INSTITUTE OF PHYSICS PUBLISHING INVERSE PROBLEMS

Inverse Problems **18** (2002) R29–R54 PII: S0266-5611(02)79161-2

TOPICAL REVIEW

Monte Carlo analysis of inverse problems

Klaus Mosegaard[1,3] **and Malcolm Sambridge**[2]

[1] Niels Bohr Institute, Department of Geophysics, Juliane Maries Vej 30, 2100 Copenhagen, Denmark
[2] Research School of Earth Sciences, Australian National University, Canberra, ACT 0200, Australia

E-mail: klaus@gfy.ku.dk

Received 19 July 2001, in final form 9 January 2002
Published 8 April 2002
Online at stacks.iop.org/IP/18/R29

Abstract

Monte Carlo methods have become important in analysis of nonlinear inverse problems where no analytical expression for the forward relation between data and model parameters is available, and where linearization is unsuccessful. In such cases a direct mathematical treatment is impossible, but the forward relation materializes itself as an algorithm allowing data to be calculated for any given model.

Monte Carlo methods can be divided into two categories: the sampling methods and the optimization methods. Monte Carlo sampling is useful when the space of feasible solutions is to be explored, and measures of resolution and uncertainty of solution are needed. The Metropolis algorithm and the Gibbs sampler are the most widely used Monte Carlo samplers for this purpose, but these methods can be refined and supplemented in various ways of which the neighbourhood algorithm is a notable example.

Monte Carlo optimization methods are powerful tools when searching for globally optimal solutions amongst numerous local optima. Simulated annealing and genetic algorithms have shown their strength in this respect, but they suffer from the same fundamental problem as the Monte Carlo sampling methods: no provably optimal strategy for tuning these methods to a given problem has been found, only a number of approximate methods.

(Some figures in this article are in colour only in the electronic version)

1. Introduction

About a hundred years ago, it was recognized that integrals of the form

$$I = \int_{\mathcal{X}} h(\boldsymbol{x}) f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \tag{1}$$

[3] Author to whom any correspondence should be addressed.

where $h(\boldsymbol{x})$ and $f(\boldsymbol{x})$ are functions for which $h(\boldsymbol{x})f(\boldsymbol{x})$ is integrable over the space $\mathcal{X}$, and $f(\boldsymbol{x})$ is a non-negative valued, integrable function satisfying

$$\int_{\mathcal{X}} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1 \tag{2}$$

could, in principle, be evaluated numerically by generating random samples $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ of $\boldsymbol{x}$ using $f(\boldsymbol{x})$ as a probability distribution (see, e.g., Housholder (1951)). An approximation to the integral could then be calculated as

$$I \approx \frac{1}{N}\sum_{n=1}^{N} h(\boldsymbol{x}_n). \tag{3}$$

Practical use of this method has today become a reality through application of *Monte Carlo* algorithms running on high-speed computers. Monte Carlo methods are numerical processes that produce so-called *pseudo-random numbers*, that is, a series of numbers that appear random if tested with any reasonable statistical test. The basic operation of a Monte Carlo algorithm is generation of pseudo-random numbers uniformly distributed over the interval [0, 1]. Once such a sample x_i is produced, it can be transformed into a pseudo-random sample from any given one-dimensional probability distribution $f(x)$, using simple rules.

As long as we work in one dimension, the Monte Carlo method is inefficient, and hence not a useful alternative to more direct methods for numerical evaluation of (1). If however, $\boldsymbol{x}$ belongs to a high-dimensional space, the Monte Carlo method may become the only feasible method. All other numerical methods, using N points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ in an M-dimensional space $\mathcal{X}$ to produce an approximation to, e.g., (1), have an absolute error that decreases no faster than $N^{-1/M}$, whereas the absolute error of the Monte Carlo method decreases as $N^{-1/2}$, that is, independent of the dimension of the space (see, e.g., Fishman (1996)).

The history of Monte Carlo methods is long, but their application to the solution of scientific problems begins with von Neumann, Ulam and Fermi who used a Monte Carlo method in nuclear reaction studies. The name 'the Monte Carlo method' was first used by Metropolis and Ulam (1949), and 4 years later Metropolis *et al* (1953) published their Markov chain-based algorithm for (asymptotic) sampling of Gibbs–Boltzmann distributions in high-dimensional spaces. This algorithm, now known as the Metropolis algorithm, was in fact the first major scientific algorithm to be run on a digital computer. It was a biased random walk whose individual steps (iterations) were based on very simple probabilistic rules. One important feature of the Metropolis algorithm was that full information on the distribution $p(\boldsymbol{x})$ to be sampled was unnecessary: as long as ratios $p(\boldsymbol{x}_i)/p(\boldsymbol{x}_j)$ between the value of p at any two selected points, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, could be calculated upon request (by some numerical procedure), the algorithm worked.

Monte Carlo methods are becoming increasingly important for the solution of nonlinear inverse problems in two different, but related, situations. In the first situation we need a near-optimal solution (measured in terms of data fit and adherence to given constraints) to the problem. In the second situation, the inverse problem is formulated as a search for solutions fitting the data within a certain tolerance, given by data uncertainties. In a non-probabilistic setting this means that we search for solutions with calculated data whose distance from the observed is less than a fixed, positive number. In a Bayesian context, the tolerance is 'soft': a large number of samples of statistically near-independent models from the a posterior probability distribution are sought. Such solutions are consistent with data and prior information, as they fit the data 'within error bars', and adhere to 'soft' *prior* constraints given by a prior probability distribution.

Early examples of the solution of inverse problems by means of Monte Carlo methods are abundant in geophysics and other disciplines of applied physics. Since Keilis-Borok and

Yanovskaya (1967) and (Press 1968, 1970a) made the first attempts at randomly exploring the space of possible Earth models consistent with seismological data, there has been considerable advances in computer technology, and therefore an increasing interest in these methods. Geman and Geman (1984) applied simulated annealing to Bayesian image restoration, and derived an expression for the posterior distribution from the prior distribution, a model of the image blurring mechanism, and a Gaussian noise model. Through an identification of the posterior distribution with a Gibbs–Boltzmann distribution, they estimated a maximum *a posteriori* model, using a simulated annealing algorithm. In their paper, they suggested using the Metropolis algorithm, not only for maximum *a posteriori* estimation, but also to sample the posterior distribution. This idea was taken up by Rothman (1985, 1986) who was the first to employ *importance sampling* to solve a strongly nonlinear (that is, a multi-modal) optimization problem arising in seismic reflection surveys. Later, Cary and Chapman (1988), Landa *et al* (1989), Mosegaard and Vestergaard (1991), and Koren *et al* (1991) applied Monte Carlo methods to seismic waveform fitting.

Cary and Chapman (1988) used the Monte Carlo method to analyse the seismic waveform inversion problem in a Bayesian formulation. Waveforms and travel times from source to receiver were used as data, and the model parameters defined a horizontally stratified Earth with depth as a function of the seismic wave propagation velocity. They improved efficiency of the sampling through a method described by Wiggins (1969, 1972) in which the model space was sampled according to a prior distribution $\rho(\boldsymbol{m})$.

Marroquin *et al* (1987) adopted an approach similar to that of Geman and Geman. However, they used the Metropolis algorithm to generate samples from the posterior distribution, from which they computed model estimates.

Recent examples of using Bayes theorem and the Metropolis algorithm for calculating approximate *a posteriori* probabilities for an inverse problem are given by Pedersen and Knudsen (1990), Koren *et al* (1991), Gouveia and Scales (1998), Dahl-Jensen *et al* (1998), Khan *et al* (2000), Rygaard-Hjalsted *et al* (2000), and Khan and Mosegaard (2001).

Not all Monte Carlo inversion adopted the Bayesian viewpoint. The initial introduction of Monte Carlo techniques into geophysics by Keilis-Borok and Yanovskaya (1967) and Press (1968, 1970a) was concerned with only uniform sampling of a parameter space, without taking a Bayesian approach. These papers generated considerable interest, and also a debate over how to interpret the resulting ensemble of Earth models, especially when extra constraints were imposed, Haddon and Bullen (1969), Anderssen (1970), Anderssen and Senata (1971, 1972), Anderssen *et al* (1972), Wiggins (1972), Kennett and Nolet (1978). In addition, uniform random search techniques have been used for model optimization in geophysical inversion (again without invoking Bayesian principles). Notable examples include: estimation of seismic attenuation structure in the Earth, Burton and Kennett (1974a, 1974b), Burton (1977); seismic surface wave attenuation studies, Biswas and Knopoff (1974), Mills and Fitch (1977); estimation of seismic and density structure, Worthington *et al* (1972, 1974), Goncz and Cleary (1976), Kennett (1998); magnetotelluric studies, Jones and Hutton (1979); estimation of mantle viscosity, Ricard *et al* (1989); and plate rotation vectors, Jestin *et al* (1994).

Most of the applications of genetic algorithms (Holland 1975) within geophysical inverse problems are also non-Bayesian. That is, they are Monte Carlo techniques that generate an ensemble of samples from a parameter space, which must then be made use of in some way. The introduction of genetic algorithms into geophysics occurred in the early 1990s (Stoffa and Sen 1991, Gallagher *et al* 1991, Wilson and Vasudevan 1991, Sambridge and Drijkoningen 1992, Scales *et al* 1992, Sen and Stoffa 1992, Smith *et al* 1992). They have since been applied to a wide range of geophysical problems. (Many references on geophysical applications can be found in Gallagher and Sambridge (1994) and Sambridge and Mosegaard (2001).)

More recently a new class of ensemble-based Monte Carlo search technique known as a neighbourhood algorithm (Sambridge 1999a) has been developed for geophysical inverse problems. This approach follows a non-Bayesian approach for sampling of a parameter space, but can be used within a Bayesian formulation for analysing the resulting ensemble (Sambridge 1999b).

Monte Carlo methods are essentially practical tools for dealing with (usually complicated) probability distributions, and this is the main reason for their usefulness in inverse problem analysis. Solutions $\boldsymbol{m}$ to inverse problems can usually be described by a function $f(\boldsymbol{m})$ over the model space, measuring the model's ability to fit the data and/or given *a priori* constraints. In this paper, we will call $f(\boldsymbol{m})$ the *fitness function*, a term borrowed from the theory of genetic algorithms.

In Bayesian inversion the fitness function is the so-called posterior probability distribution given by

$$f(\boldsymbol{m}) = C_f L(\boldsymbol{m})\rho(\boldsymbol{m}) \tag{4}$$

where C_f is a normalization constant, $L(\boldsymbol{m})$ is a likelihood function and $\rho(\boldsymbol{m})$ a prior probability distribution. The likelihood function is usually of the form

$$L(\boldsymbol{m}) = C_L \exp(-S(\boldsymbol{m})) \tag{5}$$

where C_L is a constant, and $S(\boldsymbol{m})$ is a *misfit function*, measuring the deviation of the observed data from the data calculated from $\boldsymbol{m}$. The prior distribution assigns a data-independent weight to $\boldsymbol{m}$ depending on how acceptable it is according to other available information.

In a non-probabilistic context, the function may be an indicator function describing which models are acceptable according to data, and which are not. For instance, for a given positive constant S we may define

$$f(\boldsymbol{m}) = \begin{cases} 1 & \text{if } S(\boldsymbol{m}) \leqslant S \\ 0 & \text{if } S(\boldsymbol{m}) > S. \end{cases} \tag{6}$$

As mentioned in the introduction, another way of using Monte Carlo methods for analysis of inverse problems is for *model construction*, that is, finding a set of model parameters that, in some sense, gives a (near-) optimal fit to data and prior information. Model construction is an optimization problem, which in a Bayesian context can be formulated as a search for the model(s) where $f(\boldsymbol{m})$ attains its maximum. In a non-probabilistic context, the problem can be formulated as a search for the minimum of the misfit function $S(\boldsymbol{m})$.

Theoretically, the model construction problem can be viewed as a special case of the more general sampling problem: sampling, e.g., a likelihood function (5), where we have artificially decreased the standard deviation of the data uncertainties, corresponds to minimizing the misfit $S(\boldsymbol{m})$. We shall therefore begin the next section with a review of the essential Markov chain sampling theory. Later we will specialize this to cover the model construction problem. The heuristic methods, simulated annealing and genetic algorithms, designed to improve the speed of model construction will then be covered. This order of presentation is not in accordance with the historic development, but it reveals some of the fundamental strengths and weaknesses of the Monte Carlo methods, and hopefully points in directions where future research efforts should be made.

2. Monte Carlo methodology

Monte Carlo methods are natural when solving inverse problems within a purely probabilistic framework. The reason for this is the following: the fundamental building blocks of the

theory are probability densities, and these objects can be viewed mathematically as limits of histograms (normalized, and with vanishing column widths) generated by a random process, e.g., a Monte Carlo algorithm. For this reason, any probability distribution can be represented by a group of Monte Carlo algorithms, namely the Monte Carlo algorithms that sample it. On the other hand, to every Monte Carlo algorithm there is a probability distribution, namely the one that it samples. Most conceivable operations on probability densities (e.g., computing marginals and conditionals, integration, combining independent information, etc) have their counterparts in operations on/by their corresponding Monte Carlo algorithms. In this way, Monte Carlo algorithms provide a way of manipulating probability densities—even densities that cannot be expressed mathematically in closed form.

Our first task will be to describe which distribution is sampled by a given Monte Carlo algorithm, and how a Monte Carlo algorithm can be made to sample a prescribed probability distribution.

2.1. Sampling known one-dimensional probability distributions

If random rules have been defined to select points such that the probability of selecting a point in the volume element $\mathrm{d}x_1 \ldots \mathrm{d}x_N$ is $p(\boldsymbol{x})\mathrm{d}x_1 \ldots \mathrm{d}x_N$, then the points selected in that way are called *samples* of the distribution $p(\boldsymbol{x})$. Depending on the rules defined, successive samples $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \ldots$, may be dependent or independent.

Before going into more complex sampling situations, let us briefly review a few methods for sampling a probability distribution that is 'completely known' in the sense that it can be described by an explicit mathematical expression. Two important methods are given below (formulated for a probability distribution over a one-dimensional space).

Method 1. Let p be an everywhere nonzero probability distribution with distribution function P, given by

$$P(s) = \int_{-\infty}^{x} p(s)\,\mathrm{d}s, \tag{7}$$

and let r be a random number chosen uniformly at random between 0 and 1. Then the random number x generated through the formulae

$$x = P^{-1}(r) \tag{8}$$

has probability distribution p.

More special, yet useful, is the following way of generating Gaussian random numbers.

Method 2. Let r_1 and r_2 be random numbers chosen uniformly at random between 0 and 1. Then the random numbers x_1 and x_1 generated through the formulae

$$x_1 = \sqrt{-2\ln r_2}\cos(2\pi r_1) \tag{9}$$

$$x_2 = \sqrt{-2\ln r_2}\sin(2\pi r_1) \tag{10}$$

are independent and Gaussian distributed with zero mean and unit variance.

These theorems are easily demonstrated, and straightforward to use in practice. They can be extended to higher dimensions.

2.2. Sampling 'unknown' probability distributions

For most practical inverse problems, the model space is so vast, and evaluation of the fitness function $f(\boldsymbol{x})$ is so computer intensive, that only algorithms which evaluate $f(\boldsymbol{x})$ once (or only a few times) in each iteration, are useful. In such cases we will say that, for the algorithm, values of $f(\boldsymbol{x})$ 'are only available on request'.

If $p(\boldsymbol{x})$ is a multi-dimensional probability distribution over $\mathcal{X}$ whose values are only available on request, it is still straightforward to design a random walk that samples it. Imagine, for instance, that we have a number M satisfying

$$M \geqslant \max(p(\boldsymbol{x})). \tag{11}$$

Then, the following algorithm will sample $p(\boldsymbol{x})$.

Algorithm 1 (the primitive algorithm). *In the nth step of the algorithm, choose a candidate sampling point $\boldsymbol{x}_{cand,n}$ randomly (using a uniform distribution over the sampling space) but accept it only with probability*

$$p_{accept} = \frac{p(\boldsymbol{x}_{cand,n})}{M}. \tag{12}$$

The set of thus accepted candidate points are samples from the probability distribution $p(\boldsymbol{x})$.

However, this algorithm would not be useful in practice, because in many problems $p(\boldsymbol{x})$ could have narrow maxima, (which contribute significantly to integrals over $p(\boldsymbol{x})$), but are sampled rarely (or not at all) when only a limited number of samples can be taken. One of the things that slows down the primitive algorithm is that the ratio

$$\frac{p(\boldsymbol{x})}{M} \tag{13}$$

is very small almost everywhere in $\mathcal{X}$, especially when $p(\boldsymbol{x})$ has large values only in a small part of the space $\mathcal{X}$, and hence the likelihood of acceptance is very small.

The *Metropolis algorithm*, an example of an *importance sampling algorithm*, works around this problem by comparing $p(\boldsymbol{x}_{cand,n})$, not with a large number M, but with the smaller number $p(\boldsymbol{x}_{current,n})$, where $\boldsymbol{x}_{current,n}$ is the current point being visited. The probability of accepting the candidate point $\boldsymbol{x}_{cand,n}$ in the Metropolis algorithm is

$$p_{accept} = \begin{cases} \dfrac{p(\boldsymbol{x}_{cand,n})}{p(\boldsymbol{x}_{current,n})} & \text{when } p(\boldsymbol{x}_{cand,n}) \leqslant p(\boldsymbol{x}_{current,n}) \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

Furthermore, in this algorithm $\boldsymbol{x}_{cand,n}$ is often chosen from a relatively small neighbourhood around $\boldsymbol{x}_{current,n}$ so as to further reduce the chance that the ratio $p(\boldsymbol{x}_{cand,n})/p(\boldsymbol{x}_{current,n})$ is small.

Importance sampling allows us to sample the space with a sampling density proportional to the given probability density, without excessive (and useless) sampling of low-probability areas of the space. This is not only important, but in fact vital in high-dimensional spaces, where a very large proportion (approaching 100%) of the volume may have near-zero probability density. Figure 1 illustrates the superiority of the Metropolis algorithm over the primitive algorithm, even in a one-dimensional example.

3. Sampling through random walks

In the following, we shall describe the essential properties of random walks performing *importance sampling*.

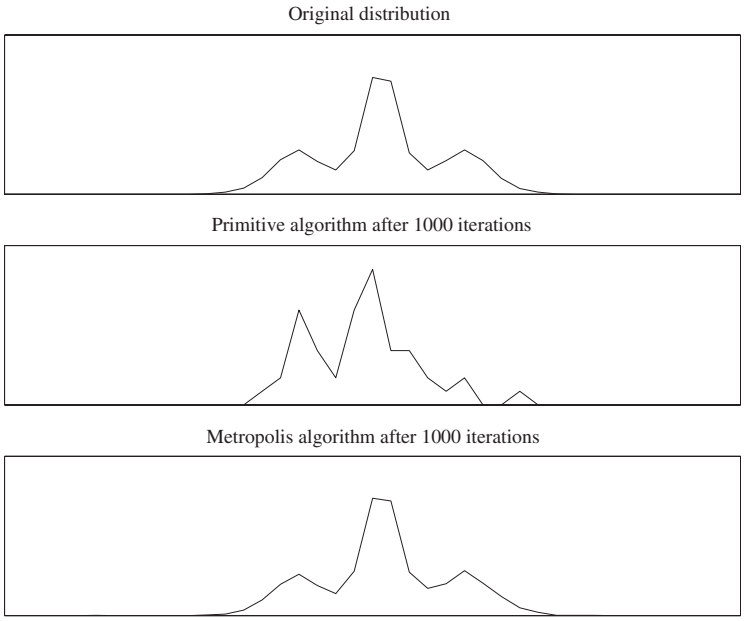

Figure 1. A probability distribution (top) were sampled 1000 times by the primitive algorithm (middle) and the Metropolis algorithm (bottom). The superiority of the Metropolis algorithm over the primitive algorithm is seen even in this one-dimensional example. In the Metropolis case all 1000 iterations yielded a sample contributing to the final result, whereas for the primitive algorithm, only 42 samples were accepted for histogram building.

A random walk is a Markov chain. This means that the probability of moving from a point $\boldsymbol{x}_i$ to a point $\boldsymbol{x}_j$ in the space $\mathcal{X}$ in a given step (iteration) is independent of the path travelled by the 'walker' in the past. Let us define the conditional probability distribution $P_{ij}(\boldsymbol{x}_i|\boldsymbol{x}_j)$ of visiting $\boldsymbol{x}_i$, given that the walker currently is at point $\boldsymbol{x}_j$. We will call $P_{ij}(\boldsymbol{x}_i|\boldsymbol{x}_j)$ the *transition probability distribution*, and for simplicity in the following we will drop the double subscript and write $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$. P is a probability distribution with respect to its first argument, so we have

$$\int_{\mathcal{X}} P(\boldsymbol{x}_i|\boldsymbol{x}_j)\,\mathrm{d}\boldsymbol{x}_i = 1. \tag{15}$$

As a special case, P may be a discrete distribution, in which case we imagine the set of points $\boldsymbol{x}_j$ $(i = 1, \ldots, J)$ to be a discrete, finite subset of $\mathcal{X}$. The random walk we are about to construct here will then only be allowed to visit points in the considered grid, and $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$ will be a discrete distribution over the grid points.

Given a random walk defined by the transition probability distribution $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$. Assume now that we have a distribution $s_n(\boldsymbol{x})$ describing the position of the random walker after n steps. Each step of the random walk will modify this distribution, and if $s_n(\boldsymbol{x}) \to p(\boldsymbol{x})$ for $n \to \infty$ we say that $p(\boldsymbol{x})$ is an *equilibrium (or stable) probability distribution* for the random walk. That $p(\boldsymbol{x})$ is an equilibrium distribution means that if it is the distribution of the position of the random walker at one time, it remains the distribution of its position after one more step (and hence, forever). Technically, this can be expressed by the fact that $p(\boldsymbol{x})$ is an eigenfunction with eigenvalue 1 of the linear integral operator with kernel $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$:

$$\int_{\mathcal{M}} P(\boldsymbol{x}_i|\boldsymbol{x}_j)\,p(\boldsymbol{x}_j)\,\mathrm{d}\boldsymbol{x}_j = p(\boldsymbol{x}_i). \tag{16}$$

If the random walk, for any initial distribution $s_0(\boldsymbol{x})$, equilibrates to the same distribution $p(\boldsymbol{x})$, we say that $p(\boldsymbol{x})$ is *the* (unique) equilibrium distribution for $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$.

The next section describes the basic idea behind importance sampling by the Metropolis algorithm, which is designed to have any chosen function $p(\boldsymbol{x})$ as its unique equilibrium distribution.

3.1. Design of random walks with given equilibrium distributions

A random walk with a given equilibrium distribution $p(\boldsymbol{x})$ must satisfy the condition that once the sampling distribution $s(\boldsymbol{x})$ is equal to $p(\boldsymbol{x})$ it must remain equal to $p(\boldsymbol{x})$. This equilibrium can also be expressed through the following condition.

Condition 1 (microscopic reversibility). *The probability, at any time, that the random walker enters an infinitesimal neighbourhood $\mathcal{N}_j$, surrounding the point $\boldsymbol{x}_j$, equals the probability that it leaves $\mathcal{N}_j$.*

There are infinitely many ways of satisfying the above requirement. As we shall see in a later section, the way a genetic algorithm establishes microscopic reversibility (equilibrating to its unknown stable distribution) is rather complex. In contrast, the Metropolis algorithm and the Gibbs sampler rely on a very simple principle, summarized in the following condition.

Condition 2 (detailed balance). *For any pair of points $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$, the probability, at any time, that the random walker jumps from the infinitesimal neighbourhood $\mathcal{N}_j$, surrounding $\boldsymbol{x}_j$, to the infinitesimal neighbourhood $\mathcal{N}_i$ (of the same volume), surrounding $\boldsymbol{x}_i$, equals the probability that it jumps from $\mathcal{N}_i$ to $\mathcal{N}_j$.*

From this last principle, an algorithm that has a given distribution $p(\boldsymbol{x})$ as an equilibrium distribution can easily be derived.

Consider two points in $\mathcal{X}$, say $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$. Detailed balance means that the transition probability distribution satisfies the following symmetry condition:

$$P(\boldsymbol{x}_i|\boldsymbol{x}_j)\,p(\boldsymbol{x}_j)\,\mathrm{d}\boldsymbol{x}_j\,\mathrm{d}\boldsymbol{x}_i = P(\boldsymbol{x}_j|\boldsymbol{x}_i)\,p(\boldsymbol{x}_i)\,\mathrm{d}\boldsymbol{x}_i\,\mathrm{d}\boldsymbol{x}_j \tag{17}$$

where $p(\boldsymbol{x})$ is the desired equilibrium distribution, which we will, without loss of generality, assume to be nonzero everywhere in $\mathcal{X}$.

Equation (17) means that $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$ must satisfy

$$\frac{P(\boldsymbol{x}_i|\boldsymbol{x}_j)}{P(\boldsymbol{x}_j|\boldsymbol{x}_i)} = \frac{p(\boldsymbol{x}_i)}{p(\boldsymbol{x}_j)}. \tag{18}$$

There are, of course, infinitely many solutions $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$ to the above equation, but we shall try to design an efficient algorithm by keeping the transition probabilities everywhere as large as possible between pairs of points.

The first question is now: how can we implement an algorithm with transition probability densities satisfying (18)? The answer to this question is interesting, as it reveals two important, and conflicting, characteristics of random-walk-based algorithms.

In the so-called Metropolis algorithm, (18) is implemented through two randomized operations which, together, form an iteration:

(i) The exploration step. The first operation consists of proposing a 'candidate' point $\boldsymbol{x}_i$ using a so-called *proposal distribution* $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$, where $\boldsymbol{x}_j$ is the currently visited point. The proposal distribution is symmetric:

$$U(\boldsymbol{x}_i|\boldsymbol{x}_j) = U(\boldsymbol{x}_j|\boldsymbol{x}_i), \tag{19}$$

but otherwise it is arbitrary, in the sense that its form is chosen before running the algorithm, and is, in principle, independent of any knowledge on $p(\boldsymbol{x})$. $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ embodies the 'strategy' by which the algorithm explores $\mathcal{X}$, when searching for new samples from the distribution $p(\boldsymbol{x})$.

(ii) The exploitation step. The second operation is to decide if the candidate point should be accepted as the next sample. Any acceptance probability of the form

$$p_{accept} = \psi(\boldsymbol{x}_i, \boldsymbol{x}_j)/p(\boldsymbol{x}_j), \tag{20}$$

where $\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a symmetric function, can be used. The simplest acceptance probability (and the one giving the highest transition probabilities for given $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$) is obtained by putting $\psi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \min(p(\boldsymbol{x}_i), p(\boldsymbol{x}_j))$. This gives the traditional Metropolis acceptance probability

$$p_{accept} = \begin{cases} \dfrac{p(\boldsymbol{x}_i)}{p(\boldsymbol{x}_j)} & \text{for } p(\boldsymbol{x}_i) \leqslant p(\boldsymbol{x}_j) \\ 1 & \text{otherwise.} \end{cases} \tag{21}$$

If the candidate point is rejected, the current point is repeated (thus counting one more time). The acceptance probability describes the 'greediness' of the algorithm. The smaller p_{accept} is for $p(\boldsymbol{x}_i) \leqslant p(\boldsymbol{x}_j)$, the more 'greedy' the algorithm is.

The above procedure means that

$$P(\boldsymbol{x}_i|\boldsymbol{x}_j) = U(\boldsymbol{x}_i|\boldsymbol{x}_j)\, p_{accept} \tag{22}$$

giving

$$\frac{P(\boldsymbol{x}_i|\boldsymbol{x}_j)}{P(\boldsymbol{x}_j|\boldsymbol{x}_i)} = \frac{U(\boldsymbol{x}_i|\boldsymbol{x}_j)}{U(\boldsymbol{x}_j|\boldsymbol{x}_i)} \frac{\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\psi(\boldsymbol{x}_j, \boldsymbol{x}_i)} \frac{p(\boldsymbol{x}_i)}{p(\boldsymbol{x}_j)}. \tag{23}$$

Due to the imposed symmetry (19) on $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ and $\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)$, equation (23) is the detailed balance condition (18).

It can be shown (see, e.g., Kaipio *et al* 2000) that if our transition probability distribution $P(\boldsymbol{x}_i|\boldsymbol{x}_j)$, satisfies two particular conditions (in addition to microscopic reversibility), then $p(\boldsymbol{x})$ will be the only equilibrium distribution for the algorithm, and so it will converge towards $p(\boldsymbol{x})$ no matter what the starting distribution is. The two conditions are:

(i) Aperiodicity. The probability that an iteration of the algorithm results in the trivial move $\boldsymbol{x}_j \longrightarrow \boldsymbol{x}_j$ is non-zero. That this is clearly satisfied, can easily be seen from the form (21) of the acceptance probability.

(ii) Irreducibility. It is possible to go from any point $\boldsymbol{x}_j$ to any other point $\boldsymbol{x}_i$ in $\mathcal{X}$, given a sufficient number of iterations. It is up to the designer of the proposal distribution to make sure that this requirement is satisfied.

The majority of Monte Carlo methods used in practice satisfy the above two conditions, and therefore converge toward (equilibrate at) a unique distribution. In this paper we describe three algorithms belonging to this category: the *Metropolis algorithm* (described above), the *Gibbs sampler* and the *genetic algorithm*. The first two algorithms equilibrate at *known* distributions (and they are, in fact, designed to equilibrate at these distributions), whereas the equilibrium distribution of the genetic algorithm, defined in terms of the parameters of the algorithm, has not yet been found.

Convergence issues. The second important question is now: How do we maximize the efficiency of our algorithm? Unfortunately, a definitive answer to this question has not yet been found, although it is of utmost importance for the practical applicability of the Metropolis algorithm. The proposal distribution $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ embodies the complete search strategy of the algorithm, so the problem of efficiency can, e.g., be formulated in the following way.

Given a measure of distance $\mathrm{dist}(p_1, p_2)$ in the space of probability densities over $\mathcal{X}$, and a (usually small) positive number M. Find an irreducible proposal distribution $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ that minimizes the expected number of iterations needed to obtain

$$\mathrm{dist}(s, p) \leqslant M, \tag{24}$$

where $s(\boldsymbol{x})$ is the sampling distribution and $p(\boldsymbol{x})$ is the equilibrium distribution. A possible distance measure can, e.g., be

$$\mathrm{dist}(p_1, p_2) = \max_{\boldsymbol{x}\in\mathcal{X}} |p_1(\boldsymbol{x}) - p_2(\boldsymbol{x})|. \tag{25}$$

In the special case where $\mathcal{X}$ is a discrete space with a relatively small number of points $\boldsymbol{x}_j$, and the algorithm can be described by a completely known transition probability matrix $\boldsymbol{P} = \{P(\boldsymbol{x}_i|\boldsymbol{x}_j)\}$, the convergence speed may be estimated through an eigenvalue analysis of P_{ij}. However, this situation virtually never occurs in practice. Knowing $\boldsymbol{P}$ completely would require knowing $p(\boldsymbol{x}_j)$ for all j, and in that case the inverse problem is already solved!

Andresen *et al* (1988) showed that an approximate eigenvalue analysis can be made by lumping points of similar values of $p(\boldsymbol{x})$ into a small number of 'states' between which transition probabilities can be estimated empirically. For a given choice of $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ an initial run of the algorithm is used to monitor the frequency of transitions between the lumped states. The matrix of (normalized) transition frequencies is then used as an approximation to the transition probability matrix for transitions between lumped states. Finally, the second largest eigenvalue of this transition probability matrix is used to estimate the *relaxation time* for the algorithm (the time taken by the algorithm to reduce the distance between its sampling distribution $s(\boldsymbol{x})$ and its equilibrium distribution $p(\boldsymbol{x})$ by a factor $1/e$). The relaxation time is related to the convergence time we are looking for, although the authors do not explicitly explore this relationship.

Lacking theoretical guidance, it has become common practice to tune $U(\boldsymbol{x}_i|\boldsymbol{x}_j)$ empirically (Hastings 1970). An acceptable proposal distribution must keep the so-called *burn-in period* for the algorithm at a minimum. The burn-in period is the time it takes for the algorithm, from its initial state, before it reaches a point where its outputs (probability densities of sampled models, parameters of sampled models, frequency of accepted models, etc) are approximately stationary over the considered number of iterations. Experience has shown that a frequency of accepted models (after the burn-in period) of 25–50% indicates that the algorithm is performing well (Gelman *et al* 1996).

Multistep iterations. Often, it is convenient to split up an iteration into a number of substeps, having their own transition probability densities. A typical example is a random walk in an N-dimensional space where we are interested in dividing an iteration of the random walk into N substeps, where the nth move of the random walker is in the direction parallel to the nth axis.

The question is now: if we want to form an iteration consisting of a series of substeps, can we give a sufficient condition to be satisfied by each substep such that the complete iteration has the desired convergence properties?

It is easy to see that if the individual substeps in an iteration all have the same distribution $p(\boldsymbol{x})$ as their equilibrium distribution (not necessarily unique), then the complete iteration also has $p(\boldsymbol{x})$ as an equilibrium distribution. This follows from the fact that the equilibrium

distribution is an eigenfunction with eigenvalue 1 for the integral operators corresponding to each of the substep transition probability distributions. Then it is also an eigenfunction with eigenvalue 1, and hence an equilibrium distribution, for the integral operator corresponding to the transition probability distribution for the complete iteration.

If this transition probability distribution is to be the unique equilibrium distribution for the complete iteration, then the random walk must be irreducible. That is, it must be possible to go from any point to any other point by performing iterations consisting of the specified substeps.

If the substeps of an iteration satisfy these sufficient conditions, there is also another way of defining an iteration with the desired, unique equilibrium distribution. Instead of performing an iteration as a series of substeps, it is possible to define the iteration as consisting of one of the substeps, chosen randomly (with any distribution having nonzero probabilities) among the possible substeps. In this case, the transition probability distribution for the iteration is equal to a linear combination of the transition probability densities for the individual substeps. The coefficient of the transition probability distribution for a given substep is the probability that this substep is selected. Since the desired distribution is an equilibrium distribution (eigenfunction with eigenvalue 1) for the integral operators corresponding to each of the substep transition probability distribution, and since the sum of all the coefficients in the linear combination is equal to 1, it is also an equilibrium distribution for the integral operator corresponding to the transition probability distribution for the complete iteration. This equilibrium distribution is unique, since it is possible, following the given substeps, to go from any point to any other point in the space.

Of course, a substep of an iteration can, in the same way, be built from sub-substeps, and in this way acquire the same (not necessarily unique) equilibrium distribution as the sub-substeps.

The Gibbs sampler. In the Gibbs sampler, one iteration consists of a number of substeps, each having its own transition probability distribution. In a typical implementation of the Gibbs sampler, operating in a K-dimensional model space, each iteration consists of K substeps, one for each parameter. The kth substep perturbs only the kth parameter, and it has its own transition probability distribution $P_k(\boldsymbol{x}_i|\boldsymbol{x}_j)$. The kth substep runs as follows:

(i) The proposal distribution is defined as

$$U_k(\boldsymbol{x}_i|\boldsymbol{x}_j) = \begin{cases} \dfrac{p(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_k \in \mathcal{N}_j^k} p(\boldsymbol{x}_k)} & \boldsymbol{x}_i \in \mathcal{N}_j^k \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

where $\mathcal{N}_j^k$ is the set of points deviating from $\boldsymbol{x}_j$ only in the kth parameter.

(ii) The acceptance probability is

$$p_{accept} = 1 \tag{27}$$

for all the proposed candidate models.

In each substep of the Gibbs sampler, no more than one parameter is perturbed (or is possibly left unchanged), so after completion of one iteration (consisting of all K substeps), all parameters have been perturbed.

That this algorithm samples p can be seen in the following way: the transition probability distribution for each substep is given by

$$P_k(\boldsymbol{x}_i|\boldsymbol{x}_j) = U_k(\boldsymbol{x}_i|\boldsymbol{x}_j), \tag{28}$$

which satisfies detailed balance equation (18), since if $\boldsymbol{x}_i \in \mathcal{N}_j^k$, then

$$P_k(\boldsymbol{x}_i|\boldsymbol{x}_j)\,p(\boldsymbol{x}_j) = U_k(\boldsymbol{x}_i|\boldsymbol{x}_j)\,p(\boldsymbol{x}_j) \tag{29}$$

$$= \frac{p(\boldsymbol{x}_i)}{\sum_{\boldsymbol{x}_k \in \mathcal{N}_j^k} p(\boldsymbol{x}_k)}\,p(\boldsymbol{x}_j) \tag{30}$$

$$= \frac{p(\boldsymbol{x}_j)}{\sum_{\boldsymbol{x}_k \in \mathcal{N}_i^k} p(\boldsymbol{x}_k)}\,p(\boldsymbol{x}_i) \tag{31}$$

$$= U_k(\boldsymbol{x}_j|\boldsymbol{x}_i)\,p(\boldsymbol{x}_i) \tag{32}$$

$$= P_k(\boldsymbol{x}_j|\boldsymbol{x}_i)\,p(\boldsymbol{x}_i) \tag{33}$$

where we have used that $\mathcal{N}_j^k = \mathcal{N}_i^k$. Since each substep of an iteration satisfies microscopic reversibility, so does the entire iteration, and the algorithm samples the target distribution p asymptotically.

The advantage of the Gibbs sampler is that the acceptance probability is always 1, so there are no rejected moves. The disadvantage lies in the construction of the proposal distribution. In contrast to the Metropolis algorithm, $U_k(\boldsymbol{x}_i|\boldsymbol{x}_j)$ is here constructed directly from the desired equilibrium distribution by evaluating $p(\boldsymbol{x})$ 'along a line' in $\mathcal{X}$. This is feasible when calculation of $p(\boldsymbol{x})$ is computationally inexpensive, but this is not the case in many practical inverse problems. When evaluation of $p(\boldsymbol{x})$ is computer intensive, the Metropolis algorithm may be a better choice. An example of the use of a Gibbs sampler in a case where p can be efficiently evaluated for all perturbations of a single model parameter can be found in Rothman (1986).

3.2. Bayesian inference

We mentioned in the introduction that in a Bayesian formulation, the fitness function is the so-called posterior probability distribution over the model space, given by

$$f(\boldsymbol{m}) = C_f L(\boldsymbol{m})\rho(\boldsymbol{m}).$$

This distribution carries all information available on models originating from the data, and from data-independent prior information.

If a Monte Carlo algorithm (typically the Metropolis algorithm or a Gibbs sampler) is used to generate a large number of samples $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N$ from $f(\boldsymbol{m})$, we can use these samples to estimate averages over the model space. Any average of a function $h(\boldsymbol{m})$ over the model space $\mathcal{M}$ (e.g., an expectation or a covariance) can be approximated by (3). The mean $\langle m_i \rangle$ of the ith model parameter m_i can be estimated by putting $h(\boldsymbol{m}) = m_i$, and the posterior covariance between the ith and jth model parameters is approximated by putting $h(\boldsymbol{m}) = (m_i - \langle m_i \rangle)(m_j - \langle m_j \rangle)$. If we wish to calculate the probability of an event $\mathcal{E}$ in $\mathcal{M}$, containing all models in model space with a given feature, it is done by putting $h(\boldsymbol{m})$ equal to the indicator function

$$h(\boldsymbol{m}) = \begin{cases} 1 & \text{if } \boldsymbol{m} \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

As an example, $\mathcal{E}$ may be all sampled models of the Earth containing a sharp boundary (appropriately defined) in a certain depth interval. Finally, it should be mentioned that samples from the one-dimensional marginal $f(m_k)$ are obtained simply by collecting values of m_k from samples $\boldsymbol{m} = (m_1, \ldots, m_k, \ldots, m_M)$ of $f(\boldsymbol{m})$.

The above procedure is general and simple, but the practical problem is often to *discover* model features that have a high probability. In classical terminology, these features are *well*

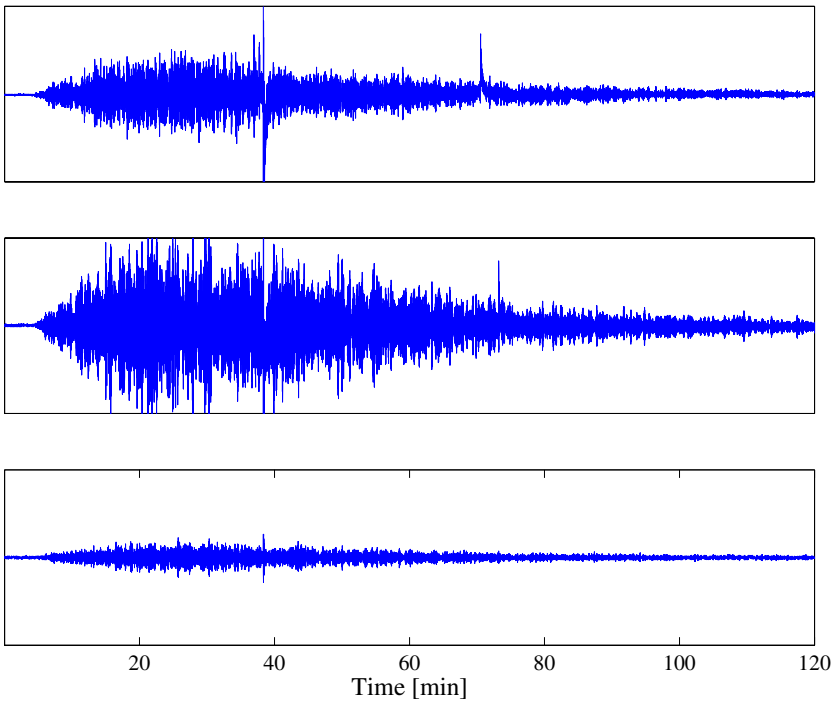

Figure 2. This figure shows a Lunar three-component seismic record from a meteoroid impact recorded at station 16 on day 319, 1976. The seismogram at the top is the N–S component (S positive), the middle is E–W (W positive) and the seismogram at the bottom is the vertical component (up positive). The seismograms commence at 23 h 16 min 50 s.

resolved. Simply looking at the Monte Carlo-generated samples from $f(\boldsymbol{m})$ is often the most efficient way to discover such structure. Well-resolved structure is seen as a structure that occurs frequently in the output, and the simplest way to discover it, is to plot all output models side by side, or on top of each other. Another way is to display all output models sequentially as pictures in a movie, preferably with different degrees of smoothing (Mosegaard and Tarantola 1995).

3.3. Example: inversion of seismic travel-time data from the Moon

Khan *et al* (2000) used the Metropolis algorithm to reanalyse the Lunar seismic data from the Apollo project, 1969–1977. The problem was to infer Lunar P- and S-wave velocity structure from observed arrival times of seismic disturbances, generated by moonquakes, meteorite impacts and artificial impacts (spacecraft modules hitting the Moon surface). Figure 2 shows a sample Lunar three-component seismic record from a deep moonquake, recorded on February 2, 1970. First arrival times for the P- and S-waves were read from 177 seismograms, and a spherically symmetric P- and S-wave velocity model, as well as source locations and epicentre times (time of seismic energy release), were sought to explain this dataset. The P- and S-wave velocity models were assumed to be piecewise linear functions of radius, with 56 break points each (the Lunar radius is 1738 km). Depths and velocities at these break points were used as the unknown model parameters of the problem, together with the source locations and epicentre times for 80 seismic events. All in all, the total number of unknown parameters for this problem was 450.

The *a priori* probability density was defined as follows: each velocity parameter was allowed to vary in the interval between 1 and 50 km s^{-1}, its logarithm assigning a uniform distribution between these values. Ray theory requires a certain smoothness of the model, which was realized by assuming a minimum layer thickness (distance between two consecutive break points) of 5 km. Source coordinates were unconstrained.

This problem has the typical 'pathology' that no analytical relation between data and model parameters exists. Only a rather computer intensive algorithm, tracing rays through any given P- and S-velocity model, is available, so Monte Carlo analysis is the most suitable method for this problem.

The Metropolis algorithm used in this case updated one velocity (or source-) parameter at a time. The likelihood function was given by (5), where

$$S(\boldsymbol{m}) = \sum_n \frac{|d_{obs}^{(n)} - d^{(n)}(\boldsymbol{m})|}{\sigma_n}$$

$d_{obs}^{(n)}$ denoting the observed data (travel times), $d^{(n)}(\boldsymbol{m})$ synthetic data, computed (by ray tracing) from the model $\boldsymbol{m}$, and σ_n is the uncertainty (standard deviation) of the nth datum. The uncertainty was 1 s for artificial impacts, between 4 and 7 s for deep moonquakes, and between 4 and 26 s for shallow moonquakes and meteorite impacts.

The proposal distribution was defined as follows: in each iteration, a new model parameter was chosen at random. It was then perturbed using a uniform distribution centred at the current value, and having a half-width of 1.1 km s^{-1} for velocities, 2 km for the upper 11 layer boundaries, 8 km for the lower 45 layer boundaries, 1° for source longitudes/latitudes and 4 km for source depths. Layer boundary perturbations were, however, modified so as to maintain the ordering of boundaries. This was done by appropriate reduction of the half-width of the distribution of proposed boundary depths.

The chosen proposal distribution gives a burn-in time of approximately 1000 iterations (out of 1 370 000 iterations in total) and a subsequent average frequency of acceptance of about 40–50%. Marginal *a posteriori* frequency distributions of P- and S-wave velocities are shown in figure 3. A certain layering of the Lunar velocity structure is revealed, of which an 'upper mantle' thickness of about 560 km and a crustal thickness of approximately 45 km were the most surprising features (see Khan *et al* (2000) and Hood and Zuber (2000), for a further discussion of these results). However, it is evident from the figure that the marginal uncertainties are large, and that the output should be interpreted with great care.

Another instructive example of Monte Carlo analysis of an inverse problem, this time from electrical impedance tomography, can be found in Kaipio *et al* (2000). This paper also gives further details on the theory of Monte Carlo methods.

3.4. Random walks in non-Bayesian ensemble inference

Although random walks are at the heart of every Monte Carlo algorithm, they are not only useful within a Bayesian formulation of an inverse problem. Many authors have used random walks to sample parameter spaces without defining a posterior probability distribution (4). As mentioned above, the earliest Monte Carlo algorithms used in geophysics were non-Bayesian. An example is the work of Press (1978, 1970a), which consisted of a series of nested uniform random searches in parameter space. A convenient way to view non-Bayesian ensemble inference is in terms of a two-stage approach, consisting of a search stage and an appraisal stage.

In the search stage, an algorithm (based on random walks) is used to collect samples, and the predictions of these models are compared to the data. In many cases the search

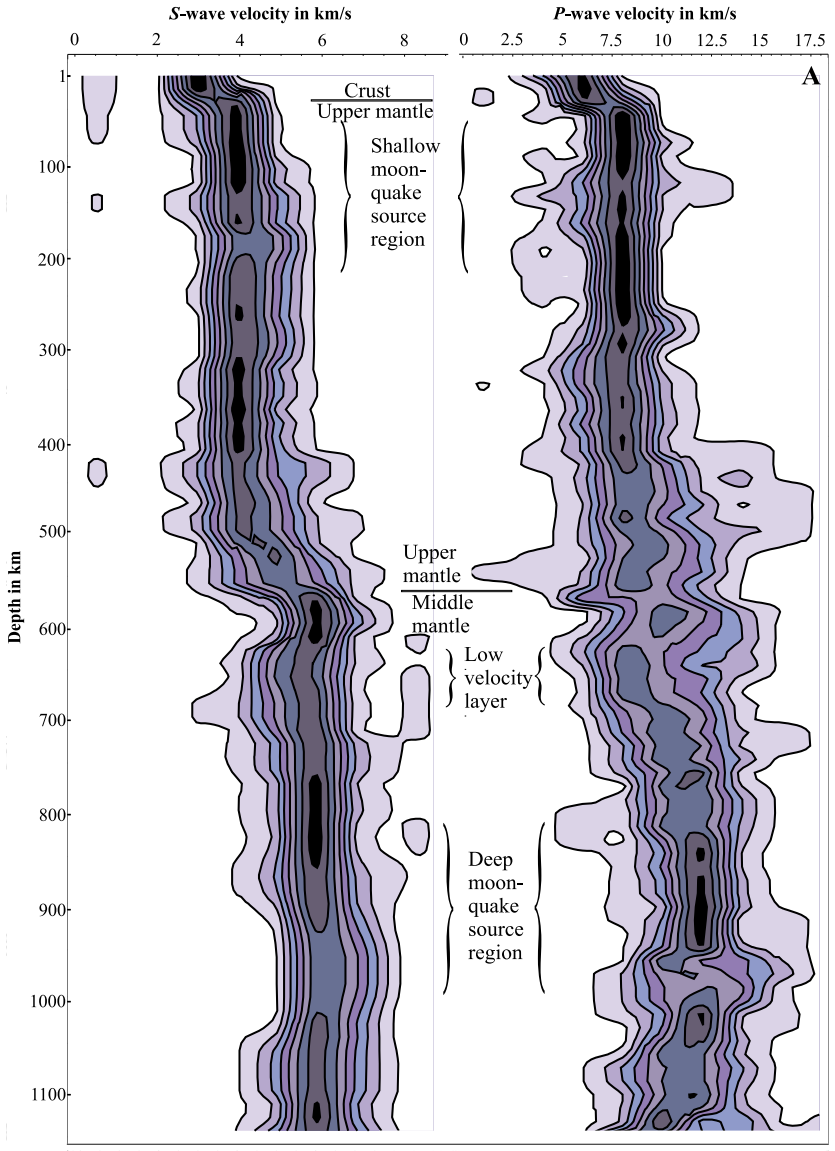

Figure 3. Marginal posterior velocity distributions for the velocity structure of the Moon. 50 000 models were used in constructing the two results. For each kilometre in depth, a histogram reflecting the marginal *a posteriori* probability distribution of sampled velocities has been computed. These marginals are lined up, and contour lines define nine equal-sized probability density intervals for the distributions.

process is adaptive, i.e. it makes use of the samples collected to guide the search for new models. Clearly, there is a strong connection with optimization problems, and indeed many direct search (i.e. non-derivative based) optimization algorithms have been used as search algorithms in inverse problems. Examples include simulated annealing, genetic algorithms and evolutionary programming techniques, and the neighbourhood algorithm (see below). Since many of these overlap with the area of global optimization we defer a discussion of them to the next section.

In the appraisal stage the objective is to make use of the complete ensemble of parameter space samples to draw inferences from the data. If an optimization algorithm has been used in the search stage to maximize a likelihood function (5), then the temptation is often to select the best data fitting model only and examine it in detail. However, this is almost always insufficient because of the noise in the data and the non-uniqueness of the underlying inverse problem. Even within a finite-dimensional parameter space one usually finds that if one model fits the data to an acceptable level (taking into account noise in the data), then an infinite number will, and so the best data fit model may well be mis-leading. (Just as in a Bayesian formulation the maximum of the posterior is seldom representative of the overall ensemble.)

An alternative to taking single 'best fit' models is to try and characterize the subset of data acceptable models in the collected ensemble, which may be useful if the search algorithm has sufficiently explored the parameter space. The earliest efforts in this direction consisted of simply directly comparing the acceptable models in the ensemble. This was the approach taken by Press (1968, 1970a, 1970b), however in that case just six Earth models were found from 5 million which fit all of the available seismic data and prior constraints.

Another way of characterizing an acceptable ensemble is to look for particular properties, or features, which all data acceptable models share, e.g. ones which have the least structure, or detail, as measured by some particular criteria. This extremal model approach was proposed by Parker (1977) in the context of nonlinear inverse problems, and can also be applied to the appraisal problem. Again the degree to which one can sensibly draw inferences from the ensemble depends crucially on the type of sampling performed during the search stage. Therefore, just as in a Bayesian approach, the search and appraisal stages are linked. For example, to generate many data acceptable models one needs the search algorithm to first find and then 'map out' the regions of parameter space where the fit to data is acceptable (which is often a difficult task, see Sambridge (2001)); while for the extremal model approach one needs the search stage to optimize a combination of data fit and model property, (often called a regularizing property).

Other methods have been proposed for characterizing a multi-dimensional ensemble of data-acceptable models. A summary can be found in Sen and Stoffa (1995). Examples include Vasco *et al* (1993) who used cluster analysis techniques, while Douma *et al* (1996) used a projection onto empirical orthogonal functions to determine the common features in their data-acceptable ensemble of Earth models (seismic wave speed as a function of depth). Another popular approach has been through semi-graphical methods (Basu and Frazer 1990, Nolte and Frazer 1994, Lomax and Snieder 1995, Kennett 1998). A draw back of many of these techniques is that they are best suited to the situation where the acceptable ensemble forms a single cluster, and not multiple unconnected clusters in parameter space. Recently, Sambridge (2001) has proposed a technique to map out the acceptable ensemble which is applicable to the multiple cluster case.

3.5. Design of random walks for optimization

3.5.1. Simulated annealing: Metropolis and Gibbs as optimizers.

The simulated annealing algorithm. When crystalline material is slowly cooled through its melting point, highly ordered, low-energy crystals are formed. The slower the cooling, the lower the final lattice energy. This physical process is a natural optimization method where the lattice energy E is the objective function to be minimized. Large numerical systems can be run through a similar optimization process if we identify parameters of the system with state space variables, and the objective function of the optimization problem with the energy E.

Thermal fluctuations in the system are simulated by randomly perturbing its parameters, and the size of the fluctuations are controlled by a temperature parameter T.

The simulated annealing algorithm (Kirkpatrick *et al* 1983) is a specialization and modification of the Metropolis algorithm, in that the desired equilibrium distribution for a constant value of the temperature parameter is the Gibbs–Boltzmann distribution

$$p_B(\boldsymbol{x}) = \exp(-E(\boldsymbol{x})/T)Z(T) \tag{35}$$

where $1/Z(T)$ is a normalization constant. This is the distribution over the state space of a statistical mechanical system in equilibrium with a heat bath of temperature T, and this is the kind of system we wish to simulate with this algorithm. Simulation of chemical annealing is now performed by gradually lowering the temperature T from a high value to near-zero. Close to $T = 0$ the Gibbs–Boltzmann distribution approximates a delta function at the global minimum for $E(\boldsymbol{x})$ (if it is unique).

Simulated annealing can also be realized by generating samples from the Gibbs–Boltzmann distribution by means of a Gibbs sampler. Rothman (1986) solves the so-called residual statics problem of reflection seismology in this way.

The Nulton–Salamon annealing schedule. Strictly speaking, simulated annealing is only guaranteed to work for infinitely slow 'cooling'. The practical problem is therefore: How can we decrease the temperature of the system in a finite number of steps, such that the final value of E, on average, is as close as possible to the global minimum for E?

Nulton and Salamon (1988) proposed an annealing method based on the idea that the numerical system should be kept as close to 'thermal equilibrium' as possible at all times. This was done by keeping the actual mean value $\langle E\rangle$ of the objective function at a constant distance

$$v = \frac{\langle E\rangle - \langle E\rangle_{eq}}{\sigma_E(T)} \tag{36}$$

from the theoretical (but never realized) equilibrium mean value $\langle E\rangle_{eq}$. In (36) $\sigma_E(T)$ is the standard deviation of $E(\boldsymbol{x})$ which, of course, fluctuates from iteration to iteration. v is known as the 'thermodynamic speed' and sometimes also as the 'thermodynamic distance'. The following differential equation for the annealing temperature schedule $T(t)$ can now be derived:

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -\frac{vT}{\epsilon(T)\sqrt{C(T)}} \tag{37}$$

where $C(T)$ is the *heat capacity* of the system, and $\epsilon(T)$ is its *relaxation time*. Andresen *et al* (1988) estimate an approximate, temperature-dependent, transition probability matrix $P_E(T)$ for transitions between 'energy levels' by monitoring transition frequencies during the annealing process. For each temperature, the heat capacity $C(T)$ can be evaluated from the eigenvector of $P_E(T \to \infty)$ with eigenvalue 1, and the relaxation time $\epsilon(T)$ can be calculated from the second largest eigenvalue of $P_E(T)$.

The thermodynamic speed v in equation (37) is calibrated, through some initial experimentation, to the problem at hand, such that the annealing schedule is close to zero after a predetermined number of iterations. The total number of iterations is, of course, limited by the available computer resources (Jakobsen *et al* 1988, Andresen *et al* 1988, Koren *et al* 1991).

In a more direct approach, the so-called *ensemble-based simulated annealing* (EBSA), heat capacities and relaxation times are estimated using the statistics of a set (ensemble, population) of parallel simulated-annealing walkers at the same temperature. In contrast to genetic algorithms, the individual members of an EBSA ensemble/population do not interact directly. Instead, thermodynamic properties of the problem are calculated from ensemble

averages, and optimal annealing schedules can be calculated on-the-fly or after an initial test run (Salamon *et al* 2002, Mosegaard and Vestergaard 1991).

3.5.2. Genetic algorithms. Genetic algorithms were originally devised as a model of adaptation in an artificial system, by Holland (1975). Early reference works are by Davis (1987), Goldberg (1989), and a useful tutorial can be found in Whitley (1994). Genetic algorithms fall into the class of Monte Carlo techniques because they also use random numbers to control the sampling of a parameter space. In contrast to simulated annealing which uses an analogy with a physical optimization process, genetic algorithms are based on analogy with biological evolution.

Within the past decade genetic algorithms have been applied in a wide range of the areas within the physical sciences, and for a range of purposes, only one of which is optimization. Many excellent references and web pages exist describing genetic algorithms and their variants (see the above works and references cited therein). Here we describe a simple genetic algorithm and compare it to other Monte Carlo importance sampling techniques by examining how it might be used to sample a particular probability distribution.

A key feature of genetic algorithms is the representation of physical variables of interest by a simple string data structure, and usually a binary string. It is straightforward to represent many optimization problems involving real-valued variables into a set of binary string representations. For a single variable, x_i, the simplest approach would be to choose upper and lower bounds, u_i and l_i, together with a discretization interval, and assign a binary number to each value the variable could take. For example, one might encode a real variable x_i into a binary number b_i, using

$$b_i = B\left\{\frac{x_i - l_i}{u_i - l_i}(2^l - 1)\right\} \tag{38}$$

where l is the length of the binary string produced, and we have introduced the operator $B\{y\}$ to indicate taking the binary value of y after rounding down to the nearest integer.

The genetic algorithm uses the bit string data structure to manipulate an ensemble of parameter space samples at each iteration. Here we restrict ourselves to a description of the basic three-operator genetic algorithm. Many variants exist, details of which can be found in the references cited.

Selection. From the initial population $\pi = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, of N strings an interim population of N parents is generated by randomly selecting strings with replacement, where the probability of selection, p_s, is a function of the fitness of each string. A common choice is

$$p_s(\boldsymbol{x}_j) = A \exp[-f(\boldsymbol{x}_j)/B] \tag{39}$$

where A is a normalizing constant,

$$A^{-1} = \sum_i \exp[-f(\boldsymbol{x}_i)/B] \tag{40}$$

and B is some measure of the spread of fitness values in the current population, e.g. the standard deviation. Since the population size is unchanged, the selection operator will generate multiple copies of some models in $\boldsymbol{x}_i$ at the expense of others, which may be extinguished completely. It is in this stage that the fitness of each string influences its chances of survival.

Crossover. From the parent population of N strings a new generation of N strings is produced by mixing pairs together. All of the parents are randomly paired to produce $N/2$ couples.

A cross-over probability p_c is assigned and each pair of strings is chosen for crossover with probability, p_c. If crossover is selected then the two strings are cut at a uniformly random point along their paths and two new strings are formed by interchanging the sub-strings. If crossover is not selected then the two parent strings are passed unscathed to the next generation.

Mutation. The final stage involves the process whereby any bit in an individual string is allowed to flip between 0 and 1 with probability p_m. This allows some degree of local diversity to be introduced into the population.

The action of the three types of process is to produce a new generation of N bit strings each of which has a new fitness value and the whole process can be repeated. After many iterations the population has the potential to evolve towards a fitter on average state. In general there appears to be no clear theory on how to choose the values of control parameters p_s, p_m N and B, to obtain an optimal performance for particular problems. Usually some level of empirical tuning is common. More sophisticated versions of genetic algorithms may also introduce extra control parameters, e.g. using different types of mapping from fitness to survival probability in (39), or perhaps different forms of crossover involving multiple string cuts.

The fundamental theorem. A theoretical analysis of the way genetic algorithms process information was performed by Holland (1975), which resulted in the *fundamental theorem*. Goldberg (1989) discusses the theorem in detail and shows how it explains the processing of *Schema* over each iteration. Schema are simply patterns within the binary strings representing each sample in parameter space. For a bit string of length seven ($l = 7$) an example of a schemata would be,

$$H = *11*0*1 \tag{41}$$

where the asterisk represents either a 1 or 0. H is the subset of bit strings differing in only the 1.0, 4.0 and 6.0 bits, and clearly represents many different possible combinations of bit-strings. Two properties of schema can be defined. The first is the *order*, represented by the operator $o(H)$, which gives the number of fixed positions (i.e. the number of 1 s or 0 s in the schemata), and the second is the *defining length*, $\delta(H)$, which is the distance between the first and last specified string position. For example, the schemata in (41) would have,

$$o(H) = 4, \qquad \text{and} \qquad \delta(H) = 5.$$

These quantities appear in the fundamental theorem which predicts how the number of schema changed between iterations of a genetic algorithm due to the three basic operations of selection, crossover and mutation. If we define $m(H, t)$ as the expected number of examples of schema H at iteration t, then the fundamental theorem says,

$$m(H, t+1) \geqslant m(H, t)\frac{f(H)}{\overline{f}}\left[1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m\right] \tag{42}$$

where $f(H)$ represents the average fitness of all strings in the current population that correspond to schema H, and $\overline{f}$ is the average fitness of the population. Since each bit-string in a population belongs to many schema the genetic algorithm manipulates many schema in parallel at each iteration. The main conclusion that follows from (42) is that short, low-order, above-average schemata receive exponentially increasing numbers of copies in subsequent generations. Most theoretical analysis of genetic algorithms performance in optimization has centred on the way in which schema are created and destroyed.

During the 1990s, convergence properties of genetic algorithms were studied extensively. Conditions for guaranteed convergence towards optimal solutions were worked out for

important classes of algorithms, and convergence rates in terms of control parameters were estimated. Here, we shall not elaborate further on this interesting topic, but readers who want further details may, e.g., consult papers by Gao (1998) and Greenhalgh and Marshall (2000).

Genetic algorithms and sampling of probability distributions. Genetic algorithms were not designed to sample given probability distributions, but the efficiency of these algorithms for optimization has led researchers to search for ways of modifying them, so that they could be used for sampling (see, e.g., Sen and Stoffa (1995)). Here we shall take a close look at the problems we are facing when we try to develop GAs for sampling, and explain why the problem has not yet been solved.

A genetic algorithm is not based on a single random walk in $\mathcal{X}$. Instead, it is based on an ensemble of, say, N random walkers, operating simultaneously, and dependent on each other. This process can be understood if we consider a new space $\mathcal{P} = \mathcal{X}^N$, the *population space*, whose points are all possible *populations* consisting of points (here called *individuals*) from the original space $\mathcal{X}$. Note that, traditionally, $\mathcal{X}$ is a discrete space where the points are represented by strings of binary numbers. In $\mathcal{P}$ we define the probability of a population as the product of the probabilities of the individual points in the population.

We now re-examine each of three operators in a simple genetic algorithm, to determine whether they can be designed to sample $\mathcal{P}$ uniformly (that is, with a constant probability distribution). If this is the case then it is a simple matter to modify the GA such that it samples a given distribution $p(\boldsymbol{x})$ over $\mathcal{P}$. The reader can easily verify that the following algorithm will indeed sample $p(\boldsymbol{x})$:

(i) Given a point $\boldsymbol{x}_j \in \mathcal{P}$ currently visited by the algorithm. Use the uniform sampler to propose a point $\boldsymbol{x}_i$ as a candidate to be the next sample.
(ii) Accept only $\boldsymbol{x}_i$ with probability

$$p_{accept} = \begin{cases} \dfrac{p(\boldsymbol{x}_i)}{p(\boldsymbol{x}_j)} & \text{for } p(\boldsymbol{x}_i) \leqslant p(\boldsymbol{x}_j) \\ 1 & \text{otherwise.} \end{cases} \tag{43}$$

As we shall see, it is straightforward to design the mutation and the crossover substeps to give a uniform distribution, but not the selection substep. We will consider them in reverse order.

The mutation substep. The mutation substep is easy to design such that it has the uniform distribution over $\mathcal{P}$ as its equilibrium distribution. In this substep we select one individual $\boldsymbol{x}_j$ from the population space $\pi = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, uniformly at random (that is, giving each individual probability $1/N$ of being chosen). We now generate a new candidate individual $\boldsymbol{x}_i$ using an irreducible proposal distribution over $\mathcal{X}$ that is uniform in a neighbourhood surrounding (and including $\boldsymbol{x}_j$). Finally, we accept it with probability 1. This operation is one step of a random walk in $\mathcal{X}$ that, should it be iterated, would sample $\mathcal{X}$ uniformly. It is also clear that the entire population will, in this way, sample the population space $\mathcal{P}$ uniformly.

The crossover substep. In crossover we select one pair of individuals, $\xi_q = (\boldsymbol{x}_j, \boldsymbol{x}_k)$ from the population, uniformly at random, and generate a new candidate pair $\xi_r = (\boldsymbol{x}_i, \boldsymbol{x}_l)$ using a proposal distribution $U(\xi_r|\xi_q)$ over $\mathcal{X}^2$ (which in this case is *not* irreducible) that assigns equal probability to all possible pairs ξ_r obtained by choosing an integer $1 \leqslant \nu \leqslant \dim(\mathcal{X})$ uniformly at random, and then replacing the first ν components of $\boldsymbol{x}_j$ with the first ν components of $\boldsymbol{x}_k$, and vice versa. Finally, we accept the new pair ξ_r with probability 1.

This operation is one step of a random walk in $\mathcal{X}^2$ with the uniform distribution over $\mathcal{X}^2$ as an equilibrium distribution. It is also clear that the corresponding move of the entire population is one step of a random walk in $\mathcal{P}$ with the uniform distribution over $\mathcal{P}$ as an equilibrium distribution.

The selection substep. The problems arise in the selection substep. It is tempting, from the population π_j of N individuals to generate a new population π_i of N individuals by selecting models from π_j with uniform probability of selection, i.e. set $p_s(\pi_j)$ equal to a constant rather than using (39). As stated above this will generate multiple copies of some models in π_i at the expense of others, which may be removed completely.

The problem is now that the selection substep so defined violates microscopic reversibility and therefore has a non-uniform equilibrium distribution. The point is that selection is irreversible: it can only remove individuals from the population, not create them again. Hence, our results concerning multi-step iterations do not apply. Forming complete iterations by combining mutation, crossover and selection will not lead to a uniform sampling, and no modification in selection probabilities will make it do so.

These observations do not mean that it is impossible to find mutation-, crossover- and selection-substeps that, taken together in each iteration, will sample the population space uniformly. It only means that these three substeps cannot be designed independently. Some mechanism must be built into the mutation- and crossover substeps that compensate for the irreversibility of the selection substep, such that microscopic reversibility is restored. Unfortunately, no solution has yet been found to this intricate problem. If it can be found, it will no doubt have great practical significance.

It is sometimes claimed that a GA who 'has a desired distribution as an approximate equilibrium distribution' is a sufficiently good solution to the above-mentioned problem, because even a Metropolis algorithm designed to sample the distribution will only give approximate solutions (due to limited sampling time). There are two reasons why this argument is incorrect:

(i) A clear definition what is meant by 'an approximate equilibrium distribution' is usually lacking.

(ii) A genetic algorithm with only an 'approximately correct' equilibrium distribution will, asymptotically, deviate more and more from the desired equilibrium distribution (and nobody knows how much). In contrast to this, a Metropolis algorithm having the desired equilibrium distribution will, asymptotically, sample this distribution.

This reasoning points to a clear difference between genetic algorithms and other Monte Carlo sampling methods.

3.5.3. The neighbourhood algorithm. The neighbourhood algorithm is a recently developed Monte Carlo search algorithm that has found a number of applications in seismic inverse problems (Sambridge 1998, 1999a, 1999b, 2001). This is an ensemble based search technique that uses concepts from computational geometry to guide the sampling of a parameter space.

The basic idea is that at each iteration the multi-dimensional parameter space is partitioned into a set of *Voronoi polytopes* constructed (uniquely) about the samples generated from previous iterations, $\boldsymbol{x}_j, (j = 1, \ldots, n)$. Voronoi polytopes (cells) are simply nearest-neighbour regions, as determined by a suitable distance measure. An L_2-norm is a common choice. More formally, any new point $\boldsymbol{x}$, lies within the ith Voronoi cell if,

$$\|\boldsymbol{x} - \boldsymbol{x}_i\| < \|\boldsymbol{x} - \boldsymbol{x}_j\| \qquad (j = 1, \ldots, i-1, i+1, \ldots, n). \tag{44}$$

Voronoi cells have some useful properties in that they are always unique, space filling, and adapt to the density of the samples about which they are constructed (one per cell), i.e. as the

local sampling density increases the size of each Voronoi cell decreases. Since each Voronoi cell forms a neighbourhood about one of the previously generated samples, they can be used to guide further sampling. One choice is to place samples uniformly randomly in selected Voronoi cells. A Gibbs sampler is ideal for this purpose. For example, if the jth Voronoi cell were chosen for sampling then a random walk would begin from model $\boldsymbol{x}_j$. The proposal distribution for the Gibbs sampler at the kth substep, is to be given by (26), with

$$p(\boldsymbol{x}_i) = \begin{cases} 1, & \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \leqslant \|\boldsymbol{x}_i - \boldsymbol{x}_l\| \ (l = 1, \ldots, n) \\ 0, & \text{otherwise} \end{cases} \tag{45}$$

and the acceptance probability is unity. Selection of cells to re-sample can be made either deterministically or probabilistically on the basis of the fitness function evaluated at the corresponding samples. A simple choice would be to re-sample only, say, the 10% of cells with the higher fitness values. The number of cells chosen, n_r, would become a control parameter of the algorithm. Since these cells are not necessarily neighbours of each other, sampling can be performed in multiple regions simultaneously.

An alternative to re-sampling chosen Voronoi cells is to again use a standard Gibbs sampler to generate new samples distributed according to the *neighbourhood approximation* of the the fitness/objective function. This is formed by setting the fitness to a constant inside each Voronoi cell. When a Gibbs sampler is applied to the neighbourhood approximation, it is equivalent to setting the fitness function, $f(\boldsymbol{x})$, of any point equal to that of its nearest point among the current set, i.e. we have,

$$f_{NA}(\boldsymbol{x}) = f(\boldsymbol{x}_j) \qquad \{\boldsymbol{x}_j : \min(\|\boldsymbol{x} - \boldsymbol{x}_l\|) \ (l = 1, \ldots, n)\}. \tag{46}$$

In this case the proposal distribution for the Gibbs sampler becomes (26), with

$$p(\boldsymbol{x}_i) = f_{NA}(\boldsymbol{x}_i). \tag{47}$$

Note that the Gibbs sampler is used to sample from the neighbourhood approximation of the fitness function (47), or the Voronoi cells directly (45), and in both cases no further fitness evaluations are required for the random walks. An iteration is complete when the required number of new models, n_s, are selected from the chain of samples produced by the random walks. Only then is the fitness of each new sample calculated and the Voronoi cells (updated to included the latest n_s samples) for the next iteration. (Note that n_s is a second control parameter.) In each of the two scenarios represented by (47) and (45), multiple random walks can be performed in parallel starting from the previous samples $f(\boldsymbol{x}_j)(j = 1, \ldots, n)$.

Like a genetic algorithm the neighbourhood approach generates a new ensemble of samples at each iteration, but does so using the Voronoi cell concept to identify 'promising regions' of parameter space. It also has similarities with other Monte Carlo methods, since it makes use of a Gibbs sampler. We note here that one could equally well make use of the neighbourhood approximation of the fitness function within other optimization procedures like simulated annealing or genetic algorithms, i.e. to intermittently replace expensive evaluations of the fitness function, with cheaper but more approximate evaluations of $f_{NA}(\boldsymbol{x})$. Sambridge (1999a) has shown that the neighbourhood algorithm can result in quite a complex 'self-adaptive' search process in global optimization.

In addition to searching a parameter space for optimization the neighbourhood algorithm may also be used in the appraisal problem, i.e. analysing the resulting ensemble. Sambridge (1999b) has shown how one can use the neighbourhood approximation within a Bayesian formulation to estimate resolution and parameter covariances from an arbitrarily distributed ensemble of samples. Initial results with this technique for both search and appraisal are quite promising but more experience is required. In addition it is not yet known whether the search algorithm could be applied to combinatorial optimization, because of the lack of a general definition of a Voronoi cell (nearest neighbour region) for combinatorial problems.

4. Conclusions

Monte Carlo methods provide a systematic way of dealing with (discrete) inverse problems for which we have incomplete knowledge of the relationship between data and model parameters. This is the case, e.g., for many highly nonlinear problems, where the forward relation is insusceptible to mathematical analysis, and is only given by a complex algorithm. Monte Carlo methods can be divided into two groups, the first of which is devoted to sampling from a probability density, and the second is designed to search for near-optimal solutions to the problem.

The most widely used sampling algorithms are the Metropolis algorithm and the Gibbs sampler, which are used in cases where a thorough resolution and uncertainty analysis is called for. Although the equilibrium theory of these algorithms is simple and well mapped-out, many theoretical and practical problems concerning their 'speed of equilibration' remain to be solved.

The dominating stochastic optimization algorithms are simulated annealing and the genetic algorithms. In many applications, the genetic algorithm has shown its strength as a search method. Attempts at constructing a sampling algorithm from the same principles as the genetic algorithm has failed so far, but if the problem can be solved, it will no doubt be a great step forward in many practical applications.

In this paper we have covered the basic Monte Carlo algorithms currently in use in applications. Several variants of these algorithms exist, many of which are adaptations of the basic methods, or exploit special properties of the problem considered. The readers are referred to the literature for information on these methods. Important examples are *reversible jump MCMC* (Green 1995) and *Simulated Tempering* (Geyer and Thompson 1995, Marini and Parisi 1992). Readers who wish to dive into the basic theory of Markov chains may consult Kemeny and Snell (1976), or Seneta (1981).

Acknowledgments

The authors would like to thank Amir Khan for providing the figures illustrating the inversion of the Apollo seismic data. KM would like to thank Albert Tarantola, Peter Salamon, Bjarne Andresen and many other colleagues for stimulating cooperation on Monte Carlo methods over the years.

References

Anderssen R S 1970 The character of non-uniqueness in the conductivity modelling problem for the Earth *Pure Appl. Geophys.* **80** 238–59

Anderssen R S and Senata E 1971 A simple statistical estimation procedure for Monte Carlo inversion in geophysics *Pure Appl. Geophys.* **91** 5–13

Anderssen R S and Senata E 1972 A simple statistical estimation procedure for Monte Carlo inversion in geophysics: efficiency and Hempel's paradox *Pure Appl. Geophys.* **96** 5–14

Anderssen R S, Worthington M H and Cleary J R 1972 Density modelling by Monte Carlo inversion—I methodology *Geophys. J. R. Astron. Soc.* **29** 433–44

Andresen B, Hoffman K H, Mosegaard K, Nulton J, Pedersen J M and Salamon P 1988 *J. Physique* **49** 1485

Basu A and Frazer L N 1990 Rapid determination of critical temperature in simulated annealing inversion *Science* **249** 1409–12

Biswas N N and Knopoff L 1974 The structure of the upper mantle under the United States from dispersion of Rayleigh waves *Geophys. J. R. Astron. Soc.* **36** 515–39

Burton P W 1977 Inversions of high frequency $Q_\gamma^{-1}(f)$ *Geophys. J. R. Astron. Soc.* **48** 29–51

Burton P W and Kennett B L N 1974a Upper mantle zone of low *Q Nature* **238** 84

Burton P W and Kennett B L N 1974b Upper mantle zone of low *Q Nature* **238** 87–90

Cary P W and Chapman C H 1988 Automatic 1D waveform inversion of marine seismic refraction data *Geophys. J.* **93** 527–46

Dahl-Jensen D, Mosegaard K, Gundestrup N, Clow G D, Johnsen S J, Hansen A W and Balling N 1998 Past temperatures directly from the Greenland ice sheet *Science* **9** 268–71

Davis L 1987 Genetic algorithms and simulated annealing *Research Notes in Artificial Intelligence* (London: Pitman)

Douma H, Snieder R and Lomax A 1996 Ensemble inference in terms of empirical orthogonal functions *Geophys. J. Int.* **127** 363–78

Fishman G S 1996 *Monte Carlo. Concepts, Algorithms, and Applications* (New York: Springer)

Gallagher K and Sambridge M 1994 Genetic algorithms: a powerful tool for large-scale non-linear optimization problems *Comput. Geosci.* **20** 1229–36

Gallagher K, Sambridge M S and Drijkoningen G G 1991 Genetic algorithms: an evolution on Monte Carlo methods in strongly non-linear geophysical optimization problems *Geophys. Res. Lett.* **18** 2177–80

Gao Y 1998 An upper bound on the convergence rates of canonical genetic algorithms *Complexity International* vol 5

Gelman A, Roberts G O and Gilks W R 1996 Efficient Metropolis jumping rules *Bayesian Statistics* vol 5, ed J M Bernardo, J O Berger, A P Dawid and A F M Smith (Oxford: Clarendon) pp 599–608

Geman S and Geman D 1984 Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images *IEEE Trans. Patt. Analysis Mach. Int.* **6** 721–41

Geyer C J and Thompson E A 1995 Annealing Markov chain Monte Carlo with applications to ancestral inference *J. Am. Stat. Assoc.* **90** 909–20

Goldberg D E 1989 *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA: Addison-Wesley)

Goncz J H and Cleary J R 1976 Variations in the structure of the upper mantle beneath Australia, from Rayleigh wave observations *Geophys. J. R. Astron. Soc.* **44** 507–16

Gouveia W P and Scales J A 1998 Bayesian seismic waveform inversion: parameter estimation and uncertainty analysis *J. Geophys. Res.* **103** 2759–79

Green P 1995 Reversible jump Markov chain Monte Carlo computation and Bayesian model determination *Biometrica* **82** 711–32

Greenhalgh D and Marshall S 2000 Convergence criteria for genetic algorithms *SIAM J. Comput.* **30** 269–82

Haddon R A W and Bullen K E 1969 An earth model incorporating free earth oscillation data *Phys. Earth Planet. Inter.* **2** 35–49

Hastings W K 1970 Monte Carlo sampling methods using Markov Chain and their applications *Biometrika* **57** 97–109

Holland J H 1975 *Adaptation in Natural and Artificial Systems* (Ann Arbor, MI: The University of Michigan Press)

Hood L and Zuber M 2000 Recent refinements in geophysical constraints on Lunar origin and evolution *The Origin of the Earth and Moon* ed R Canup and K Righter (Tucson, AZ: University of Arizona Press)

Housholder A S (ed) 1951 *Monte Carlo Method (Mathematics Series 12)* (Washington, DC: National Bureau of Standards)

Jakobsen M O, Mosegaard K and Pedersen J M 1988 Global model optimization in reflection seismology by simulated annealing *Model Optimization in Exploration Geophysics 2* ed E Vogel (Wiesbaden: Braunschweig) p 361

Jestin F, Huchon P and Gaulier J M 1994 The Somalia plate and the East African rift system; present-day kinematics *Geophys. J. Int.* **116** 637–54

Jones A G and Hutton R 1979 A multi-station magnetotelluric study in southern Scotland; II, Monte Carlo inversion of the data and its geophysical and tectonic implications *Geophys. J. R. Astron. Soc.* **56** 351–68

Kaipio J P, Kolehmainen V, Somersalo E and Vauhkonen M 2000 Statistical inversion and Monte Carlo sampling methods in electrical impedance tomography *Inverse Problems* **16** 1083–618

Keilis-Borok V I and Yanovskaya T B 1967 Inverse problems of seismology *Geophys. J.* **13** 223–34

Kemeny J G and Snell J L 1976 Finite Markov chains *Springer Undergraduate Texts in Mathematics* 2nd edn

Kennett B L N 1998 On the density distribution within the earth *Geophys. J. Int.* **132** 374–82

Kennett B L N and Nolet G 1978 Resolution analysis for discrete systems *Geophys. J. R. Astron. Soc.* **53** 413–25

Kennett B L N and Sambridge M 1992 Earthquake location; genetic algorithms for teleseisms *Phys. Earth Planet. Inter.* **75** 103–10

Khan A and Mosegaard K 2001 New information on the deep lunar interior from an inversion of lunar free oscillation periods *Geophys. Res. Lett.* **28** 1791

Khan A, Mosegaard K and Rasmussen K L 2000 A new seismic velocity model for the Moon from a Monte Carlo inversion of the Apollo Lunar seismic data *Geophys. Res. Lett.* **27** 1591–4

Kirkpatrick S C, Gelatt D and Vecchi M P 1983 Optimization by simulated annealing *Science* **220** 671–80

Koren Z, Mosegaard K, Landa E, Thore P and Tarantola A 1991 Monte Carlo estimation and resolution analysis of seismic background velocities *J. Geophys. Res.* **96** 20 289–99

Landa E, Beydoun W and Tarantola A 1989 Reference velocity model estimation from prestack waveforms; coherency optimization by simulated annealing *Geophysics* **54** 984–90

Lomax A and Snieder R 1995 Identifying sets of acceptable solutions to non-linear geophysical inverse problems which have complicated misfit functions *Nonlinear Process. Geophys.* **2** 222–7

Marini E and Parisi G 1992 Simulated tempering: a new Monte Carlo scheme *Europhys. Lett.* **19** 451–8

Marroquin J, Mitter S and Poggio T 1987 Probabilistic solution of ill-posed problems in computational vision *J. Am. Stat. Assoc.* **82** 76–89

Metropolis N, Rosenbluth M N, Rosenbluth A W, Teller A H and Teller E 1953 Equation of state calculations by fast computing machines *J. Chem. Phys.* **21** 1087–92

Metropolis N and Ulam S M 1949 The Monte Carlo method *J. Am. Stat. Assoc.* **44** 335–41

Mills J M and Fitch T J 1977 Thrust faulting and crust-upper mantle structure in East Australia *Geophys. J. R. Astron. Soc.* **48** 351–84

Mosegaard K and Rygaard-Hjalsted C 1999 Bayesian analysis of implicit inverse problems *Inverse Problems* **15** 573–83

Mosegaard K and Tarantola A 1995 Monte Carlo sampling of solutions to inverse problems *J. Geophys. Res.* **100** 12 431–47

Mosegaard K and Vestergaard P 1991 A simulated annealing approach to seismic model optimization with sparse prior information *Geophys. Prospect.* **39** 599–611

Nolte B and Frazer L N 1994 Vertical seismic profile inversion with genetic algorithms *Geophys. J. Int.* **117** 162–79

Nulton J D and Salamon P 1988 Statistical mechanics of combinatorial optimization *Phys. Rev.* A **37** 1351–6

Parker R L 1977 Understanding inverse theory *Ann. Rev. Earth Planet Sci.* **5** 35–64

Parker R L 1994 *Geophysical Inverse Theory* (Princeton, NJ: Princeton University Press)

Pedersen J B and Knudsen O 1990 Variability of estimated binding parameters *Biophys. Chem.* **36** 167–76

Press F 1968 Earth models obtained by Monte Carlo inversion *J. Geophys. Res.* **73** 5223–34

Press F 1970a Earth models consistent with geophysical data *Phys. Earth Planet. Inter.* **3** 3–22

Press F 1970b Regionalized Earth models *J. Geophys. Res.* **75** 6575–81

Ricard Y, Vigny C and Froidevaux C 1989 Mantle heterogeneities, geoid, and plate motion; a Monte Carlo inversion *J. Geophys. Res.* **94** 13 739–54

Rothman D H 1985 Nonlinear inversion statistical mechanics, and residual statics corrections *Geophysics* **50** 2784–96

Rothman D H 1986 Automatic estimation of large residual statics corrections *Geophysics* **51** 332–46

Rygaard-Hjalsted C, Mosegaard K and Olsen N 2000 Resolution studies of fluid flow models near the core–mantle boundary through Bayesian inversion of geomagnetic data *Methods and Applications of Inversion: Proc. IIC98 Conf. (Copenhagen, 1998)* ed P C Hansen, B H Jacobsen and K Mosegaard, pp 255–75

Salamon P, Sibani P and Frost R 2002 *Facts, Conjectures and Improvements for Simulated Annealing (SIAM Monographs on Mathematical Modeling and Computation)* at press

Sambridge M 1998 Exploring multi-dimensional landscapes without a map *Inverse Problems* **14** 427–40

Sambridge M 1999a Geophysical inversion with a neighbourhood algorithm—I. Searching a parameter space *Geophys. J. Int.* **138** 479–94

Sambridge M 1999b Geophysical inversion with a neighbourhood algorithm—II. Appraising the ensemble *Geophys. J. Int.* **138** 727–46

Sambridge M 2001 Finding acceptable models in nonlinear inverse problems using a neighbourhood algorithm *Inverse Problems* **17** 387–403

Sambridge M and Drijkoningen G G 1992 Genetic algorithms in seismic waveform inversion *Geophys. J. Int.* **109** 323–42

Sambridge M and Mosegaard K 2001 Monte Carlo methods in geophysical inverse problems *Rev. Geophys.* submitted

Scales J A, Smith M L and Fischer T L 1992 Global optimization methods for multimodel inverse problems *J. Comput. Phys.* **103** 258–68

Sen M K and Stoffa P L 1992 Rapid sampling of model space using genetic algorithms *Geophys. J. Int.* **108** 281–92

Sen M K and Stoffa P L 1995 Global optimization methods in geophysical inversion *Advances in Exploration Geophysics* vol 4 (Amsterdam: Elsevier)

Seneta E 1981 *Non-Negative Matrices and Markov Chains* 2nd edn (Berlin: Springer)

Shibutani T, Sambridge M and Kennett B 1996 Genetic algorithm inversion for receiver functions with application to crust and uppermost mantle structure beneath Eastern Australia *Geophys. Res. Lett.* **23** 1829–32

Smith M L, Scales J A and Fischer T L 1992 Global search and genetic algorithms *Geophysics: The Leading Edge of Exploration* pp 22–6

Stoffa P L and Sen M K 1991 Nonlinear multiparameter optimization using genetic algorithms: inversion of plane wave seismograms *Geophysics* **56** 1794–810

Vasco D W, Johnson L R and Majer E L 1993 Ensemble inference in geophysical inverse problems *Geophys. J. Int.* **117** 711–28

Whitley D L 1994 A genetic algorithm tutorial *Stat. Comput.* **4** 65–85

Wiggins R A 1969 Monte Carlo inversion of body wave observations *J. Geophys. Res.* **74** 3171–81

Wiggins R A 1972 The general inverse problem: implication of surface waves and free oscillations for Earth structure *Rev. Geophys. Space Phys.* **10** 251–85

Wilson W G and Vasudevan K 1991 Application of the genetic algorithm to residual statics estimation *Geophys. Res. Lett.* **18** 2181–4

Worthington M H, Cleary J R and Anderssen R S 1972 Density modelling by Monte Carlo inversion—II comparison of recent earth models *Geophys. J. R. Astron. Soc.* **29** 445–57

Worthington M H, Cleary J R and Anderssen R S 1974 Upper and lower mantle shear velocity modelling by Monte Carlo inversion *Geophys. J. R. Astron. Soc.* **36** 91–103