet thresholding model comes in two pieces: First of all the front end which was used to generate multiple independent draws and secondly the program itself, which generates a single draw from the target distribution using dominated coupling from the past.

A.1.1 Automating several runs of the program

In order to automate the process of running the program multiple times to obtain several independent draws from the posterior, a shell script and various little C programs were written. The pseudo random number generator in Section 7.1.2 is an example of one of these little C programs.

The shell script for running the program on the 'heavisine' example data was as follows[1]:

[1] Due to page width restrictions, one line in this script has been split across two

Appendix A. Implementational details

```
#!/bin/csh

if ( -e /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt.gz) then
  /bin/rm /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt.gz
else if ( -e /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt.bz2 ) then
  /bin/rm /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt.bz2
else if ( -e /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt ) then
  /bin/rm /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt
endif

touch /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt
rand $1 >numbers
foreach i (‘sequence $1‘)
cat protofile >file$i
uncat numbers $i >>file$i
/bin/nice -10 cftpsim file$i \
              >> /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt
/bin/rm file$i
end
/bin/rm numbers
bzip2 /home/magka/Cfiles/Data/hvs/out/noisyhvsdwt
```

The first few lines (up to `rand $1 >numbers`[2]) simply set up the output files, removing any previous occurrences of them if they exist. The next line calls the program `rand`, which generates two times the number passed to it as the first parameter of random numbers between 0 and $2^{31} - 1$ and puts this output in the file `numbers`. The random number generator used in `rand` is that of Section 7.1.2. Next we enter a `for` loop, which runs once for each `i` in the output of `sequence $1`. The program `sequence` simply generates a list of integers between 0 and $n-1$, where n is the first parameter passed to it. This `for` loop first writes an input file for the program from the concatenation of `protofile` and the next two numbers in the file `numbers` (this is done using the program `uncat`). It then runs the program

lines. To indicate this the character "\" is used to indicate that the following line should be treated as if it were appended to the current line.

[2] A note on variables in shell scripts: If our shell script is called, for example, `run`, then we would call it using the command `./run` from the command line. If the script took any *parameters* then we would call it using the command `./run par1 par2`, etc. depending on the number of parameters required by the script. These parameters are referenced within the script using the notation `$1` for the first parameter, `$2` for the second parameter, and so on. It is also possible to use variables which exist solely within the scope of the script. An example of this is `i` in the `foreach` loop in our script. It is defined in the `foreach` statement and referenced as `$i`.

with this input file and finally deletes the input file to save cluttering the file space with hundreds of files each time the program is run. The last two lines remove the file `numbers` and `bzip2`[3] the output file to save disk space. An example of the file `protofile` used as part of the input file is as follows:

```
/home/magka/Cfiles/Data/hvs/noisyhvs512dwt
0.2973434
0.15
0.7
3.0
128.0
1
```

Each line denotes one variable input to the program. The first is the location of the file containing the data. The second is the value of σ, the third is the value of τ^2, the fourth is the value of λ, the fifth is the value of γ, the sixth is the initial value of $-M$ used and the seventh is the value of the exponent of J_{jk} in the prior.

The scripts used to generate multiple independent draws from the posterior from different data sets are the same except for differences in file names.

A.1.2 A systematic deconstruction of the implementation

Having covered some general aspects of the implementation we now turn to the specifics. There are 23 different functions (other than standard library routines) in the implementation of the program. The program was split into these separate functions for several reasons:

1. To split the algorithm into separate pieces which could be tested and written individually.

2. To make it possible to split the program between several files so that if a piece was changed it was only necessary to recompile that piece and link the pieces rather than having to recompile the entire program.

[3] `bzip2` is a block-sorting file compressor which usually offers better compression (especially of text files) than the more traditional Lempel-Ziv coding used by compression programs such as `gzip`.

3. To make it possible to re-use parts of the code without any additional editing.

The entire algorithm can be written in as follows:

1. Read parameter values and file name of the data file.
2. Scan the data file to find the number of data locations.
3. Allocate memory for this data and all of the other information stored about each location.
4. Read data into the memory which has just been allocated.
5. Calculate the initial values of the dominating process at time 0
6. Repeat:
   (a) Generate the dominating process from time 0 backwards to time $-M$.
   (b) Run minimum and maximum processes forwards in time from $-M$ to 0.
   (c) Double M.

   until the minimum and maximum processes have coalesced.
7. For each location
   (a) If there are $J_{jk} > 0$ points alive at the location then generate a $N\left(\frac{\tau^2 J_{jk}\widehat{d}_{jk}}{\sigma^2+\tau^2 J_{jk}}, \frac{\sigma^2\tau^2 J_{jk}}{\sigma^2+\tau^2 J_{jk}}\right)$ number and output that to standard out.
   (b) If there are no live points at the location write 0 to standard out.
8. Print the value of $-M$ at which coalescence occurred to standard error.

We now proceed to expand upon this algorithm and fill in some of the details. Each time we come across a new function in the C code we name it and say which file it is contained within so that it can be referred to with ease in the sources.

Step 1 is carried out by the function `getparameters` which is in the file `getp.c`. This basic input step was put in a separate function to make it possible to have two different methods of input: interactive and non-interactive. The interactive

version prompts for the values of the parameters, while the non-interactive version reads the values from a file which is specified as the first (only) argument to the program on the command line. Steps 2-4 are fairly trivial and step 5 is where the simulation proper begins.

Step 5 is carried out by the function `initialise` in the file `init.c`. The algorithm for this function is:

Repeat, for each location in the process (index by 'i'):

1. Calculate the natural logarithm of the value of λ for location 'i'.
2. If this is less than 4 (i.e. if $\lambda \lesssim 54$) use the exact value of λ
3. Otherwise set this location to 'always on' (see Section 7.1.5) and
   (a) If $\log(\lambda) < 16$ (i.e. if $\lambda \lesssim 8,886,281$) find the value of the dominating process at this location at time 0 and set the number of points alive in the maximum and minimum processes to this value.
   (b) Otherwise set the values of the maximum and minimum processes to unity.[4]
4. If the location was not 'always on' then
   (a) Decide how many points at the current location are alive in the dominating process at time 0.
   (b) Give the current location a `markseed` and a `timeseed`.
5. Label each point.
6. Calculate the value of p_{min} for point i.

The reasons for steps 1 to 3 are covered in Section 7.1.5. Step 4(b) is covered in Section 7.1.3. Step 5 is there so that when, later in the program, the order of the locations is sorted according to the time until the next birth or death happens

[4] This is done to indicate that λ for this location is very big. The probability of an 'always on' point having only one point alive (and thus the wrong thing happening due to the use of the value one as the signal here) is so small that in practise it will never happen.

at that location we can still determine where in the configuration that location belongs. Step 6 is used to set up the initial configuration of the minimum process.

Returning to the main program, step 6(a) is carried out by the function `dominating` in the file `dom.c`. The algorithm for this function is:

1. Calculate how many events (births and deaths) happen to each location between time 0 and time $-M$.

2. For each location i in turn:

   (a) If location i is not 'always on' and the number calculated in step (1) is non-zero then:

      i. Give each point which is alive in the dominating process at time 0 a birth time and a mark and allocate a pointer to the point.

      ii. Run through the birth-death process for the dominating process backwards from 0 to $-M$. Give each point a mark and store the birth and death times. Also allocate a pointer to each point.

      iii. Sort the pointers according to birth times so that we now have a list sorted by death times and one sorted by birth times. Run through the process to find the maximum number of points which are alive at location i at any one time.

      iv. Continue to run through the births until either one is before -M or there are no points left. Store the birth, death and mark info for the first point to be born at location i after time -M.

      v. Store the number of points alive in the dominating process at time -M as the initial value of the maximum process. Use the points' marks to determine whether to keep these in the minimum process. Store the death times and sort them.

   (b) Otherwise store indicators to show that there are no death times or birth times and if the point is not 'always on' set the value of the maximum and minimum processes to zero.

Step 1. is done so that memory can be allocated in step 2. (and later in the program in part of step 6(b) of the main algorithm) and so that step ii. and others can be carried out the correct number of times. Steps i.–v. do lots of housekeeping operations and find the state of the process at time $-M$ and the point which is born soonest after time $-M$. Step (b) is more housekeeping, telling the program later on that there are no births or deaths for these locations between time $-M$ and 0.

Step 6(b) is the real core of the program and is carried out by the function `process` in the file `process.c`. It takes the dominating process generated in the previous step and uses it to run the maximum and minimum processes from time $-M$ back to time 0. The algorithm for this function is:

1. Work out which locations are in $X \oplus G$.
2. Sort two arrays of pointers which point to the data stored about each location — one according to the time until the next birth, the other according to the time until the next death.
3. Repeat:
   (a) If next event is a death:
      i. Delete the point from the max and min processes and update $X \oplus G$.
      ii. Re-order the array of pointers which is sorted according to the time until the next death according to the next death time at the location which has just experienced a death (if there are no more live points at this location then put it at the end of the list).
   (b) Otherwise (a birth):
      i. Calculate $m((x \oplus G) \setminus (Y(-M, u) \oplus G))$ (recall from Section 4.1.4 that $Y(-M, u) \oplus G$ is our notation for $X \oplus G$ at the time instant u, when we have simulated from time $-M$) for that point in both processes to find the rejection probabilities on page 123.

ii. If the rejection probability for the maximum process is less than the point's mark then add the point and add its death time to the list for that location. If this point is the only one at this location then update $X \oplus G$. Re-order the array of pointers which is sorted according to the time until the next death if necessary.
iii. Repeat the previous step for the minimum process. The locations will not need to be re-ordered.
iv. Find the details of the next birth at the location we are looking at (the one which has just had a birth).
v. Re-order the array of pointers which is sorted according to the time until the next birth.

Until all events at all points have been dealt with.

4. Determine whether the min and max processes are the same and return this fact.

Step 1 above is carried out by the function `covered` in the file `cover.c` The algorithm for this is simple but long-winded, so we do not describe it here, but refer the interested reader to the source code.

Step 2 is carried out by the function `ssort` in the file `ssort.c`. It is a shell sort algorithm using the gaps recommended in (Knuth 1998b). Comparisons are done by the function `compare` in the file `process.c`.

If the input file has more than 64 data points in it then step 3(a.i) is carried out by the two functions `mindeath` and `maxdeath` in the files `mindeath.c` and `maxdeath.c` respectively. If not then `covered` is used as in point (1) above. This is because if the dataset is small it is most efficient just to re-generate the whole of $X \oplus G$, whereas if the dataset is large it is more efficient to work out exactly which locations could be affected by a death at the given location and only re-generate $X \oplus G$ at those locations. `mindeath` and `maxdeath` are simply very long and complicated conditionals which determine where the location is and then

work out which points in $x \oplus G$ are retained in $X \oplus G$ when x is removed from the process. Due to the complex structure that the point process lives on, these functions account for more than half of the total code. Step 3(a.ii) simply shuffles the list up and then inserts the location where the death has just occurred at the correct place.

In step 3(b.i) $m((x \oplus G) \setminus (Y(-M, u) \oplus G))$ is calculated by the function `measure` in the file `measure.c`. Updating $X \oplus G$ for the max processes in step 3(b.ii) is done by the function `maxbirth` in the file `maxbirth.c`. The algorithm for this is simple but long-winded, so we refer the interested reader to the source code. The re-ordering part is only done if the death time of the point which has just been born is sooner than any other death time at that location and is done by the same shuffle-up method as in step 3(a.ii). Step 3(b.iv) is done in the same way as we found the first birth times in the previous step of the main algorithm and step 3(b.v) again uses the same method as step 3(a.ii).

Returning once again to the main algorithm, step 7 is carried out by the function `rnorm` in the file `rnorm.c`. This is an implementation of the Box-Muller scheme for generating Gaussian random variables suggested in Press et al. (1993).

A.2 The Attractive-Repulsive Process

As with the implementation discussed in the previous section, the algorithm for the attractive-repulsive process comes in two pieces, the front end for generating multiple independent draws and the "back end", which generates single draws from the desired distribution when passed a collection of parameter values and seeds. We begin by discussing the front end.

A.2.1 Automating several runs of the program

A procedure very similar to that outlined in Section A.1.1 was used to simulate multiple independent draws with given parameter values. We briefly outline this procedure below.

Appendix A. Implementational details

A very slightly edited version of the shell script used to run the program multiple times is reproduced below:

```
#!/bin/csh

if ( -e k_all ) then
    /bin/rm k_all
endif
if ( -e t_all ) then
    /bin/rm t_all
endif
if ( -e f_all ) then
    /bin/rm f_all
endif
if ( -e g_all ) then
    /bin/rm g_all
endif
if ( -e mipd.out ) then
    /bin/rm mipd.out
endif
if ( -e events0 ) then
    /bin/rm events*
endif

./rand3 $1 >numbers
touch k_all
touch f_all
touch g_all
touch t_all
touch mipd.out
foreach i (`./sequence $1`)
    cat protofile >file$i
    ./uncat3 numbers $i >>file$i
    echo "1.0 1.0" >events$i
    /bin/nice -15 ./attrepsim file$i >>events$i
    /bin/nice -15 ./tork events$i >events{$i}\k
    cat events{$i}\k >>k_all
    /bin/nice -15 ./tort events$i >events{$i}\t
    cat events{$i}\t >>t_all
    /bin/nice -15 ./torg events$i >events{$i}\g
    cat events{$i}\g >>g_all
    /bin/nice -15 ./torf events$i >events{$i}\f
    cat events{$i}\f >>f_all
    /bin/nice -15 tormipd events$i >> mipd.out
    /bin/rm file$i
```

```
end
/bin/rm numbers
```

As can be seen, the script does not only run the program (`attrepsim`) multiple times, it also runs several other programs on the output of the program. These programs (`tork`, `tort`, `torg`, `torf` and `tormipd`) calculate the K, T, G and F functions and the minimum inter-event distances for the point patterns generated by the main program. See Sections 4.2 and 4.4.2 for details of what these functions are. The reason for the `tor` prefix to each of their names is because they use toroidal boundary conditions. The programs `rand3` and `uncat3` perform the same functions as `rand` and `uncat` described in Section A.1.1 except that they work on triples of numbers rather than pairs, since three seeds are needed for `attrepsim`, compared with two for `cftpsim`. An example of the input file `protofile` is as follows:

```
100
1000.0
0.05
0.03
0.1
0.03
0.1
32
```

As before, each line denotes one variable input to the program. The first is the value of λ, the second is the value of γ_1, the third is the value of γ_2, the fourth and sixth are parameters of G_1, the fifth and seventh are parameters of G_2 and the eighth is the initial value of $-M$ used. The reason for two parameters for G_1 and G_2 is that the program actually uses ellipses centred at the events rather than circles. The first parameter is then the length of the semi-axis in the x-direction and the second is the length of the semi-axis in the y-direction.

A.2.2 A systematic deconstruction of the implementation

Having covered general aspects of the implementation we now turn to specifics. Other than standard library routines there are 28 different functions in the implementation of the program. Reasons for splitting the program up in this fashion are given in Section A.1.2.

We begin by discussion the algorithm briefly before going into each of the steps in more detail. It should be noted that much of the implementation is very similar to that of Section A.1.2. As a result, names for similar functions and files are often similar or even the same between the two implementations. This does not mean, however, that files or functions with the same name are identical to those discussed in Section A.1.2. In fact only the random number generators were used without modification.

The overall algorithm is as follows:

1. Read parameter values.
2. Calculate some summaries of these parameters.
3. Repeat:
   (a) Calculate the initial values of the dominating process at time 0
   (b) Generate the dominating process from time 0 backwards to time $-M$.
   (c) Run minimum and maximum processes forwards in time from $-M$ to 0.
   (d) Double M.

   until the minimum and maximum processes have coalesced.
4. Print the locations of the events in the coalesced process to standard out.
5. Print the value of $-M$ at which coalescence occurred to standard error.

Step 1 is carried out by the function `getparameters` in the file `getp.c` and is fairly trivial. Step 2 calculates $\lambda\gamma_2^{-m(G_2)}$ and $\gamma_1^{-m(G_1)}$ to minimise the number of parameters it is necessary to pass in steps 3(a) and 3(b) (i.e. purely for aesthetic

reasons, as the passing of the extra parameters and the extra calculations in each loop are likely to make very little difference!)

Steps 3(a) and 3(b) are carried out by the function `dominating` in the file `dom.c`. The algorithm for this function is:

1. Calculate initial distribution by generating a Poisson($\lambda\gamma_2^{-m(G_2)}$) number and scattering this many events Uniformly in the unit square:
   (a) Give each event a $U[0,1] \times U[0,1]$ position, a $U[0,1]$ mark and an Exponential(1) birth time[5] (remember, we initially simulate backwards in time, so 'births' happen after 'deaths').
   (b) Insert each event into our tree[6] of currently living events according to its birth time.

2. Generate the dominating process from time 0 backwards to time $-M$ by repeating the following steps until all births and deaths in $(-M, 0)$ are accounted for:
   (a) Generate an Exponential($\lambda\gamma_2^{-m(G_2)}$) random number. This is the time until the next death (recall again that we move backwards in time).
   (b) If the time until the next death is less than the time until the next birth:
      i. Give the new event a $U[0,1] \times U[0,1]$ position, a $U[0,1]$ mark and an Exponential(1) birth time.
      ii. Insert the event into our tree of currently living events according to its birth time.
   (c) Otherwise (a birth occurs next) remove the event which is due to be born.
   (d) Update the time-until-next-death and time-until-next-birth variables.

[5] We don't care when the event dies, as we know that it is alive at time 0, and we don't care what happens any further 'forward' in time.

[6] See Section 7.2.2 for a discussion of the role of tree structures in our algorithm.

3. Decide which events in the dominating process at $-M$ are alive in the minimum process by rejecting those whose mark is less than $\gamma_1^{-m(G_1)}\gamma_2^{m(G_2)}$.
4. Transform the tree of events so that it is ordered by death time rather than birth time.

Steps 1(a) and 2(b.i) were taken care of by the function `newnode`, Steps 1(b) and 2(b.ii) were done by the function `insert_dom`, Step 2(c) was performed using the function `remove_left`, Step 3 used the function `in_min` in conjunction with the function `applyinorder` and Step 4 used the function `birth_to_death`. All of these functions other than `in_min` (which is in the file `dom.c`) are found in the file `treet.c`. The functions `insert_dom` and `applyinorder` are discussed in Section 7.2.2

Returning to the main algorithm, we come to step 3(c), which is the core of the coupling from the past algorithm. This was carried out by the function `process` in the file `process.c`. It takes the maximum and minimum processes, generated at time $-M$ in the previous step, and runs them forwards to time 0 using the transitions of the dominating processes. The algorithm for this function is:

1. Run minimum and maximum processes forwards in time from $-M$ to 0 by repeating the following steps until there are no more births or deaths before 0:
   (a) If the next event (in the sense defined on page 96) is a death, remove the event and update the variable holding the next death time.
   (b) Otherwise (next event (in the sense defined on page 96) is a birth):
      i. Calculate the probability of accepting the event.
      ii. If we accept the event in the maximum process:
         A. Insert it into the tree of events according to its death time.
         B. If we also accept the event in the minimum process, set the flag which says that.
         C. If new event dies before any other event, update the variable holding the next death time.

iii. Otherwise (rejecting the event) free up the space allocated for that event.

iv. Update the value of the next event to be born.

2. Test whether the maximum and minimum processes have coalesced.

Step 1(a) is carried out by the function `remove_left` in the file `treet.c` and using the function `left`, also in the file `treet.c`, to find the next death time. Step 1(b.i) uses the function `calc_p` in the file `ellipse.c`. The algorithm for this function is:

1. Count the number of events in the minimum and maximum processes.
2. Make an array containing the foci of ellipses centred at the events in the process, and calculate the foci of the ellipse centred at the new event. Do this for both G_1 and G_2.
3. Calculate how much extra area the ellipse centred at the new event adds to $m(X \oplus G_1)$ and $m(X \oplus G_2)$ in both the minimum and maximum processes.
4. Use the values calculated in the previous step to calculate the acceptance probabilities.

Step 1 used the functions `count` and `count_min` in the file `ellipse.c` together with the function `applyinorder` in the file `treet.c`. Step 2 used the functions `makeellipseinorder` and `makeellipse` in the file `treet.c`. Step 3 used the function `extra` in the file `ellipse.c`. This lengthy function uses a 20×20 grid to discretize the problem, and is covered briefly in Section 7.2.3. Step 4 is trivial.

Returning to the algorithm for Step 3(c) of the main algorithm, Step 1(b.ii.A) is performed by the function `insert_real` in the file `treet.c`, which is discussed in Section 7.2.2. Steps 1(b.ii.B), 1(b.ii.C) and 1(b.iii) are trivial. Step 1(b.iv) is performed by the function `nextbirth` in the file `process.c`. Step 2 uses the function `difference` in the file `process.c` together with the function `testinorder` in the file `treet.c`.

Steps 4 and 5 of the main algorithm are trivial.

Bibliography

Abramovich, F. and Y. Benjamini (1996). Adaptive thresholding of wavelet coefficients. *Computational Statistics and Data Analysis 22*, 351–361.

Abramovich, F., T. Sapatinas, and B. W. Silverman (1998). Wavelet thresholding via a Bayesian approach. *Journal of the Royal Statistical Society, Series B 60*, 725–749.

Asmussen, S., P. W. Glynn, and H. Thorisson (1992). Stationary detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation 2*, 130–157.

Baddeley, A. and R. Turner (2000). Practical maximum pseudolikelihood for spatial point patterns. *Australian and New Zealand Journal of Statistics 42*, 283–322.

Baddeley, A. J. and B. W. Silverman (1984). A cautionary example on the use of second-order methods for analyzing point patterns. *Biometrics 40*, 1089–1093.

Baddeley, A. J. and M. N. M. van Lieshout (1995). Area-interaction point processes. *Annals of the Institute for Statistical Mathematics 47*, 601–619.

Bartlett, M. S. (1964). The spectral analysis of two-dimensional point processes. *Biometrika 51*, 299–311.

Benjamini, Y. and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B 57*, 289–300.

Berman, M. and R. Turner (1992). Approximating point process likelihoods with GLIM. *Applied Statistics 41*, 31–38.

Berthelsen, K. K. and J. Møller (2001). Perfect simulation and inference for spatial point processes. Technical Report R-01-2009, Department of Mathematical Sciences, Aalborg University.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B 36*, 192–236.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician 24*, 179–195.

Besag, J. (1977a). Some methods of statistical analysis for spatial data. *Bulletin of the International Statistical Institute 47*, 77–92.

Besag, J. E. (1977b). Comment on "modelling spatial patterns" by B.D. Ripley. *Journal of the Royal Statistical Society, Series B 39*, 193–195.

Brooks, S. P. (1998). Markov chain Monte Carlo method and its application. *The Statistician 47*, 69–100.

Cai, H. (1999). Exact sampling using auxiliary variables. In *Proceedings of the Statistical Computing Section*, pp. 139–142. American Statistical Association.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. New York: John Wiley & Sons.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. Philadelphia, Pennsylvania: SIAM.

Diggle, P. J. (1978). On parameter estimation for spatial point processes. *Journal of the Royal Statistical Society, Series B 40*, 178–181.

Donoho, D. L. and I. M. Johnstone (1994). Ideal spatial adaption by wavelet shrinkage. *Biometrika 81*, 425–455.

BIBLIOGRAPHY

Donoho, D. L. and I. M. Johnstone (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association 90*, 1200–1224.

Fernández, R., P. A. Ferrari, and N. L. Garcia (1999). Perfect simulation for interacting point processes, loss networks and Ising models. arXiv:math.PR/9911162.

Fill, J. A. (1998). An interruptible algorithm for perfect sampling via Markov chains. *The Annals of Applied Probability 8*, 131–162.

Fill, J. A. and M. Huber (2000). The randomness recycler: A new technique for perfect sampling. arXiv:math.PR/0009242.

Fill, J. A., M. Machida, D. J. Murdoch, and J. S. Rosenthal (2000). Extension of Fill's perfect rejection sampling algorithm to general chains. *Random Structures and Algorithms 17*, 290–316.

Gamerman, D. (1997). *Markov Chain Monte Carlo*. London: Chapman & Hall.

Geyer, C. J. and E. A. Thompson (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society, Series B 54*, 657–699.

Green, P. J. and D. J. Murdoch (1998). Exact sampling for Bayesian inference: towards general purpose algorithms (with discussion). In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (Eds.), *Bayesian Statistics 6*, pp. 301–321. Oxford University Press. Presented as an invited paper at the 6th Valencia International Meeting on Bayesian Statistics, Alcossebre, Spain, June 1998.

Häggström, O., M. N. M. van Lieshout, and J. Møller (1999). Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli 5*, 641–658.

Hobert, J. P., C. P. Robert, and D. M. Titterington (1999). On perfect simulation for some mixtures of distributions. *Statistics and Computing 9*, 287–298.

Holt, A. W. (1963). *A Mathematical and Applied Investigation of Tree Structures*. Thesis, University of Pennsylvania.

Huber, M. (1999). Perfect sampling without a lifetime commitment. Preprint.

Jensen, J. L. and J. Møller (1991). Pseudolikelihood for exponential family models of spatial point processes. *Annals of Applied Probability 1*, 445–461.

Jost, J. (1998). *Postmodern Analysis*. Berlin: Springer-Verlag.

Kelly, F. P. (1979). *Reversibility and Stochastic Networks*. Chichester: John Wiley & Sons.

Kelly, F. P. and B. D. Ripley (1976). A note on Strauss's model for clustering. *Biometrika 63*, 357–360.

Kendall, W. S. (1997). On some weighted Boolean models. In D. Jeulin (Ed.), *Advances in Theory and Applications of Random Sets*, pp. 105–120. World Scientific Publishing Company.

Kendall, W. S. (1998). Perfect simulation for the area-interaction point process. In L. Accardi and C. C. Heyde (Eds.), *Probability Towards 2000*, pp. 218–234. Springer.

Kendall, W. S. and J. Møller (1999). Perfect Metropolis-Hastings simulation of locally stable point processes. To appear in *Advances in Applied Probability*.

Kerscher, M. (1998). Regularity in the distribution of superclusters? *Astronomy and Astrophysics 336*, 29–34.

Kingman, J. F. C. and S. J. Taylor (1966). *Introduction to Measure and Probability*. Cambridge: Cambridge University Press.

Knuth, D. E. (1997). *Fundamental Algorithms* (Third ed.), Volume 1 of *The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley.

BIBLIOGRAPHY

Knuth, D. E. (1998a). *Seminumerical Algorithms* (Third ed.), Volume 2 of *The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley.

Knuth, D. E. (1998b). *Sorting and Searching* (Second ed.), Volume 3 of *The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley.

Lindvall, T. (1992). *Lectures on the Coupling Method*. New York: John Wiley & Sons.

Loizeaux, M. A. (2001). *Bayesian inference for spatial point processes via perfect sampling*. Ph. D. thesis, Department of Statistics, Florida State University.

Lovász, L. and P. Winkler (1995). Exact mixing in an unknown Markov chain. *Electronic Journal of Combinatorics 2*. Paper #R15.

Lund, J. and E. Thönnes (2000). Perfect simulation of point processes given noisy observations. Technical Report 366, Department of Statistics, University of Warwick.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence. 11*, 674–693.

Matheron, G. (1975). *Random Sets and Integral Geometry*. New York: John Wiley & Sons.

Meyn, S. P. and R. L. Tweedie (1993). *Markov Chains and Stochastic Stability*. London: Springer-Verlag.

Mira, A., J. Møller, and G. O. Roberts (2001). Perfect slice samplers. *Journal of the Royal Statistical Society, Series B 63*, 593–606.

Møller, J. and G. K. Nicholls (1999). Perfect simulation for sample-based inference. Technical Report R-99-2011, Department of Mathematical Sciences, Aalborg University.

Møller, J. and K. Schladitz (1999). Extensions of Fill's algorithm for perfect simulation. *Journal of the Royal Statistical Society, Series B 61*, 955–969.

Møller, J. and R. Waagepetersen (1998). Markov connected component fields. *Advances of Applied Probability 30*, 1–35.

Murdoch, D. J. and P. J. Green (1998). Exact sampling from a continuous state space. *Scandinavian Journal of Statistics 25*, 483–502.

Murdoch, D. J. and J. S. Rosenthal (2000). Efficient use of exact samples. *Statistics and Computing 10*, 237–243.

Nason, G. P. (1996). Wavelet shrinkage using cross-validation. *Journal of the Royal Statistical Society, Series B 58*, 463–479.

Norris, J. R. (1997). *Markov Chains*. Cambridge: Cambridge University Press.

Perlis, A. J. and C. Thornton (1960). Symbol manipulation by threaded lists. *Communications of the ACM 3*, 195–204.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1993). *Numerical Recipes in C* (Second ed.). Cambridge: Cambridge University Press.

Propp, J. G. and D. B. Wilson (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms 9*, 223–252.

Propp, J. G. and D. B. Wilson (1998). How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms 27*, 170–217.

Ripley, B. D. (1976). The second-order analysis of stationary point processes. *Journal of Applied Probability 13*, 255–266.

Ripley, B. D. (1977). Modelling spatial patterns (with discussion). *Journal of the Royal Statistical Society, Series B 39*, 172–212.

Ripley, B. D. and F. P. Kelly (1977). Markov point processes. *Journal of the London Mathematical Society 15*, 188–192.

BIBLIOGRAPHY

Ripley, B. D. and B. W. Silverman (1978). Quick tests for spatial interaction. *Biometrika 65*, 641–642.

Ross, S. M. (1990). *A Course in Simulation*. New York: Macmillan.

Schladitz, K. and A. J. Baddeley (2000). A third order point process characteristic. *Scandinavian Journal of Statistics 27*, 657–671.

Silverman, B. W. and T. C. Brown (1978). Short distances, flat triangles and Poisson limits. *Journal of Applied Probability 15*, 815–825.

Stein, C. (1981). Estimation of the mean of a multivariate normal distribution. *Annals of Statistics 9*, 1135–1151.

Stoyan, D., W. S. Kendall, and J. Mecke (1995). *Stochastic Geometry and its applications* (Second ed.). Chichester: John Wiley & Sons.

Strauss, D. J. (1975). A model for clustering. *Biometrika 62*, 467–475.

Sutherland, W. A. (1975). *Introduction to Metric and Topological Spaces*. Oxford: Oxford University Press.

Thönnes, E. (1999). Perfect simulation of some point processes for the impatient user. *Advances in Applied Probability 31*, 69–87.

van Lieshout, M. N. M. and A. J. Baddeley (1996). A nonparametric measure of spatial interaction in point patterns. *Statistica Neerlandica 50*, 344–361.

Widom, B. and J. S. Rowlinson (1970). A new model for the study of liquid-vapor phase transitions. *Journal of Chemical Physics 52*, 1670–1684.

Williams, D. (1991). *Probability with Martingales*. Cambridge: Cambridge University Press.