Los Alamos Report LA-UR-02-4105
summary of talk presented at
7th Valencia International Meeting on Bayesian Statistics
June 5, 2002, Tenerife, Spain

Use of probability gradients in hybrid MCMC and a new convergence test

KENNETH M. HANSON
Los Alamos National Laboratory, USA
kmh@lanl.gov

SUMMARY

The hybrid Markov Chain Monte Carlo technique affords a robust means for sampling multidimensional probability density functions with high efficiency, provided one can calculate the gradient of φ = minus-log-probability in a time comparable to calculating the probability itself. The latter condition is met using the technique of adjoint differentiation. The gradient of φ may also be used to form a measure of the degree of convergence of the MCMC sequence. The proposed statistic is the ratio of sample estimates for the variance of the distribution calculated two different ways, one based on using integration by parts of the integral for the variance and the other based on the standard second-moment calculation. The efficiency of the hybrid MCMC technique and the usefulness of the convergence test will be demonstrated for simple multivariate normal distributions.

Keywords: ADJOINT DIFFERENTIATION; MARKOV CHAIN MONTE CARLO; HYBRID MCMC; HAMILTONIAN METHOD; HAMILTONIAN DYNAMICS; LEAPFROG METHOD; METROPOLIS MCMC; CONVERGENCE TEST; MULTIVARIATE DISTRIBUTIONS; SIMULATION SCIENCE; SIMULATION VALIDATION.

1. INTRODUCTION

A broad effort is underway in simulation science to develop methods for conducting Bayesian inference in regard to large simulation codes (Hanson et al., 2002). Of particular interest are methods that can cope with large numbers of parameters, say hundreds or more, in a context where a function evaluation can take several hours or even days to perform on state-of-the-art computer systems. The ultimate goal is to treat problems involving large simulations, for example, ocean or atmospheric models, aerodynamics, and hydrodynamics. In this kind of analysis, it is essential to develop techniques that reduce the number of function evaluations required to reach a specified degree of accuracy in estimating the uncertainties in the output of these models, as well as drawn inferences about the uncertainties in the components of the models themselves.

In Bayesian analysis, the posterior probability distribution characterizes the uncertainty in the model parameters estimated from a given set of measurements. It has become evident that the Markov Chain Monte Carlo (MCMC) technique provides a straightforward

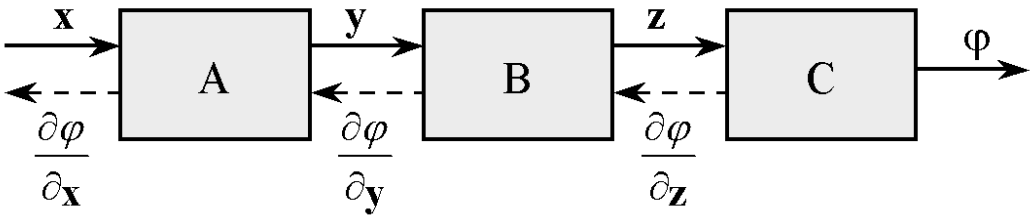

Figure 1. *A data-flow diagram describing a sequence of transforms of an input data vector* $\boldsymbol{x}$ *into a scalar output functional* φ*. The data structures* $\boldsymbol{x}$*,* $\boldsymbol{y}$ *and* $\boldsymbol{z}$ *may be large. The gradient of* φ *with respect to the* $\boldsymbol{x}$ *is most efficiently evaluated by propagating derivatives of* φ *with respect to the intermediate variables in the reverse (adjoint) direction, shown as dashed lines.*

way to explore the posterior, and hence to characterize the uncertainty in parameters. (Besag et al., 1995; Gilks et al., 1996; Chen et al., 2000; Robert and Casella, 1999). MCMC effectively generates a sequence of model realizations, randomly drawn from the posterior distribution.

In this paper I focus on the calculation of the gradient of $\varphi(\boldsymbol{x}) = -\log(p(\boldsymbol{x}))$, where $p(\boldsymbol{x})$ is a probability density function (for example, the posterior) that depends on numerous continuous variables, represented by the vector $\boldsymbol{x}$, and on a few uses of this gradient in the context of Bayesian analysis. Since the gradient provides information about the local behavior of $\varphi(\boldsymbol{x})$, it should be extremely helpful in coping with multivariate analysis situations. For example, it is well known that the optimization of smoothly varying functions is achieved much more efficiently by algorithms that use function gradients than ones that do not. The technique of adjoint differentiation provides an efficient means to calculate the gradient, as will be explained below. One promising use of $\nabla\varphi$ is the hybrid technique for MCMC (Duane, 1987; Neal, 1996). Experience has shown that the hybrid Markov Chain Monte Carlo technique affords a robust means for sampling multidimensional probability density functions with high efficiency, provided one can calculate $\nabla\varphi$ in a time comparable to calculating the probability itself (Hanson, 2001).

2. ADJOINT DIFFERENTIATION

We wish to address problems that require minimizing a scalar function φ by varying the many (10^3 to 10^6 or more) variables that comprise the parameters in a simulation model. This optimization problem would be intractable without knowing the gradient of φ, or sensitivities, with respect to the parameters on which it depends. A technique to calculate these crucial sensitivities, called adjoint differentiation (Thacker, 1991), is apparently still relatively unappreciated. Using the adjoint differentiation technique, the derivatives with respect to all the parameters can be calculated in a time comparable to that for the forward calculation.

To illustrate the principles of adjoint differentiation, suppose that a calculation proceeds as a sequence of transformations, as shown in Fig. 1. The independent variables in the data structures, designated by the vector $\boldsymbol{x}$, are transformed in block A to produce the dependent variables $\boldsymbol{y}$. These variables are transformed in blocks B and C to produce

the dependent data structure $\boldsymbol{z}$ and the final scalar quantity φ, respectively. We call the sequence of transformations A $\rightarrow$ B $\rightarrow$ C $\rightarrow \varphi$ the forward calculation. We assume that the transformations are general, with the only restriction being that they are differentiable. Each transformation is self-contained; it requires only its input variables to calculate its output variables, for example, module B uses only its input $\boldsymbol{y}$ to calculate its output $\boldsymbol{z}$. Therefore, each transformation should require nothing more than its input to implement the derivative of its output variables with respect to its input variables. The data structures are likewise general.

The derivative of φ with respect to the ith component of $\boldsymbol{x}$ is obtained using the chain rule,

$$\frac{\partial \varphi}{\partial x_i} = \sum_{jk} \frac{\partial \varphi}{\partial z_k} \frac{\partial z_k}{\partial y_j} \frac{\partial y_j}{\partial x_i} \,. \tag{1}$$

Even if the transformations are nonlinear, this expression amounts to a product of matrices. The order of the summations can obviously be done in two different ways. If the sum over j is done before the sum over k, the calculation proceeds in the same direction as the forward model calculation. As the dimensions of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are assumed to be large, this sequence results in very large intermediate matrices (for example, with elements $\frac{\partial y_j}{\partial x_i}$), which we would like to avoid.

On the other hand, if the sum on k is done before that on j, the sequence of calculations is reversed. This sequence implies intermediate data structures (for example, with elements $\frac{\partial \varphi}{\partial y_j}$) that correspond to the forward data structures (for example, with elements y_j) implying storage requirements that are the same as those of the forward calculation. If the forward transformations are nonlinear, the values of the variables from the forward calculation may be required for the adjoint calculation. The backward flow of the adjoint derivatives is depicted in Fig. 1 by the dashed arrows.

The significant conclusion is that with adjoint differentiation the derivatives of a scalar quantity, φ in the above example, with respect to all the variables in the model can be computed in a time comparable to the forward calculation.

The decomposition of any complex calculation into components, as depicted in Fig. 1, can usually be accomplished at several different degrees of abstraction. The most basic level focuses on the sequence of actual CPU operations performed by the computer. Clearly, the sequence of operations can be described in terms of a very long sequence similar to that shown in the figure, with parallel side branches. Each CPU step consists of a simple arithmetic operation involving typically two inputs and one output. The process of adjoint differentiation is not too difficult to imagine at this level, although keeping track of the interconnectivity between inputs and outputs would seem to be a nightmare.

At a higher level of abstraction, one might focus on the forward computer code, written in whatever language is being used (FORTRAN, C, C++, Splus, Matlab, etc.). Hanson and Cunningham (1996) coined the acronym Adjoint Differentiation In Code Technique (ADICT) to describe a particular approach to adjoint differentiation. The distinctive feature of ADICT is that a separate computer code is used to compute the adjoint derivatives. That code is based on the simulation code with the explicit intent to "differentiate"

the forward calculation or numerical algorithm. For optimization of a functional based on computation, it is often desirable to have the gradient of the computation obtained by the ADICT principle, instead of the physics equations that the computation is supposed to approximate (Rightley et al., 1997).

It is often possible to deal with a forward calculation at an even higher level of abstraction, namely in terms of modules. In modern programming practice, codes are typically decomposed into subroutines or functions, which essentially are computational modules. If it is useful to do this for the forward calculation, it is doubly useful to use this modularization to create an adjoint calculation.

The Bayes Inference Engine (BIE) was created at Los Alamos to reconstruct geometrically-defined objects from their radiographs (Hanson and Cunningham, 1996). In the construction of the BIE, the adjoint code for each transformation module was manually developed on the basis of its forward code, or the specific algorithm it implements. This task can sometimes be formidable, but one that can be learned (Griewank, 2000). For example, such manual coding was employed to obtain the adjoint differentiation of a module that converted a surface, described in terms of triangles, to a 3D voxelated grid, which involves some fairly complex calculations (Battle et al., 1997). Object-oriented design and programming is tremendously advantageous for linking calculations together to form a data-flow diagram made of autonomous transformations, as in the BIE (Hanson and Cunningham, 1999). In a separate demonstration of the usefulness of adjoint differentiation, Hanson, Cunningham, and Saquib (1997, 1998b) were able to invert the diffusion equation for the diffusion of infrared light in a phantom representing biological tissue.

Adjoint differentiation of computer codes, and the more general technique involving differentiation of computer codes, have been pursued for several decades, as described in Thacker (1991) and Griewank (2000). It is remarkable that these modern techniques are not better known in the general fields of applied mathematics, simulation science, and statistics. However, the field of automatic differentiation of codes is actively being developed by a band of devoted researchers (Corliss et al., 2002; Griewank, 2000) and I believe that it will eventually have its due.

3. HYBRID MCMC ALGORITHM

In 1987 Duane et al. proposed a novel approach to performing MCMC, which is based on an analogy to a dynamical system of physical particles, such as a gas. The resulting algorithm is often referred to as hybrid MCMC because it alternates between Gibbs and Metropolis MCMC steps, as will be described shortly. Actually, the name 'hybrid MCMC' does not distinguish it from any number of other algorithms that employ a combination of Gibbs and Metropolis steps. For that reason, this algorithm might be better called 'Hamiltonian MCMC.'

A big advantage of hybrid MCMC is that large steps can be taken in the parameter space with only a small number of evaluations of φ and the gradient of φ. This algorithm and its refinements have been crucially important for realizing calculations critical to quantum field theory (Horowitz, 1991).

In hybrid MCMC, for each parameter x_i on which the target probability density function (pdf) $q(\boldsymbol{x})$ depends, an additional parameter p_i is introduced, which represents the parameter's associated momentum (Neal, 1996). A Hamiltonian is constructed as a sum of a potential-energy term, given by $\varphi = -\log(q(\boldsymbol{x}))$ and a kinetic-energy term:

$$H(\boldsymbol{x},\boldsymbol{p}) = \varphi(\boldsymbol{x}) + \sum_i \frac{p_i^2}{2m_i} \quad , \tag{2}$$

where m_i is a fictitious mass associated with the momentum p_i. Now the goal is to draw random samples from a new pdf that is proportional to $\exp(-H)$. This new pdf is proportional to the product of $q(\boldsymbol{x})$ and a factor involving the momentum variables, which are independent of the $\boldsymbol{x}$ variables. After a sequence representing samples $\{\boldsymbol{x}^k, \boldsymbol{p}^k\}$ from this new pdf is obtained, the set $\{\boldsymbol{x}^k\}$ by itself represents samples drawn from $q(\boldsymbol{x})$, by the marginalization property of MCMC sequences.

Each iteration of the algorithm starts with a Gibbs step to pick a new momentum vector from the uncorrelated normal distribution in the momenta corresponding to the second term in H. Then the trajectory in the $(\boldsymbol{x},\boldsymbol{p})$ space that maintains a constant H value is followed using the leapfrog technique, which consists of the following three substeps:

$$p_i(t+\frac{\tau}{2}) = p_i(t) - \frac{\tau}{2}\left.\frac{\partial\varphi}{\partial x_i}\right|_{\boldsymbol{x}(t)} \quad , \tag{3a}$$

$$x_i(t+\tau) = x_i(t) + \frac{\tau}{m_i}\, p_i(t+\frac{\tau}{2}) \quad , \tag{3b}$$

$$p_i(t+\tau) = p_i(t+\frac{\tau}{2}) - \frac{\tau}{2}\left.\frac{\partial\varphi}{\partial x_i}\right|_{\boldsymbol{x}(t+\tau)} \quad , \tag{3c}$$

where τ represents the time increment for the leapfrog step. The first and third updates in momentum are half time steps and have the effect of making the scheme accurate to second order in τ. After l leapfrog steps corresponding to a total trajectory time of $T = l\tau$, a Metropolis acceptance/rejection decision is made to guarantee that the sequence is in statistical equilibrium with q. This deterministic approach allows large steps in the parameter space to be taken with only a few evaluations of φ and the gradient of φ. The ability to take large steps is the essential feature of the hybrid method that makes it attractive. The Metropolis test is required to maintain detailed balance in the MCMC sequence, that is, to guarantee that the probability of moving from the starting position to the final position exactly equals the reverse jump.

Figure 2a displays five successive H trajectories for an isotropic, unit-variance two-dimensional normal distribution. The total time T for each trajectory is chosen to be abnormally large to display the shape of the H trajectories. Each H trajectory forms an ellipse, whose tilt and eccentricity are determined by the coordinates and momentum vector at its starting position. The momentum vector is randomly drawn from a 2D unit-variance normal distribution.

For cases like this, in which the target pdf is an isotropic normal distribution with the masses chosen to match the variance, every Hamiltonian trajectory comes back to its

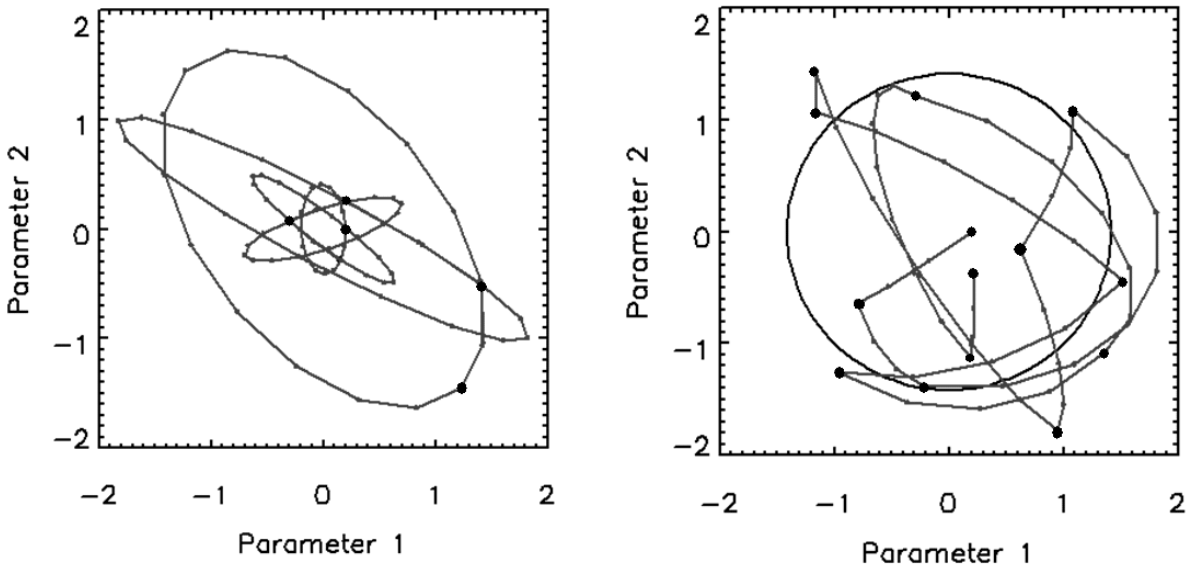

Figure 2. *Examples of the operation of the hybrid MCMC algorithm for a 2D unit-variance uncorrelated normal distribution. a) On left, five iterations with $\tau = 0.4$ and $T = 7.2$, showing the ellipsoidal nature of the Hamiltonian trajectories. The large dots indicate the ends of the H trajectories, and the smaller ones the endpoints of each leapfrog step. The value of T is much too large to efficiently generate MCMC samples. b) On the right, 12 iterations with $\tau = 0.4$ and T randomly drawn from a uniform distribution between 0 and $T_{max} = 2$, showing the kind of desirable behavior expected for the hybrid algorithm.*

starting point when T is a multiple of $2\pi = 6.28$. This observation indicates that certain values for the length of the Hamiltonian trajectories must be avoided in order to not get stuck in a cyclical pattern. To realize an adequate random sampling of $q(\boldsymbol{x})$, a reasonable approach is to randomize T by picking its value from a uniform distribution from 0 to T_{max} for each Hamiltonian trajectory. Figure 2b shows the resulting H trajectories for a 2D isotropic normal distribution.

The choices for τ, T_{max}, and m_i are clearly important for achieving the best efficiency. If τ in the leapfrog method is chosen to be too large, H will not be held constant enough, which will result in a high rejection rate in the Metropolis step. A good rule of thumb is that the length of the leapfrog steps should be kept smaller than the width of the $\boldsymbol{x}$-$\boldsymbol{p}$ distribution. On the other hand, it is desirable for the total length of the H trajectory, which is proportional to T, to be a large fraction of the width of the target pdf. As pointed out above, the values of T must be randomized. Thus, T_{max} should be chosen to produce H trajectories whose lengths are on the order of a few times the width of the target distribution. As with most other MCMC algorithms, one must explore the parameter space of the algorithm to optimize its efficiency for any particular application. It seems reasonable that one would like to pick the mass associated with each component that is about the same as the variance of the target distribution along that component in order to maintain circular trajectories in $\boldsymbol{x}$-$\boldsymbol{p}$ space. Because the variance of the target distribution is usually not known, one must guess a value, which may need to be adjusted

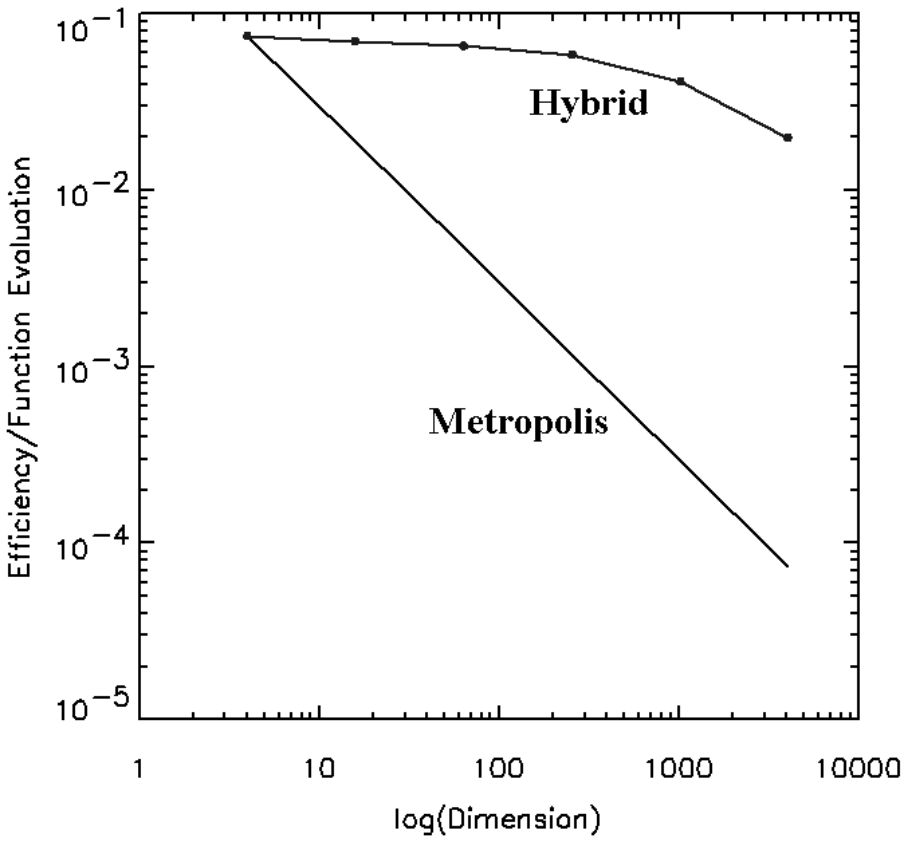

Figure 3. *Comparison of the efficiencies of the Metropolis and the hybrid MCMC algorithms for multidimensional isotropic normal distributions. It is assumed that the gradient calculation needed for the hybrid technique takes the same amount of time as the posterior calculation, which is often the case when adjoint differentiation is employed.*

later on. The parameters of the hybrid algorithm have not been optimized in this study in any detail, so the above advice should be taken as preliminary.

The number of function evaluations required by the leapfrog technique is easily counted. The first and third leapfrog substeps require the evaluation of $\nabla\varphi$. However, the gradient in the third step is at the same location as the beginning of the next leapfrog step, so long as there is no Metropolis rejection. The Metropolis test requires the evaluation of φ at the end of the trajectory. In the adjoint differentiation scheme described in the preceding section, this evaluation of φ is typically done as part of the gradient calculation. The bottom line is that l leapfrog steps will typically require l evaluations of φ and l evaluations of $\nabla\varphi$. From above, l is approximately given by T/τ, for which a value of around 8 is a reasonable estimate.

The potential usefulness of the hybrid MCMC algorithm has been confirmed by testing its performance on multivariate normal distributions. The important question is whether the hybrid method can avoid the decrease in efficiency for high dimensions, which has already been observed for the Metropolis method for isotropic normal distributions (Han-

son et al., 1998a). Figure 3 shows that the efficiency of the hybrid method remains high, even for quite large dimensions. The Metropolis efficiency is derived from the formula $\eta = 0.3/n$, which is appropriate for target pdfs consisting of normal distributions (Gelman et al., 1996; Hanson et al., 1998a). Per function evaluation, the efficiency of the hybrid method remains nearly constant at about 7% up to several hundred dimensions. The efficiency of the Metropolis algorithm starts at 7.5% at four dimensions and drops precipitously for higher dimensions. The hybrid method clearly provides superior performance at high dimensions. Incidentally, the average value of the sample variance for these runs is around 0.96, quite close to the actual value of 1.00.

The performance of the hybrid MCMC algorithm for correlated multivariate normal distributions has also been tested. Hanson (2001) found that for multivariate distributions with a high degree of correlation between variables, the efficiency of the hybrid MCMC was around 2% and it remained reasonably constant over the dimensional range of 16 to 128. It is worth noting that no adjustment of the algorithm was made to take into account the correlation structure of the target pdf. The performance of the Metropolis algorithm on the same target pdf for 16 dimensions (Hanson et al., 1998a) was very poor (0.11%) when an isotropic step trial distribution was used. Only after the step trial distribution was altered to match the covariance of the target pdf did it achieve an efficiency of 1.6%. Of course, the efficiency of Metropolis algorithm will decrease for high dimensions.

3. A CONVERGENCE TEST

Guaranteeing convergence of a sequence to the target pdf is a major concern in performing MCMC calculations (Gelman and Rubin, 1992; Gilks et al., 1996). The following convergence test has been suggested by Hanson (2001) for monitoring convergence in situations where $\nabla\varphi$ is available. The expression for the variance of $q(\boldsymbol{x})$ along any particular component x_i can be integrated by parts to obtain

$$\int_{-\infty}^{\infty}(x_i-\bar{x}_i)^2 q(\boldsymbol{x})\,d\boldsymbol{x} = \frac{1}{3}\int_{-\infty}^{\infty}(x_i-\bar{x}_i)^3 q(\boldsymbol{x})\,\frac{\partial\varphi(\boldsymbol{x})}{\partial x_i}\,d\boldsymbol{x} + \frac{1}{3}(x_i-\bar{x}_i)^3 q(\boldsymbol{x})\Big|_{-\infty}^{\infty}, \qquad (4)$$

where $\bar{x}_i$ is the first moment of $q(\boldsymbol{x})$ in the x_i direction and $q(\boldsymbol{x})$ is assumed to be defined over the full infinite interval.

The second term on the right-hand side of Eq. (4) can often be argued to be zero because $q(\boldsymbol{x})$ either drops off faster than $|\boldsymbol{x}|^3$ as $|\boldsymbol{x}|$ approaches ∞ or is symmetric for large $|\boldsymbol{x}|$. When that argument can be made, the two integrals are equal. But these integrals may be evaluated using the MCMC samples drawn from $q(\boldsymbol{x})$ and checked to see if they indeed are equal, to within sampling uncertainties.

For a sequence of samples $\{\boldsymbol{x}^k\}$, allegedly drawn from $q(\boldsymbol{x})$, the proposed test consists of computing the ratio

$$R = \frac{\sum_k (x_i^k-\bar{x}_i)^3\,\frac{\partial\varphi}{\partial x_i}\Big|_{\boldsymbol{x}^k}}{3\sum_k (x_i^k-\bar{x}_i^k)^2}\ , \qquad (5)$$

where $\bar{x}_i$ is the mean value of x_i, which can be estimated as the mean of the x_i^k samples. The above argument implies that R should be unity. Of course, as these quantities are

Monte Carlo estimates, they are subject to statistical fluctuations and R will fluctuate around unity, even for bona fide sample sets. Note that R generally need not be positive.

It may be argued that R will tend to be less than one when $q(\boldsymbol{x})$ is not adequately sampled by the MCMC sequence. The plausibility of this statement can be understood by considering the behavior of the numerator for a normal distribution. The derivative is proportional to $x_i - \bar{x}_i$ (taking $\bar{x}_i$ to be its actual value, not the sample mean) and so the numerator's sum is over $(x_i - \bar{x}_i)^4$. The value of R depends on the set of MCMC samples $\{\boldsymbol{x}^k\}$ from the target distribution $q(\boldsymbol{x})$. If the sequence fully spans the width of the target pdf, then R will come out to approximately unity. However, if the samples don't sufficiently reach the edges of the distribution, then the denominator will tend to be larger than the numerator because of its slower dependence on $x_i - \bar{x}_i$.

The above test does not reply on $q(\boldsymbol{x})$ having any particular functional form, as long as it drops smoothly to zero as $|\boldsymbol{x}|$ goes to infinity. On the other hand, the power of the test, in the sense of whether the uncertainty in R is small enough to determine when convergence is not reached, will depend on the functional behavior of $q(\boldsymbol{x})$. For example, if $q(x)$ is a 1D uniform distribution between two finite limits, the numerator will only get a nonzero contribution when the sample is at one of the limits and that contribution will be infinite.

The value of above test statistic is demonstrated using a two-dimensional anisotropic normal distribution with a standard deviation of 1 for x_1 and 4 for x_2. The hybrid MCMC algorithm is used to generate sample sets, although any MCMC algorithm could be used. The total length of each H trajectory is randomly chosen from a uniform distribution between 0 and T_{max}. To limit the extent to which the algorithm samples the full target pdf in this example, the H trajectories employ $\tau = 0.2$ and a rather small $T_{max} = 2$. The masses were both chosen to be unity, representative of a situation in which it is not known a priori that the variance of the target distribution is highly assymetric.

Figure 4 summarizes the properties of the test statistic given by Eq. (5) seen in 1000 runs of the hybrid MCMC algorithm for the situation described in the previous paragraph. We observe that R provides a reasonably good indication of how well the MCMC algorithm has sampled the target pdf. The behavior is different for the two components. The mean estimated variance for the first component is estimated to within 2% of its correct values (1) for just 80 Hamiltonian steps, at which point $R = 0.90 \pm 0.27$. On the other hand, for 80 steps, the estimated variance for the second component is only about half of what it should be (16) and $R = 0.43 \pm 0.24$. Even at 640 steps, the variance of the second component is on the average 4% too low. The test statistic has almost reached unity; $R = 0.87 \pm 0.26$. We conclude that the proposed test statistic provides a useful guide as how well the target distribution has been sampled.

4. CONCLUSION

I have described the technique of adjoint differentiation of computer codes. It provides a useful means for efficiently calculating the gradient of a scalar function derived from a forward simulation, for example, to obtain a Bayesian posterior functional. Adjoint differentiation is one of the modern computational-science techniques that can facilitate

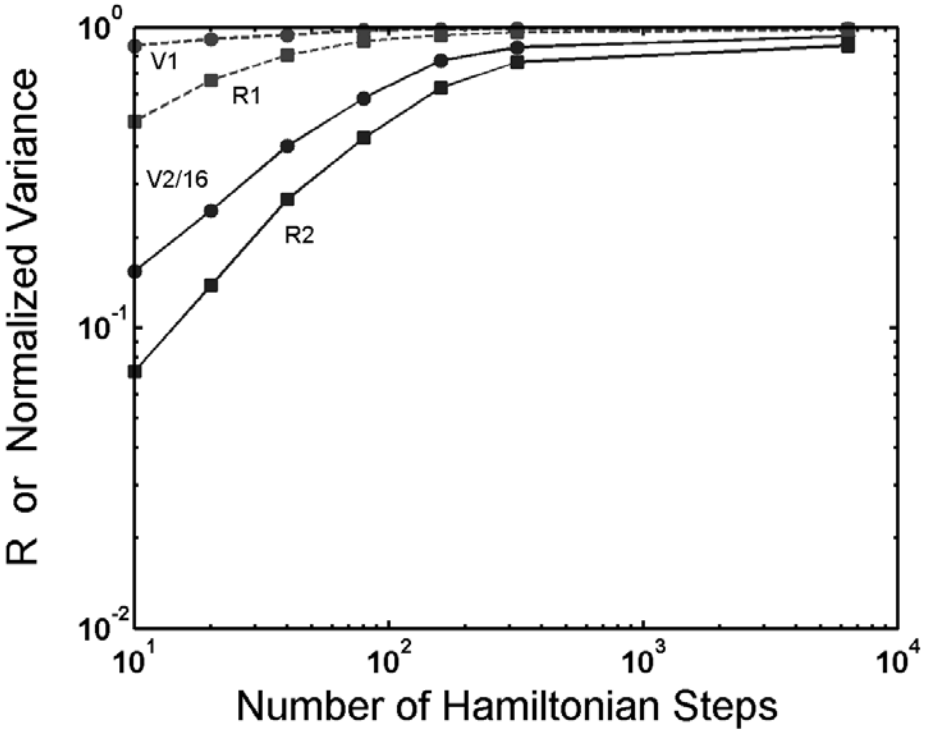

Figure 4. *The average values of the test statistic R and the average variance V obtained from sequence values generated by the hybrid MCMC algorithm. The sequences are generated for variable numbers of Hamiltonian steps. The average for each plotted point is based on 1000 MCMC sequences. The target pdf is a 2D normal distribution with a variance of unity in the first component (dashed lines) and 16 in the second (solid lines), with no correlation between the two components. The test statistic falls below unity when the sample variance is less than the actual variance of the target pdf, indicating that reduced coverage of the full width of the pdf tends to reduce the value of R.*

Bayesian inference regarding scientific simulations. Some aspects of the hybrid MCMC algorithm have been explored. The efficiency of hybrid MCMC, in terms of function evaluations, has been shown to be almost independent of the number of parameters, and is superior to that of the Metropolis algorithm for high dimensions. A new convergence test has been proposed, which employs the gradient of the minus-log-probability of the target pdf. This test may provide a useful means of confirming the convergence of an MCMC sequence to a target pdf.

REFERENCES

Battle, X. L. et al. (1997). 3D tomographic reconstruction using geometric models. *Medical Imaging: Image Processing, Proc. SPIE* 3034 (Hanson, K. M., ed.). 346–357.

Besag, J. et al. (1995). Bayesian computation and stochastic systems. *Statist. Sci.* **10**, 3–66.

Chen, M.-H. et al. (2000). *Monte Carlo Methods in Bayesian Computation.* New York: Springer.

Corliss, G. et al. eds. (2002). *Automatic Differentiation of Algorithms.* New York: Springer

Duane, S. et al. (1987). Hybrid Monte Carlo. *Phys. Lett. B* 195, 216–222.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences (with discussion). *Statist. Sci.* **7**, 457–511.

Gelman, A. et al. (1996). Efficient Metropolis jumping rules. *Bayesian Statistics 5* (J. M. Bernardo, J. O. Berger, A. P. Dawid and A. F. M. Smith, eds.). Oxford: University Press.

Giering, R. (1997). Tangent linear and Adjoint Model Compiler, *Tech. Rep.*, TAMC 4.7, Max-Planck-Institut für Meteorologie.

Gilks, W. R. et al. (1996). *Markov Chain Monte Carlo in Practice*. London: Chapman and Hall.

Griewank, A. (2000). *Evaluating Derivatives*. Philadelphia: SIAM.

Hanson, K. M. and Cunningham, G. S. (1996). A computational approach to Bayesian inference. *Computing Science and Statistics* 27 (Meyer, M. M. and Rosenberger, J. L., eds.). Fairfax Station: Interface Foundation, 202–211.

Hanson, K. M. and Cunningham, G. S.(1998a). Posterior sampling with improved efficiency. *Medical Imaging: Image Processing, Proc. SPIE* 3338 (Hanson, K. M., ed.). 371–382.

Hanson, K. M., Cunningham, G. S. and Saquib, S. S. (1998b). Inversion based on computational simulations. *Maximum Entropy and Bayesian Methods* (Erickson, G. J. et al., eds.). Dordrecht: Kluwer, 121–135.

Hanson, K. M. and Cunningham, G. S. (1999). Operation of the Bayes Inference Engine. *Maximum Entropy and Bayesian Methods* (von der Linden, W. et al., eds.). Dordrecht: Kluwer, 309-318.

Hanson, K. M. (2001). Markov Chain Monte Carlo posterior sampling with the Hamiltonian method. *Medical Imaging: Image Processing, Proc. SPIE* 4322 (Sonka, M. and Hanson, K. M., eds.). 456–467.

Hanson, K. M., Booker, J. M., and Hemez, F. M.. (2002). "Synopsis of a Workshop on Quantification of Uncertainties in Physics Simulations," *Los Alamos Report LA-UR-02-7331*.

Horowitz, A. M. (1991). A generalized guided Monte Carlo algorithm. *Phys. Lett. B* 268. 247–252.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. New York: Springer.

Rightley, M. L. J. et al. (1997). Adjoint differentiation of hydrodynamic codes. *CNLS Newsletter*, Center for Nonlinear Studies, Los Alamos National Laboratory.
(http://cnls.lanl.gov/Publications/highlights.html).

Robert, C. P. and Casella, G. (1999). *Monte Carlo Statistical Method*. New York: Springer.

Saquib, S. S. et al. (1997). Model-based image reconstruction from time-resolved diffusion data. *Medical Imaging: Image Processing, Proc. SPIE* 3034 (Hanson, K. M., ed.). 369–380.

Thacker, W. C. (1991). Automatic differentiation from an oceanographer's perspective. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (Griewank, A. and Corliss, G. F., eds.). Philadelphia: SIAM, 191–201.

ACKNOWLEDGEMENTS

I would like to thank the following people for their help in understanding MCMC and its potential usefulness: Greg Cunningham, Frank Alexander, David Haynor, David Higdon, Julian Besag, Malvin Kalos, Radford Neal, John Skilling, James Gubernatis, and Richard Silver. This work was supported by the United States Department of Energy under contract W-7405-ENG-36.