Parallel Wiener integral methods for elliptic boundary value problems: A tale of two architectures *

Michael Mascagni
Center for Computing Sciences, I.D.A.
(formerly known as Supercomputing Research Center)
17100 Science Drive
Bowie, Maryland 20715-4300
mascagni@super.org

July, 1990, updated August, 1995

1 Introduction

One of the most intriguing aspects of linear elliptic boundary value problems (BVPs) is their relationship to probability. The discovery of this relationship dates back to the beginnings of rigorous measure theory, or more specifically when measure theorists began considering how to place measures on spaces of continuous functions. The measures placed on these infinite dimensional spaces are first defined for simple sets of continuous functions with the help of the fundamental solution of certain linear parabolic partial differential equations (PDEs). Once these simple "cylinder sets" of continuous functions can be measured, it is rather easy to extend the measure to the entire space of continuous functions with standard techniques from measure theory. As

*Appears in slightly edited form in *SIAM News*, V.23, N.4, July, 1990. This work was done in the Mathematical Research Branch of the National Institute of Diabetes, Digestive, and Kidney Diseases of the National Institutes of Health, Bethesda, Maryland while the author was a National Research Council-NIH Research Associate.

a consequence of this construction, certain integrals with respect to these measures (which can be thought of as mathematical expectations with the measure thought of as a probability density) are solutions to particular linear parabolic and elliptic problems.

A curiosity of looking at a space of continuous functions with this measure is that almost all (full measure) of the continuous functions are nowhere differentiable[1]. Because of this fact, continuous functions with this measure (called Wiener measure) are almost all extremely jagged and monstrously wiggly. Thus it is straight forward to associate these spaces of continuous functions under Wiener measure with spaces of continuous Brownian motion paths. This makes perfect sense when one recalls that diffusion can be thought of as the macroscopic manifestation of microscopic Brownian motion, and that parabolic PDEs (like the diffusion equation) play a fundamental role in the construction of these Wiener measures. It turns out that this probabilistic theory for representing the solutions of linear elliptic and parabolic PDEs has many applications in analysis [2] [4], and as we will see below, in numerical computation as well [1] [3].

As a simple example of the application of these ideas to computation let us consider the Dirichlet BVP for the Laplace equation:

$$-\Delta u(x) = 0, \quad x \in \Omega, \quad u(x) = g(x), \quad x \in \partial\Omega \tag{1}$$

The probabilistic representation of equation 1, often called the Wiener integral representation, is denoted by:

$$u(x) = E_x \left[g(\beta(\tau_{\partial\Omega})) \right]. \tag{2}$$

The interpretation of equation 2 is that the solution of equation 1 at an interior point, x, is the expectation of the boundary value at the first hitting location of the sample path $\beta(\cdot)$ started from x. The Markov time, $\tau_{\partial\Omega}$, is the time at which a sample path first encounters the boundary, and is called the mean first passage time. This quantity is defined for the sample path $\beta(\cdot)$ by $\tau_{\partial\Omega} = \inf_{\beta(t)\in\Omega} t$.

An alternate interpretation of equation 2 is as a probabilistic version of the traditional Green's function representation of the solution to equation 1.

[1] Recall the fuss created by Weierstrass's construction of a single continuous, nowhere differential function by Fourier series.

This is because equation 2 is an integral of the boundary values against a boundary mass, $p(x, y)$, the probability of a sample path starting at x and first encountering the boundary at y. It is an elementary fact that $p(x, y)$ so defined is the Green's function of equation 1 [1]. If we think in terms of Brownian motion, which is intimately related to the Laplacian, then equation 2 states that the solution to equation 1 is the expected value of the first hitting boundary value of a Brownian motion started at x.

It is rather easy to see that the Wiener integral in equation 2 solves the Dirichlet problem for the Laplace equation. A function is a solution to equation 1 if (i) it has the mean value property and (ii) it has the correct boundary values. In two dimensions a function has the mean value property if its value at the center of a circle is the average of the function on the circle. Pick a point, x, in the interior of Ω, and using x as the center draw a circle (call it C) totally within Ω. Equation 2 states that the $u(x)$ is the expected boundary value at the point of first passage. By continuity, any path started at x will encounter C before hitting the boundary. Thus $u(x)$ is the expected first passage boundary value of walks started on C conditional on where you first hit C from walkers started at x. Since Brownian motion is isotropic, a Brownian path started at x will first encounter any point on C with equal probability. Thus $u(x)$ is the average of the first passage boundary values from walks started on C (which are the values of $u(x)$ on C by equation 2) and so $u(x)$ has the mean value property. In addition, if the boundary is smooth, we can see that $u(x)$ takes on the appropriate boundary values by letting x approach the boundary while using the above argument.

This functional integration approach can be used to solve discretizations of these continuous problems by utilizing random walks in place of Brownian motion. Through this formalism, and extensions of the probability to different elliptic PDEs and different BVPs, a large class of Monte Carlo methods for these problems emerge. Implementation of these random walk based Monte Carlo methods on multiple instruction multiple data (MIMD) and single instruction multiple data (SIMD) machines will be considered in the subsequent section. It will be shown how different implementations lead to different algorithms which in turn lead to different practical and analytic considerations.

2 Architecture and Implementation

Given that we wish to implement algorithms related to these probabilistic ideas on a parallel computer, it is incumbent on us to consider what aspects of these algorithms we wish to exploit in a parallel implementation. In random walk based algorithms there are two natural ways to use parallelism based on certain replicated aspects of random walks. Since the discrete versions of these algorithms are all based on collecting statistics from random walks over some grid, it is natural to associate processing elements for a parallel implementation with either the walkers or the places they walk, i.e., the grid points.

In some sense, the mapping of groups of walkers onto parallel processors is the most natural parallel decomposition, and is readily mapped onto a MIMD machine. The second mapping, that of grid points to processors, maps very naturally onto massively parallel SIMD computers. Below we discuss how the choice of one mapping over another influences the details of the algorithm and the performance aspects of the implementations. To make our discussion more concrete, let us think about MIMD implementations on either a shared memory MIMD machine (like the Cray C90) or a distributed memory machine like the IBM SP2. For the massively parallel SIMD implementation let us keep the rather old Thinking Machines CM-2 or the current Maspar MP-2 in mind.

The choice of mapping groups of random walkers onto parallel processors naturally leads us to the following algorithm for the evaluation of first passage time statistics like those required in equation 2. Each processor starts with random walkers with different starting locations on the grid. During each iteration all the walkers take a random step on the grid. Those that encounter the boundary are removed and their starting locations are scored with the boundary value of the first passage location, and new walkers are started somewhere on the grid to replace them. It is obvious that this algorithm faithfully implements the collection of statistics implied in equation 2 in an "embarrassingly" parallel fashion. In fact, this algorithm is such that it may be implemented asynchronously on independent processors until it becomes necessary to gather the statistics from each independent processor into centralized memory locations. It also makes little difference if we implement this algorithm on a shared or distributed memory machine (or a loosely coupled group of workstations) since there is no interprocessor communication

until the statistics are centrally collected.

This idea of exploiting the parallel nature of independent statistical sampling arising from certain Monte Carlo calculations is an old one, and quite easy to implement in the case of the Dirichlet BVP for the Laplace equation. The nature of this algorithm also makes the choice of a stopping criterion rather simple. Since we are statistically sampling the solution to a problem which has a well behaved underlying stochastic process, the computational error, which is only due to the statistical sampling error, should have law–of–large–number behavior. Thus if we have p processors, each of which samples the solution at every grid point, and we desire an overall variance in the sampling error of size ϵ^2, then we may run the p independent processes until all of them achieve at least an $\epsilon^2 p$ sampling variance. Then when we accumulate the p independent samples from the processors, we will be left with an overall variance of no greater than $(\epsilon^2 p)/p$, by the addition of variance. Finally, we consider the cost of generating each sample. Since our algorithm advances walkers from the interior until they hit the boundary, we average one sample every $\tau_{\partial\Omega}$ (the average length of a walk) iterations per walker. We will refer to this algorithm as the MIMD or "forward" random walk algorithm.

In contrast, the choice of mapping grid points onto the processor of a massively parallel SIMD machine, such as the CM-2 or MP-2, leads to a different set of considerations. In fact, the algorithm described above is an extremely poor choice on a SIMD machine. This is due to the fact that the CM-2 and MP-2 are physically an array of processors, each with local memory, upon which two types of interprocessor communication are implemented. General interprocessor communication is implemented via the "router" which is a large multiple slower than comparable communication over nearest neighbor connections. This later is called "NEWS" communication. Thus the task of communicating the boundary value at the first hitting location back to the walker's starting point, which will generally require the "router", seriously degrades the above "forward" random walk algorithm's performance. Because of this difference between "NEWS" and "router" performance, it is worthwhile to consider a variation on the above algorithm which abolishes the need for "router"–based communication.

If we assume that the grid points in our calculation are such that they can be embedded into a regular d–dimensional grid, then all nearest–neighbor communication on this grid can be implemented via the fast "NEWS" communication on a CM-2. If $d = 2$, this is also true on an MP-2. A sim-

ple variation of the "forward" random walk algorithm allows us to get by with only nearest–neighbor communication. If, when generating the random walks from the starting interior points to the boundary, we save the path taken through the grid, this path can be retraced to bring the boundary values into the interior via nearest–neighbor only communication. In reality, this improves the situation very little, as random walks generate extremely suboptimal routes from the boundary to given interior points. It is, however, the case that while retracing walks from a given boundary point, every grid point along the retraced path may be considered as the starting point of a new "forward" random walk which first encountered the boundary at the given boundary location. In fact, it can be proven that scoring the boundary value at each point in the retraced path is probabilistically equivalent to the "forward" random walk algorithm discussed above [3]. In addition, retracing has the advantage that we obtain one sample per walker per step. Finally, it should be obvious that the notion of retracing is superfluous, as it is more efficient to start our walkers at the boundary.

Thus by trying to avoid an extremely inefficient aspect of a particular parallel computer's design, we have been led to a variation on our original algorithm. This "backward" random walk algorithm is an improvement over the original "forward" random walk algorithm in two obvious ways: (1) it requires only nearest–neighbor communication on the computational grid, and (2) it generates samples at the rate of one per walker per iteration instead of one per walker per complete random walk ($O(\tau_{\partial\Omega})$). A not so obvious difference is based on the fact that on a SIMD machine, the MIMD rationale for the design of a stopping rule is not at all applicable. When we consider a more reasonable stopping rule for a SIMD implementation we will encounter yet another advantage of the "backward" over the "forward" random walk algorithm.

In a SIMD implementation, it makes much more sense with the "backward" random walk algorithm to start off a large cohort of walkers from the boundary with their boundary values, and then after some number of iterations terminate all of the walking and compute the statistics. This stopping rule makes more sense than waiting for an acceptable level of variance at each grid point, as was suggested for the MIMD implementation. This is because in the "backward" algorithm we are specifying the end not the beginning of random walks, and so starting new walkers will not necessarily reduce the sampling error at specified interior grid points. Since it is more natural in

the SIMD case to consider the termination of all the walks uniformly and accumulating statistics at that point, one must be able to calculate the effect this has on the evaluation of the Wiener integral in equation 2. This effect is precisely due to the fact that by placing a limit on the number of iterations in the "backward" random walk algorithm, we are sampling the random walk expected value in equation 2 over only a portion of the entire space of random walks possible on our grid. We are evaluating equation 2 over only the space of random walks up to a given length equal to the number of iterations before termination. Fortunately, this truncated expected value can be explicitly computed. Surprisingly it is a nonlinear object.

It has been shown that this expected value over the space of random walks up to a given length can be computed as the *quotient* of two Jacobi method solutions of related discrete Dirichlet BVPs for the Laplace equation [3]. In the simple case the discrete Laplacian on the two dimensional square with a uniform grid, this statistic has an expected value which is the quotient of the Jacobi solution of the BVP with the given boundary values over the Jacobi solution with unit boundary values. Since unit boundary values asymptotically yield the constant unit function solution, the asymptotic behavior is that of the ordinary Jacobi method. However, for small iteration numbers, this quotient gives remarkably good empirical convergence results as demonstrated in the comparative figure below (Figure 1).

As figure 1 shows, the "backward" random walk algorithm has initial convergence behavior comparable to the method of successive over–relaxation (SOR) with optimal relaxation parameter.

3 Concluding Comments

It is well known that these Monte Carlo methods are much inferior to many deterministic methods for these types of problems. However, in very high dimensions variants of these Monte Carlo methods are often used to solve problems in quantum mechanics. In addition, the Monte Carlo methods often serve to motivate the design and analysis of deterministic analogs which may offer some unique advantages over more conventional algorithms. Another property of these Monte Carlo algorithms that may prove useful in real computations is the fact that with them one may sample the solution at as few as one grid point.

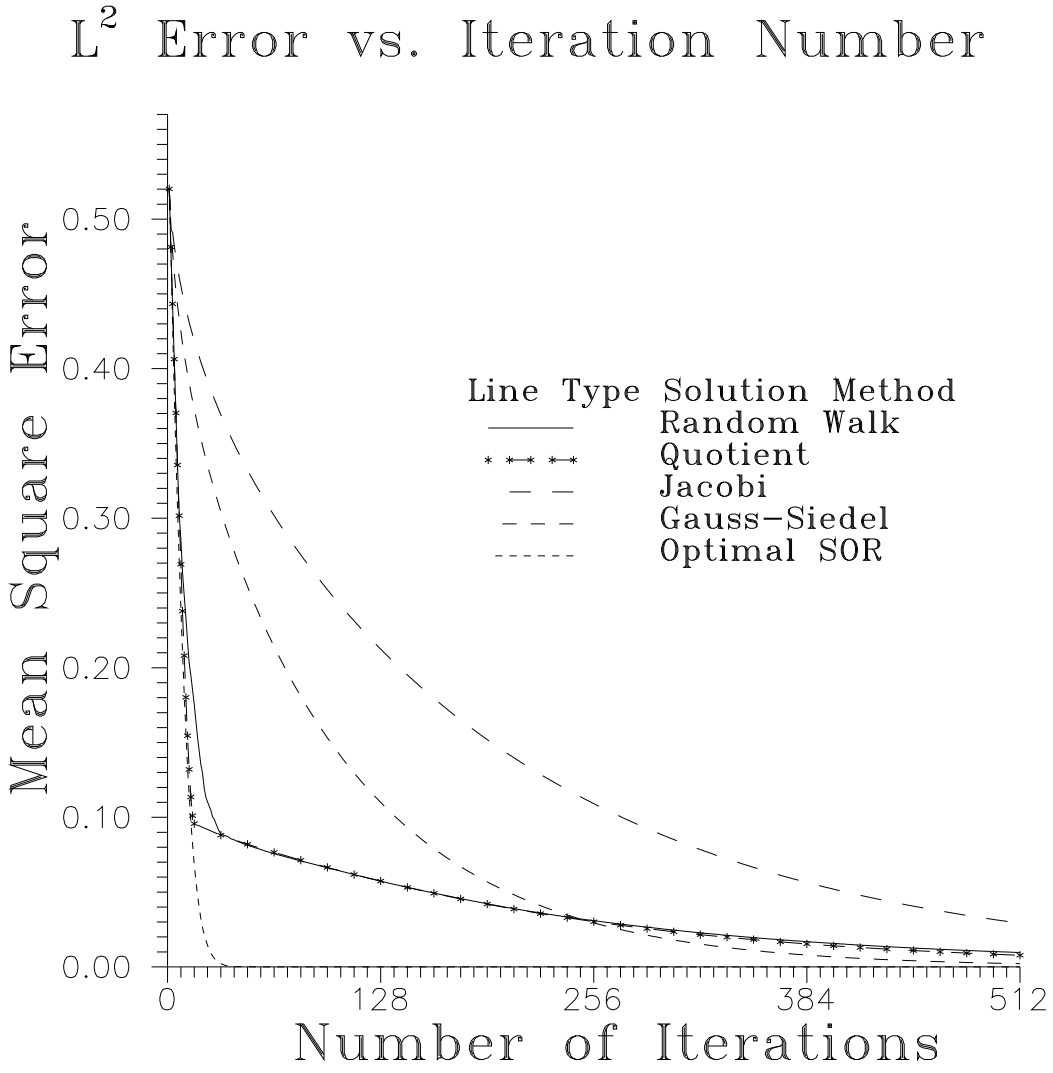

Figure 1: Empirical comparison of the Jacobi method, red–black Gauss–Seidel method, optimal red–black SOR method, the "backward" random walk method, and the nonlinear quotient method (which is the expected value of the "backward" method). For the random walk method, iteration number refers to maximal length of random walks.

We have seen a rather simple example of how implementing a given mathematical formulation for a particular problem on different types of parallel computers not only leads to different implementations, but also to different questions of the numerical analysis. The MIMD "forward" random walk implementation of these Wiener integral representations naturally motivates a stopping rule based on a sampling error tolerance. The SIMD "backward" implementation makes this type of stopping rule awkward, and leads to the idea of numerically evaluating Wiener integrals over certain natural truncations of the space of all random walks. The analysis of these SIMD inspired truncations lead to a nonlinear iterative method that is based on Jacobi iterations (and hence can be implemented in parallel without grid point coloration) and gives initial behavior similar to optimal red–black SOR, without having to choose a relaxation parameter. Thus we have an object lesson on how parallel architectures can influence not only the design, but the analysis of numerical algorithms.

References

[1] R. Courant, K.O. Friedrichs, and H. Lewy, *Über die partiellen Differenzengleichungen der mathematischen Physik*, Math. Ann., 100(1928), pg. 32–74.

[2] M. Freidlin, *Functional Integration and Partial Differential Equations*, Princeton University Press, Princeton, New Jersey, 1985.

[3] M. Mascagni, *High dimensional numerical integration and massively parallel computing*, Contemporary Mathematics, 115(1991), pg. 53–73.

[4] J. Stroock and R. Varadhan, *Multidimensional Diffusion Processes*, Springer–Verlag, New York, Berlin, 1976.