Numerical Pricing of Derivative Claims: Path Integral Monte Carlo Approach

Miloje S. Makivić*
(November 28, 1994)
NPAC Technical Report SCCS 650

Abstract

We propose a path integral Monte Carlo method for pricing of derivative securities. Metropolis algorithm is used to sample probability distribution of histories (paths) of the underlying security. The advantage of path integral approach is that complete information about the derivative security, including its parameter sensitivities is obtained in a single simulation. It is also possible to obtain results for multiple values of parameters in a single simulation. The algorithm is efficiently implemented on parallel machines using High-Performance Fortran. **
Keywords: Derivative securities, option pricing, Monte Carlo, path integrals, data parallel, High-Performance Fortran

---

*Northeast Parallel Architectures Center (NPAC), Syracuse University,
111 College Place, Syracuse, NY 13244-4100
e-mail: miloje@npac.syr.edu
Mosaic http://www.npac.syr.edu/users/miloje/
The author acknowledges partial support from Center for Research in Parallel Computation.

Numerical Pricing of Derivative Claims: Path Integral Monte Carlo Approach

Introduction

Monte Carlo simulation is an established numerical tool for pricing of derivative securities, which is particularly useful for complex valuation problems (see Boyle (1977)). Flexibility of Monte Carlo simulation is an important advantage in comparison with other methods, which are usually faster. For example, it is the only method which can incorporate empirical probability distributions of the underlying security. It is also the only method capable of pricing history dependent claims.

Standard Monte Carlo approach, introduced by Boyle (1977), relies on direct stochastic integration of the underlying Langevin equation. Given the security price at an instant in time, a new price for the next instant is generated at random according to the stochastic process of the security. Results are obtained by averaging over a large number of realizations of this process.

In this paper we propose an improved Monte Carlo method for the problem of pricing of derivative securities, which is based on the probability function for the *complete* history of the underlying security. In our current approach, this probability function is generated using the algorithm of Metropolis et al. (1953). Other methods may be used as well (for a good review, see Negele and Orland (1988)), but Metropolis algorithm is likely to have the smallest variance. Main advantages of the path integral approach are: (1) partial derivatives of the price with respect to all of the input parameters can be computed in a *single* simulation, (2) results for *multiple sets* of parameters can be computed in a single simulation, and (3) suitability for implementation on a parallel or distributed computing environment.

I. PATH INTEGRAL MONTE CARLO

Monte Carlo methods are based on the risk-neutral valuation approach of Cox and Ross (1976). Consider a derivative security which depends on an N-dimensional vector of state variables, $\mathbf{\Theta} = (\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_N)$. State vector $\mathbf{\Theta}$ is assumed to follow an arbitrary Markov process. Current time is set to $t = 0$ and the expiration date of the contract is at $t = T$. It will be assumed for the sake of simplicity that the contract is European. American contracts bring additional technical difficulties and are left for a forthcoming publication.

Riskless short-term interest rate at time t will be denoted by $r(t)$. Risk-neutral probability density of the final state vectors at time T is given by the conditional probability distribution $P(\mathbf{\Theta}(T)|\mathbf{\Theta}(0))$. Let $t_i, i = 1, ..., M$ denote a set of intermediate time slices, such that $0 < t_1 < t_2 < ... < t_M < T$. To simplify notation, we will denote these time slices using their indices only, i.e. $t_i \equiv i$. Application of the Chapman-Kolmogorov equation for Markov processes (Papoulis (1984)) leads to a recursive determination of P:

$$P(\mathbf{\Theta}(T)|\mathbf{\Theta}(0)) = \int d\mathbf{\Theta}(M) \cdots \int d\mathbf{\Theta}(2) \int d\mathbf{\Theta}(1) P(\mathbf{\Theta}(T)|\mathbf{\Theta}(M)) \cdots P(\mathbf{\Theta}(1)|\mathbf{\Theta}(0)) \tag{1}$$

We will call the collection of state vectors

$$\mathbf{\Omega} = (\mathbf{\Theta}(0), \mathbf{\Theta}(1) ... \mathbf{\Theta}(T)) \tag{2}$$

a "path" followed by the state variables. For any finite number of time slices, a path may be regarded as a point in a $\mathbf{R}^{(M+2)\times N}$ space, which will be called the "path space". The payoff of the derivative security, $F(\mathbf{\Omega})$, is a real-valued function on the path space. We will use the following shorthand notation for the multiple integral:

$$\int d\mathbf{\Theta}(T) \cdots \int d\mathbf{\Theta}(M) \int d\mathbf{\Theta}(2) \int d\mathbf{\Theta}(1) = \int D\mathbf{\Omega} \tag{3}$$

With these conventions, the valuation formula for the price Q of a European contract with payoff function, can be written in a "path integral" notation:

$$Q = \int D\mathbf{\Omega} F(\mathbf{\Omega}) P(\mathbf{\Omega}) \exp\left\{ -\int_0^T r(t) dt \right\} \tag{4}$$

where, by definition, probability of a path is:

$$P(\mathbf{\Omega}) = P(\mathbf{\Theta}(T)|\mathbf{\Theta}(M)) P(\mathbf{\Theta}(M)|\mathbf{\Theta}(M-1)) .. P(\mathbf{\Theta}(2)|\mathbf{\Theta}(1)) P(\mathbf{\Theta}(1)|\mathbf{\Theta}(0)) \tag{5}$$

Path probability is expressed in terms of a product of probabilities associated with "short term" segments (time slices) of a path. By tuning the number of time slices one can simulate arbitrary Markov processes with desired accuracy.

Jump processes (see Merton (1976)), non-stationary stochastic processes or empirical probability distributions of security returns can be readily incorporated into the path integral framework.

The interpretation of equation 4 is straightforward: one is summing over payoffs $F(\mathbf{\Omega})$ of all possible paths of the state vector from the beginning of the contract until its expiration, weighted by the probability $P(\mathbf{\Omega})$ of occurence of a particular path. The basic idea of a Monte Carlo approach is that this summation can be performed stochastically by generating paths at random and accumulating their payoffs. Major contributions to the integral come from relatively small parts of the otherwise huge path space. Therefore, to evaluate the multidimensional integral 4 efficiently, it is essential to sample different regions of path space according to their contribution to the integral, *i.e.*, to perform importance sampling (Hammersley and Handscomb (1964)). If a discrete set of L paths $\mathbf{\Omega}_\nu$, $\nu = 1, \ldots, L$, is drawn according to its probability of occurrence, $P(\mathbf{\Omega})$, the integral 4 may be approximated by:

$$< Q >_{MC} = \frac{1}{L} \sum_{\nu=1}^{L} Q(\mathbf{\Omega}_\nu) \tag{6}$$

and the error can be controlled by increasing the size of the sample, since, for large sample sizes, the central limit theorem assures that

$$< Q >_{MC} = Q + O(L^{-1/2}) \tag{7}$$

II. METROPOLIS ALGORITHM

Before we describe the advantages of promoting complete paths to be the fundamental objects of a Monte Carlo simulation, we will describe Metropolis method for generating the probability distribution of the paths, to be able to take advantage of importance sampling. Readers who are familiar with the method may wish to skip this Section. Metropolis method constructs a Markov process in the path space, which asymptotically samples the path probability distribution. This process is not related to the Markov process that governs the evolution of the state variables. Being a formal device to obtain the desired distribution, there is a lot of freedom in constructing this process, which will prove advantageous for variance reduction techniques.

The Markov process will be defined by the transition probability $W(\mathbf{\Omega}_1 \to \mathbf{\Omega}_2)$, which denotes the probability of reaching point $\mathbf{\Omega}_2$ starting from $\mathbf{\Omega}_1$. There are two restrictions on the choice of the transition probability W. First, the stochastic dynamics defined by W must be *ergodic*, i.e. every point in the path space must be accessible. The second requirement is that the transition probability must satisfy the "detailed balance condition":

$$P(\mathbf{\Omega}_1) W(\mathbf{\Omega}_1 \to \mathbf{\Omega}_2) = P(\mathbf{\Omega}_2) W(\mathbf{\Omega}_2 \to \mathbf{\Omega}_1) \tag{8}$$

These two restrictions do not specify uniquely the stochastic dynamics. We will use the transition probability proposed by Metropolis *et al.* (1953), which is known as the Metropolis algorithm:

$$W(\mathbf{\Omega}_1 \to \mathbf{\Omega}_2) = \begin{cases} P(\mathbf{\Omega}_2)/P(\mathbf{\Omega}_1)) & \text{if } P(\mathbf{\Omega}_1) \geq P(\mathbf{\Omega}_2) \\ 1 & \text{if } P(\mathbf{\Omega}_1) < P(\mathbf{\Omega}_2) \end{cases} \tag{9}$$

We now outline a proof given in Negele and Orland (1988) that this Markov chain will asymptotically sample the desired distribution $P(\mathbf{\Omega})$. If we start from an initial probability distribution $P_0(\mathbf{\Omega})$, then the probability distribution after n steps of the Markov chain will be denoted by $P_n(\mathbf{\Omega})$. Probability distributions at successive Markov steps, n and $n+1$, satisfy the following relationship:

$$P_{n+1}(\mathbf{\Omega}_{n+1}) = \int D\mathbf{\Omega}_n W(\mathbf{\Omega}_n \to \mathbf{\Omega}_{n+1}) P(\mathbf{\Omega}_n) \tag{10}$$

$P(\mathbf{\Omega})$ is the fixed point distribution of this Markov process. Substitution of $P(\mathbf{\Omega})$ for $P_n(\mathbf{\Omega})$ in the above equation, combined with the detailed balance condition, implies that $P_{n+1}(\mathbf{\Omega}) = P_n(\mathbf{\Omega}) = P(\mathbf{\Omega})$. One also has to show that the distribution converges towards $P(\mathbf{\Omega})$. A simple measure of deviation from the desired distribution is: $D_n = \int D\mathbf{\Omega} |P_n(\mathbf{\Omega}) - P(\mathbf{\Omega})|$. Deviation decreases as one goes further along the Markov chain:

$$D_{n+1} = \int D\mathbf{\Omega}_{n+1} \left| \int D\mathbf{\Omega}_n W(\mathbf{\Omega}_n \to \mathbf{\Omega}_{n+1}) P_n(\mathbf{\Omega}_n - P(\mathbf{\Omega}_{n+1}) \right|$$

$$D_{n+1} = \int D\mathbf{\Omega}_{n+1} \left| \int D\mathbf{\Omega}_n W(\mathbf{\Omega}_n \to \mathbf{\Omega}_{n+1}) [P_n(\mathbf{\Omega}_n - P(\mathbf{\Omega}_n)] \right|$$

$$D_{n+1} \leq \int D\mathbf{\Omega}_{n+1} \int D\mathbf{\Omega}_n W(\mathbf{\Omega}_n \to \mathbf{\Omega}_{n+1}) \left| P_n(\mathbf{\Omega}_n - P(\mathbf{\Omega}_n) \right| = D_n$$

Therefore, $P(\mathbf{\Omega})$ is the asymptotic probability distribution of the points generated by this random walk,

$$P(\mathbf{\Omega}) = \lim_{n\to\infty} P_n(\mathbf{\Omega}) \tag{11}$$

One can view the evolution of the original probability distribution along the Markov chain as a relaxation process towards the "equilibrium distribution", $P(\mathbf{\Omega})$. In practice, one assumes that the relaxation occurs within Markov chain of finite length R. The actual number R is usually determined by experimenting, and depends on both probabilities P and W and the desired accuracy of the simulation. Given P and W, R has to be chosen large enough that the systematic error due to the deviation from the true distribution is smaller than the statistical error due to finite size of the sample (see Eq. 7). In applications with a large number of degrees of freedom, where state vector $\mathbf{\Theta}$ may have millions of strongly coupled components, relaxation process is non-trivial. For our present purpose, state vector is low-dimensional and relaxation occurs within just a few steps along the Markov chain.

The prescription for a practical algorithm can now be summarized as follows:

1. Pick an arbitrary initial path.

2. Generate a new trial path.

3. The new path is accepted with probability W. Specifically, if $W \geq 1$, the new path is accepted without further tests. If $W < 1$, a random number between 0 and 1 is generated, and the new path is accepted if the random number is smaller than W. If the trial path is accepted, it becomes the current path $\mathbf{\Omega}_\nu$, otherwise the old path remains the current path $\mathbf{\Omega}_\nu$.

4. If we progressed enough along the Markov chain so that the relaxation is completed, (i.e. $\nu \geq R$), the current path is sampled from the desired distribution $P(\mathbf{\Omega})$. We compute the payoff function for the current path $F(\mathbf{\Omega}_\nu)$ and accumulate the result (see Eqs. 4, 6), $A = A + F(\mathbf{\Omega}_\nu)$.

5. Perform an estimate of the statistical errors due to Monte Carlo sampling procedure. If the error is above desired level of accuracy, go to (2), otherwise go to (6).

6. Compute Monte Carlo estimates of the required integrals. If L denotes the last value of the step index ν, and R is the number of relaxation steps, the total number of Monte Carlo measurements is $M_\nu = L - R$. Monte Carlo estimate of the option price $< Q >_{MC}$, given the payoff function F, is obtained as:

$$< Q >_{MC} = \frac{A}{M_\nu} = \frac{1}{M_\nu} \sum_{\nu=R+1}^{L} F(\mathbf{\Omega}_\nu) \tag{12}$$

   Error estimate requires that we also accumulate:

$$< Q^2 >_{MC} = \frac{1}{M_\nu} \sum_{\nu=R+1}^{L} F^2(\mathbf{\Omega}_\nu) \tag{13}$$

   Estimate of the sampling error is obtained as a square root of the variance of the Monte Carlo run:

$$\epsilon = (< \sigma^2 >_{MC})^{1/2}$$
$$< \sigma^2 >_{MC} = \frac{1}{M_\nu}(< Q^2 >_{MC} - < Q >^2_{MC}) \tag{14}$$

7. Stop

III. WHAT CAN BE COMPUTED IN A SINGLE SIMULATION

The most important advantage of the path-integral approach is that more information can be obtained in a single simulation than using the standard approach. The basic observation is that *all* relevant quantites can be expressed as integrals with respect to the path probability distribution. Therefore, all of them can be simultaneously computed within a single simulation with that distribution.

This is very important for computation of partial derivatives of the contingent claim's price with respect to various parameters. The standard practice is to compute derivatives using numerical differentiation. This approach introduces discretization errors in addition to statistical sampling errors. Numerical differentiation is also computationally expensive, since it requires repeating Monte Carlo simulations for nearby values of parameters. To illustrate the path-integral approach, we start from Eq. 4, and we denote explicitly dependance of the price $Q(X)$, the payoff function $F(\mathbf{\Omega}, X)$ and path probability $P(\mathbf{\Omega}, X)$ on a parameter X:

$$Q(X) = \int D\mathbf{\Omega} F(\mathbf{\Omega}, X) P(\mathbf{\Omega}, X) \tag{15}$$

We have absorbed the present-value discount factor, $\exp\left\{ - \int_0^T r(t)dt \right\}$, into the definition of payoff function $F(\mathbf{\Omega}, X)$.

The desired partial derivative is given by:

$$\frac{\partial Q(X)}{\partial X} = \int D\mathbf{\Omega} \left[\frac{\partial F(\mathbf{\Omega}, X)}{\partial X} + F(\mathbf{\Omega}, X) \frac{\partial \ln P(\mathbf{\Omega}, X)}{\partial X} \right] P(\mathbf{\Omega}, X) \tag{16}$$

Therefore, Monte Carlo estimate of a partial derivative $\frac{\partial Q(X)}{\partial X}$ of the price may be computed in the same Monte Carlo run as the price Q itself, by accumulating (see Eq. 12):

$$\left\langle \frac{\partial Q(X)}{\partial X} \right\rangle_{MC} = \frac{1}{M_\nu} \sum_\nu \left[\frac{\partial F(\mathbf{\Omega}_\nu, X)}{\partial X} + F(\mathbf{\Omega}_\nu, X) \frac{\partial \ln P(\mathbf{\Omega}_\nu, X)}{\partial X} \right] \tag{17}$$

Derivative of the path probability $\ln P(\mathbf{\Omega}_\nu, X)$ is a sum of contributions coming from individual time slices (see Eq. 5):

$$\frac{\partial \ln P(\mathbf{\Omega}_\nu, X)}{\partial X} = \sum_{i=0}^{i=T-1} \frac{\partial \ln P(\mathbf{\Theta}(i+1)|\mathbf{\Theta}(i), X)}{\partial X} \tag{18}$$

If the parameter X is the initial stock price or the strike price, this expression simplifies considerably, since only first or last time slice appears in the sum. Equation 18 implies that contributions from different time slices are independent: this is an important source of parallelism which can be exploited on a concurrent processor (see Section V).

Following Ferrenberg and Swendsen (1988), knowledge of the path probability distribution may be used to obtain results for a different probability distribution. In practice, this means that within a single Monte Carlo simulation with a particular fixed set of parameters (e.g. initial stock price, volatility, exercise price) we can compute results corresponding to other sets of parameters which are close to the one used in the simulation. Let us denote the parameters used in the simulation by a vector $\mathbf{X}$. Let us also denote a different parameter vector by $\mathbf{Y}$. Then:

$$Q(\mathbf{Y}) = \int D\mathbf{\Omega} F(\mathbf{\Omega}, \mathbf{Y}) P(\mathbf{\Omega}, \mathbf{Y}) \tag{19}$$

which can be rewritten as:

$$Q(\mathbf{Y}) = \int D\mathbf{\Omega} \frac{F(\mathbf{\Omega}, \mathbf{Y}) P(\mathbf{\Omega}, \mathbf{Y})}{P(\mathbf{\Omega}, \mathbf{X})} P(\mathbf{\Omega}, \mathbf{X}) \tag{20}$$

Thus, the change of parameters amounts to a redefiniton of the payoff function. This equation implies that by accumulating:

$$Q(\mathbf{Y})_{MC} = \frac{1}{M_\nu} \sum_\nu \frac{F(\mathbf{\Omega}_\nu, \mathbf{Y}) P(\mathbf{\Omega}_\nu, \mathbf{Y})}{P(\mathbf{\Omega}_\nu, \mathbf{X})} \tag{21}$$

for a number of different values of parameter vector $\mathbf{Y}$, while running a simulation for a fixed vector $\mathbf{X}$, one can obtain results for a window of parameters without repeating the simulation for each new set of parameters. Practical limitations arise from the need to have efficient importance sampling. If the parameters change too much, so that the true path probability distribution for that choice of parameters is significantly different from the one that is used in the simulation, benefits of importance sampling are lost. This limits the size of the parameter window which can be scanned with uniform accuracy in a single simulation. We are also looking into the possibility of using equation 21 as the basis of a control variate technique.

IV. SEQUENTIAL IMPLEMENTATION

At this point we will consider the simplest possible valuation problem. We will describe in detail a path-integral Monte Carlo evaluation of the price of a European call on a stock following the standard Ito price process with constant volatility. Interest rate is also assumed to be constant: $r(t) = r_f$. The exact solution is given by the well-known Black-Scholes (1973) formula. Evolution of the stock price logarithm, $y = \log S$ is given by the stochastic differential equation:

$$d\log S = dy = \mu dt + \sigma d\xi \tag{22}$$

Standard notation is used: μ is expected return and σ is the stock price volatility. Stochastic component $d\xi$ is a Wiener process with variance dt. In a risk-neutral world, the expected return on the stock is equal to the risk-free interest rate, which requires that $\mu = r_f - \sigma^2/2$. If the stock price logarithm at time t is y_t, risk-neutral probability distribution of stock price logarithms $y_{t+\Delta t}$ at instant $t + \Delta t$ is Gaussian:

$$P(y_{t+\Delta t}|y_t) = \exp\left\{-\frac{(y_{t+\Delta t} - y_t - \mu\Delta t)^2}{2\sigma^2\Delta t}\right\} \tag{23}$$

For this particular stochastic process, equation 23 is true for any time interval. For general continuous price processes, Gaussian form is valid only in the limit $\Delta t \to 0$. Probability of a path $\mathbf{\Omega} = (y(0), y(\Delta t)...y(M\Delta t), y(T))$, with M time slices, can be written (using simplified notation: $n \equiv n\Delta t$):

$$P(\mathbf{\Omega}) = \prod_{n=0}^{n=M} \exp\left\{-\frac{(y(n+1) - y(n) - \mu\Delta t)^2}{2\sigma^2\Delta t}\right\} \tag{24}$$

The payoff function for the call is given by:

$$F(S(T), X) = (S(T) - X)\Theta(S(T) - X) = (e^{\log y(T)} - X)\Theta(e^{\log y(T)} - X) \tag{25}$$

where X denotes the strike price and $\Theta(x)$ is the step function.

Given a path $\mathbf{\Omega} = (y(0), y(1), ..., y(n), ..., y(M), y(T))$, we can obtain any other path by a sequence of entirely local steps where we update the value of a stock price at a single time slice n. The new path differs from the old one only by the stock price value at time $n\Delta t$: $\mathbf{\Omega}' = (y(0), y(1), ..., y'(n), ..., y(M), y(T))$. The new value of the stock price logarithm is obtained in the following way: (1) At the beginning of the simulation we pick an interval of width $\Delta = \lambda\sigma^2\Delta t$. (2) For each update, we pick a random number p, such that $-1 \leq p \leq 1$. The new stock price is $y' = y + p\Delta$. The scale factor λ is chosen by experimentation, so that the acceptance rate of the updated configurations is roughly 0.5.

Following Metropolis algorithm, we have to accept or reject the new path according to the transition probability $W(\mathbf{\Omega} \to \mathbf{\Omega}')$. Combining equations 9 and 24, we obtain the following expression for the transition probability:

$$W(\mathbf{\Omega} \to \mathbf{\Omega}') = \frac{\Lambda'(n-1, n, n+1)}{\Lambda(n-1, n, n+1)} \tag{26}$$

where

$$\Lambda'(n-1, n, n+1) = \exp\left\{-\frac{(y'(n) - y(n-1) - \mu\Delta t)^2}{2\sigma^2\Delta t} - \frac{y(n+1) - y'(n) - \mu\Delta t)^2}{2\sigma^2\Delta t}\right\} \tag{27}$$

and

$$\Lambda(n-1, n, n+1) = \exp\left\{-\frac{(y(n) - y(n-1) - \mu\Delta t)^2}{2\sigma^2\Delta t} - \frac{(y(n+1) - y(n) - \mu\Delta t)^2}{2\sigma^2\Delta t}\right\} \tag{28}$$

This local algorithm is applicable to any Markov process. A non-Markov stochastic process will induce couplings between non-adjacent time slices and one would have to design an appropriate algorithm with global updates. Global updates are dependent on the nature of the stochastic process and consequently the symmetries of the resultant path probability function. It is desirable to include global updates even for Markov processes in order to reduce the variance of Monte Carlo simulation. We used a very simple global update for the present problem, based on the translational invariance of the path probability, (i.e. its dependance on differences of stock price logarithms). When y_n is updated, then y_k, $n < k \leq T$, are also shifted by the same amount. Transition probability for this move is still given by equations 27 and 28 but without the second term in the exponent, since the probability of the path segment after time slice n is not changed by a rigid shift. This move significantly improves statistical quality of results for time slices closer to the contract expiration date.

We usually start from a "deterministic" path, i.e. path given by equation 22 with $\sigma = 0$. We update paths looping over time slices sequentially using both global and local moves. When the whole path is updated it is counted as one Monte Carlo step. After the relaxation period (which is less than 100 steps), we begin accumulating option price estimate and its partial derivatives, as described in Sections I and III.

Computation of option's delta, which measures its price sensitivity, $\delta = \partial Q/\partial S(0)$, requires accumulation of $A = \partial F/\partial S(0) + F\partial \ln P/\partial S(0)$ during the Monte Carlo run. Since $S(0)$ appears only on the initial time slice and the payoff function does not depend explicitly on $S(0)$, the accumulator reduces to

$$A = F(S(T), X)\frac{(y(1) - y(0) - \mu\Delta t)}{S(0)\sigma^2\Delta t} \tag{29}$$

Volatility sensitivity κ is somewhat more complicated, since all of the time slice probabilities depend on σ. Payoff function does not depend on σ, so $A = F\partial \ln P/\partial\sigma$. Using the explicit expression for path probability, Eq. 24, we obtain:

$$A = F(S(T), X)\sum_{n=1}^{n=T}\frac{y(n) - y(n-1) - \mu\Delta t}{\sigma}\left[\frac{(y(n) - y(n-1) - \mu\Delta t)}{\sigma^2\Delta t} - 1\right] \tag{30}$$

In this sum, there are only two terms which depend on the stock price on any given time slice. Therefore, computation of κ has a constant complexity per Monte Carlo update. Furthermore, only adjacent time slices are coupled in the sum, which is important for an efficient parallel implementation.

Similar expressions are easily derived for interest rate sensitivity, ρ. We should note that these equations are not the most efficient way to compute derivatives for this particular problem. By a simple change of variables, one can rewrite the probability function to be independent of σ and $S(0)$, and the complete dependence on these parameters is lumped into the payoff function. However, unlike equations 29 and 30, the resulting expressions cannot be easily generalized to more complicated processes.

We now outline the procedure for computing option prices for multiple initial stock prices in a single simulation with initial stock price $S(0) = S_0$. Additional initial stock prices will be designated by S_j. Since only the first time slice probability depends on $S(0)$, it follows from 21 and 24 that for each additional option price j, we must accumulate:

$$A_j = F(S(T), X)\cdot\exp\left\{-\frac{(y(1) - S_0 - \mu\Delta t)^2 - (y(1) - S_j - \mu\Delta t)^2}{2\sigma^2\Delta t}\right\} \tag{31}$$

Note that F is independent of the initial price, so the same function $F(S(T), X)$ appears for all indices j.

V. PARALLEL IMPLEMENTATION

Sequential updating procedure does not exploit local character of the transition probability, which is emphasized by the notation of Eq. 26. When two paths differ on a single time slice n, ratio of their probabilities (which is the only quantity required in Metropolis algorithm) involves only variables at time slice n and its nearest neighbors, $n-1$ and $n+1$. Therefore, stock prices which are two or more time slices apart can be updated simultaneously, since the appropriate transition probabilities are decoupled. This property of the transition probabilities is generally referred to as data parallelism.

This is not a place to provide a detailed introduction to the concepts of parallel computation. The interested reader is referred to an excellent exposition in Fox et al. (1988). Furthermore, a lot of information for all levels of expertise can be obtained on-line (via Mosaic http://www.npac.syr.edu). We will briefly review a few basic concepts which will suffice for the discussion that follows. A computational problem is data parallel if a single operation can be applied to a number of data items simultaneously, provided there is hardware support. One way the parallel computers provide this hardware support is through a number of independent central processing units (nodes) which operate on subsets of data items stored in each node's local memory. Extrapolation of current trends in high-performance computing indicates that these Multiple-Instruction Multiple-Data (MIMD) distributed-memory architectures will probably provide the most cost-effective hardware solutions. This includes dedicated systems as well as networked workstations, which are conceptually similar. The only important difference is that tightly coupled systems have much higher communication bandwidth, which means that they can better exploit finer grain parallelism. We will discuss implementation of Monte Carlo option pricing on MIMD platforms.

In data parallel applications, domain decomposition is an effective parallelization strategy. The arrays are partitioned among the processors (nodes) of a parallel machine and each node performs the same set of computations on array elements it owns. Whenever a node requires knowledge of array elements owned by another node, messages are exchanged between the two nodes. The structure of the parallel code then consists of segments where independent computations are done within each node, followed by segments where information between nodes is exchanged via calls to message-passing library routines.

Local character of transition probabilities is not the only source of parallelism in this algorithm. For example, trivial parallelism follows from the fact that option prices are usually computed for a number of independent parameters. Because these computations are independent, they can be assigned to different nodes, with negligible communication overhead. In case one needs a quick result for a single option, another trivially parallel strategy is to assign to each node a short Monte Carlo run for the same parameter set and then to combine the results from each node at the end of the simulation. This is also statistically more efficient than to run a single long simulation, since the results from different nodes are uncorrelated. For applications with a large number of time slices one may exploit non-trivial parallelism: a path may be divided up into a number of smaller time intervals, which are in turn assigned to different nodes. Since only adjacent time slices are required to update stock price on a given time slice, a node will own the information it needs for most of the time slices, except those on the boundaries of the time interval assigned to the node. Each node will update sequentially time slices it owns (following the same algorithm as described in the previous section), until it encounters one of the boundary slices. At that moment, it will communicate with a neighboring node to obtain the value of the neighbor's boundary time slice. After the communication is completed, it has all the information it needs to update the last time slice. Then the whole sequence is repeated for the next Monte Carlo step. Equations 29, 30 and 31 show that the same divide and conquer strategy can be employed for the accumulation of option price and its parameter sensitivities. This sequence of communications and computations is executed by all nodes simultaneously, which brings a speed-up roughly proportional to the number of nodes.

It follows from Eq. 26 that all even time slices can be updated simultaneously and then all odd time slices can be updated simultaneously. This implies that the highest degree of parallelism can be achieved by having only 2 time slices per node. In practice, the situation is complicated by the fact that exchange of messages between nodes is a much slower operation (sometimes by factor of 10^2 or even 10^3) than a typical floating point operation within a node. This communication overhead varies significantly from one architecture to another and limits the granularity of parallelism on a particular machine. There is an optimal number of time slices per node (usually bigger than 2!) which balances local computation and communication costs. Communication overhead is very important for simulations with a small number of time slices running on a networked cluster of workstations, which has large communication latencies. In this case, it is more efficient to exploit only trivial parallelism and keep complete paths in local node memories, than to incur excessive communication overheads. As discussed earlier, this decision also depends on the character of the pricing problem itself. In addition to the number of time slices, one should also take into account the computational complexity of the operations performed on a single time slice: if it is sufficiently large, it may be advantageous to subdivide paths into shorter time segments.

Both sources of parallelism render option pricing via path integral Monte Carlo simulation a suitable candidate for efficient parallel implementation. A programmer may choose to implement the algorithm by explicitly coding the exchange of boundary time slices between nodes or to use a high level data-paralel language which has semantic support for parallelism and hides the complexities of explicit message passing constructs. We implemented Path-integral Monte Carlo code in High Performance Fortran (HPF), which is among the most prominent of such languages. HPF language specification was adopted in 1993 as a result of collaboration of academic institutions and hardware and software vendors (High Performance Fortran Forum (1993)). Dialects of HPF already existed on Connection Machine and MasPar computers and at least 4 compilers are expected to be commercially available in 1995. For a complete language specification, one can consult High Performance Fortran Forum (1993). There is also extensive amount of information which can be obtained on-line via Mosaic http://www.npac.syr.edu/hpfa.

HPF may be considered a superset of Fortran 90 with a small number of extensions to support effective use of MIMD machines. In both Fortran 90 and HPF arrays are first class objects. Array syntax allows the programer to manipulate complete arrays and thus express parallelism transparently: since the array elements are distributed among the nodes of a parallel processor, an array instruction is executed by all nodes simultaneously. Translation between addreses in the global name space and addresses of array elements in local node memories is done by the system. Also, if the array operation requires that array elements be fetched from remote node memories, appropriate communication calls are generated by the compiler transparently to the programmer. Array syntax alone is not sufficient to guarantee good performance. Non-uniform memory access on MIMD computers implies that data locality is crucial for good performance. Because of that, some of the most important HPF extensions to Fortran 90 consist of compiler directives for data distribution among nodes, which provides the programmer with a set of high-level tools for management of data layout. Using the compiler directives, different parallelization strategies described above were implemented by changing only couple of lines of the code.

Performance gain from hand coding of low level message passing constructs is negligible for this application, since the data layout is simple, static and regular. Implementation in a high-level language like HPF has the added benefits of code modularity, clarity and simplicity, as well as the ease of debugging, maintenance and portability. We tested the HPF code on a number of parallel machines: Connection Machine CM5, MasPar, DEC Alpha workstation farm and a network of Sun workstations. We have also implemented the algorithm using the native message-passing library on an IBM SP-2 parallel computer. Using the simple parallelization strategy of combining multiple independent short simulations, we obtained almost perfect parallel speed-up on all platforms.

VI. RESULTS

Our present goal is to examine feasibility and accuracy of the method, so we will begin with the simplest valuation problem of a European call on a stock with constant volatility and no dividends, where we can easily compare Monte Carlo results with the analytic Black-Scholes solution.

In Table 1, we show results for a couple of realistic parameter choices and examine their accuracy as the number of Monte Carlo steps is varied. Exact results are always within estimated confidence limits of Monte Carlo results. Statistical errors after 100000 Monte Carlo steps are less than half percent for all maturities. The error is less than tenth of a percent for 1.6×10^6 steps. These statistical uncertainties reflect improvements achieved by explicit use of all the symmetries of path probabilities, which enabled us to accumulate more independent results per path. For example, if a stock price path is reflected with respect to the deterministic path, its probability is the same, so we can accumulate results for the reflected path as well, with negligible computational cost. This can be regarded as a rudimentary variance reduction technique. Since in this paper we want to examine merits of the basic method, we leave systematic exploration of variance reduction techniques for future work. A promising technique we are experimenting with is the inclusion of additional global Monte Carlo updates. Table 1 also shows that statistical errors scale as $\epsilon \propto N^{-1/2}$ with the number of Monte Carlo steps, which is in agreement with the central limit theorem and also shows that successive Monte Carlo steps are not correlated. Correlation between Monte Carlo steps is reduced because at every step we pick with equal probabilities either the current path or any of its reflection symmetry related paths.

In Table 2, we present results for parameter sensitivities using the same parameter choices as in Table 1. They are obtained concurrently with the option price itself, as described in Section III. They show even higher level of accuracy than the corresponding option price for a given number of Monte Carlo steps. If these values were obtained by numerical differentiation, it would require at least 3 simulations besides the original one to compute the three partial derivatives. Additional simulations may also be required, depending on the statistical accuracy of Monte Carlo results. If the statistical errors are large, one would need simulations for a few nearby parameter values, combined with a least-squares fit to produce estimates of derivatives. This may lead to unacceptably large errors for higher order derivatives (like γ for example), unless statistical errors for option prices are very small. In the path integral approach there are no additional sources of errors.

In Section III we pointed out the possibility of computing Monte Carlo results for different parameters in a single simulation. This is illustrated in Table 3, where option values in a window of about 10% variation of initial stock price are computed in a single run. Within a few percent difference from the stock price used in the simulation, results are roughly of the same statistical quality as for the original price. This is a very cheap and efficient way to explore option price variations in a limited parameter range, particularly if there is uncertainty about input parameter estimates. It is clear from the table that the further one goes from the original simulation parameters the worse the statistics becomes (bigger relative errors) due to inefficient importance sampling. The same trend is apparent for longer periods to maturity, because the differences between the simulation probability distribution and the true ones are amplified for longer time periods. For shorter periods to maturity, there is an apparent asymmetry between errors, which are much smaller for the initial prices below the simulation price $S_i < S_0$ then for prices above the simulation price, $S_i > S_0$. The reason is that the stock price distribution is skewed towards higher stock prices, so that the overlap between simulation price distribution and actual price distributions is bigger for $S_i < S_0$ than for $S_i > S_0$. This effect becomes less and less important for longer time periods to maturity.

We first applied path-integral Monte Carlo approach to the simple Black-Scholes model, where exact results are easy to obtain, to show the accuracy and efficiency of the method. The results indicate that this approach shows significant potential which must be tested on realistic problems if it is to become a useful simulation tool. The application of the method to various non-trivial models of underlying asset dynamics is the thrust of our future work. As a first step in this direction we show results for a jump diffusion model (see Merton (1976)), where jumps are superimposed upon the continuous Wiener process. We will consider the following differential/difference equation corresponding to this process:

$$d\log S = dy = \mu dt + \sigma d\xi + dZ \tag{32}$$

where dZ is the stochastic variable desribing the jump process. It is assumed that number of jumps is Poisson distributed while jump size is uniformly distributed with average $\langle dZ \rangle = 0$. Finite average value of the jump size will amount to a trivial shift in the drift coefficient μ. Merton (1976) was able to obtain a series solution for the option price only under the assumption that jump size distribution is normal. This restriction can be lifted in a Monte Carlo simulation, so we chose a uniform distribution for experimentation purposes, since it is computationally cheap and there is no analytic solution. The results for a European call on an asset following this process are shown in Table 4. Prices and sensitivities are obtained concurrently and the accuracy is comparable to the one achieved on the Black-Scholes problem. Relative errors for 1×10^5 steps are below one percent for option price and below two tenths of a percent for some δ and ρ sensitivities. As for the Black-Scholes model, price sensitivities are more accurately determined than the option price itself. Relative quality of estimators depends on the form of the corresponding function (see Eq. 16) which is integrated with respect to the path probability measure. If one computed sensitivities using numerical differentiation, errors would be at least as large as the price error.

VII. CONCLUSIONS

We introduced a new Monte Carlo technique for valuation of derivative securities. The method is based on the probability distribution of complete histories of the underlying security process. We used Metropolis algorithm to generate this probability distribution. We showed that this approach is efficient, accurate and allows one to obtain a *complete* solution of the valuation problem in a *single* simulation. One can obtain only price in a single simulation using the standard Monte Carlo method. Using path-integral Monte Carlo simulation one can obtain price sensitivity with respect to all input parameters and even compute prices for multiple parameter values. Path integral method can easily incorporate global constraints on the underlying security dynamics, which may prove very useful for applications such as bond option pricing. We also showed how this approach can leverage the advancements in parallel computing technology. In future work we will incorporate variance reduction techniques into the basic algorithm, we will generalize the method to include American contracts and will also implement more realistic models of underlying asset dynamics.

TABLE 1. Comparison of Monte Carlo estimates and exact results for European call values. This table shows the level of accuracy which can be achieved as the number of Monte Carlo steps ranges from 1×10^5 to 16×10^5. Risk-free interest rate per period is set to $r_f = 0.004853$. N_t is the number of periods to maturity, σ^2 is stock price variance per period, $C(N)$ is European call value Monte Carlo estimate after N Monte Carlo steps, and $\epsilon(N)$ is the error estimate after N Monte Carlo steps. Exact results obtained using Black-Scholes formula are listed in the last column (C). Initial stock price is $S = 100$ and the strike price is $X = 100$ for all data sets in the table.

N_t	σ^2	$C(1 \times 10^5)$	$\epsilon(1 \times 10^5)$	$C(4 \times 10^5)$	$\epsilon(4 \times 10^5)$	$C(16 \times 10^5)$	$\epsilon(16 \times 10^5)$	C
1	0.001875	1.9792	0.0064	1.9769	0.0032	1.9750	0.0016	1.9761
2	0.001875	2.9508	0.0106	2.9456	0.0053	2.9430	0.0026	2.9443
3	0.001875	3.7522	0.0139	3.7492	0.0069	3.7469	0.0034	3.7482
4	0.001875	4.4710	0.0168	4.4678	0.0083	4.4673	0.0041	4.4675
5	0.001875	5.1446	0.0193	5.1324	0.0096	5.1326	0.0048	5.1327
6	0.001875	5.7822	0.0216	5.7610	0.0108	5.7608	0.0054	5.7597
7	0.001875	6.3823	0.0238	6.3599	0.0118	6.3595	0.0059	6.3576
8	0.001875	6.9456	0.0258	6.9309	0.0129	6.9337	0.0064	6.9324
9	0.001875	7.4969	0.0276	7.4881	0.0138	7.4875	0.0069	7.4883
10	0.001875	8.0335	0.0294	8.0259	0.0147	8.0235	0.0073	8.0285
11	0.001875	8.5593	0.0312	8.5511	0.0156	8.5493	0.0078	8.5549
12	0.001875	9.0780	0.0338	9.0663	0.0169	9.0660	0.0084	9.0696
1	0.002500	2.2467	0.0075	2.2436	0.0037	2.2415	0.0018	2.2411
2	0.002500	3.3269	0.0123	3.3198	0.0061	3.3168	0.0030	3.3161
3	0.002500	4.2089	0.0161	4.2024	0.0080	4.2004	0.0040	4.1999
4	0.002500	4.9935	0.0195	4.9873	0.0097	4.9864	0.0048	4.9848
5	0.002500	5.7262	0.0224	5.7086	0.0112	5.7085	0.0056	5.7065
6	0.002500	6.4168	0.0252	6.3873	0.0125	6.3855	0.0062	6.3831
7	0.002500	7.0602	0.0277	7.0302	0.0138	7.0290	0.0069	7.0255
8	0.002500	7.6629	0.0301	7.6415	0.0150	7.6434	0.0075	7.6406
9	0.002500	8.2483	0.0322	8.2351	0.0161	8.2348	0.0080	8.2335
10	0.002500	8.8169	0.0343	8.8066	0.0172	8.8041	0.0086	8.8075
11	0.002500	9.3733	0.0364	9.3634	0.0182	9.3594	0.0091	9.3654
12	0.002500	9.9226	0.0395	9.9076	0.0197	9.9028	0.0099	9.9092

TABLE 2. Option price sensitivities to input parameters. This table lists some of the price sensitivities which can be computed along the option price in a path-integral simulation. Initial stock value is set to $S = 100$ and the strike price is $X = 100$. Number of Monte Carlo steps is 1×10^5. Each parameter sensitivity estimate (suffix MC) is immediately followed by its error estimate ϵ and the exact value obtained by differentiation of the Black-Scholes formula. δ is the stock price sensitivity ($\delta = \partial C/\partial S$), κ is the volatility sensitivity ($\kappa = \partial C/\partial \sigma$), and ρ is the interest rate sensitivity ($\rho = \partial C/\partial r_f$).

N_t	r_f	σ^2	δ_{MC}	ϵ	δ	κ_{MC}	ϵ	κ	ρ_{MC}	ϵ	ρ
1	0.004853	0.001875	0.5530	0.00039	0.5532	0.1143	0.0005	0.1141	0.04443	0.00005	0.04445
2	0.004853	0.001875	0.5745	0.00045	0.5750	0.1610	0.0006	0.1600	0.09083	0.00011	0.09092
3	0.004853	0.001875	0.5914	0.00049	0.5916	0.1949	0.0008	0.1942	0.13848	0.00019	0.13852
4	0.004853	0.001875	0.6060	0.00051	0.6054	0.2219	0.0010	0.2222	0.18710	0.00027	0.18692
5	0.004853	0.001875	0.6167	0.00053	0.6175	0.2469	0.0012	0.2463	0.23555	0.00035	0.23592
6	0.004853	0.001875	0.6276	0.00054	0.6283	0.2687	0.0013	0.2673	0.28485	0.00043	0.28539
7	0.004853	0.001875	0.6386	0.00055	0.6382	0.2867	0.0014	0.2862	0.33526	0.00052	0.33523
8	0.004853	0.001875	0.6474	0.00056	0.6473	0.3034	0.0016	0.3032	0.38522	0.00061	0.38536
9	0.004853	0.001875	0.6564	0.00057	0.6558	0.3179	0.0017	0.3188	0.43615	0.00071	0.43573
10	0.004853	0.001875	0.6635	0.00057	0.6638	0.3323	0.0018	0.3330	0.48614	0.00080	0.48627
11	0.004853	0.001875	0.6707	0.00057	0.6713	0.3451	0.0019	0.3461	0.53639	0.00089	0.53694
12	0.004853	0.001875	0.6772	0.00058	0.6784	0.3577	0.0021	0.3583	0.58630	0.00099	0.58771
1	0.004853	0.002500	0.5485	0.00035	0.5486	0.1146	0.0004	0.1143	0.043845	0.000045	0.043847
2	0.004853	0.002500	0.5682	0.00040	0.5685	0.1617	0.0006	0.1605	0.089164	0.000106	0.089227
3	0.004853	0.002500	0.5838	0.00044	0.5837	0.1960	0.0009	0.1950	0.135424	0.000174	0.135430
4	0.004853	0.002500	0.5966	0.00046	0.5964	0.2236	0.0010	0.2235	0.182223	0.000248	0.182195
5	0.004853	0.002500	0.6074	0.00048	0.6075	0.2490	0.0012	0.2481	0.229169	0.000324	0.229369
6	0.004853	0.002500	0.6167	0.00049	0.6175	0.2715	0.0013	0.2697	0.276238	0.000402	0.276847
7	0.004853	0.002500	0.6270	0.00050	0.6266	0.2901	0.0015	0.2892	0.324568	0.000488	0.324553
8	0.004853	0.002500	0.6354	0.00051	0.6350	0.3072	0.0016	0.3068	0.372574	0.000572	0.372425
9	0.004853	0.002500	0.6433	0.00052	0.6428	0.3226	0.0017	0.3230	0.420538	0.000658	0.420414
10	0.004853	0.002500	0.6506	0.00052	0.6502	0.3373	0.0019	0.3380	0.468612	0.000745	0.468478
11	0.004853	0.002500	0.6569	0.00053	0.6572	0.3508	0.0020	0.3519	0.516353	0.000833	0.516584
12	0.004853	0.002500	0.6629	0.00053	0.6637	0.3641	0.0021	0.3648	0.563794	0.000929	0.564700

TABLE 3. Computation of option prices for multiple parameters in a single simulation. This table shows the level of accuracy which can be obtained if multiple option prices are computed in a single simulation. Number of Monte Carlo steps is 1×10^5, initial stock price is $S_0 = 100$, strike price is $X = 100$, volatility per period is $\sigma^2 = 0.0025$, and riskless interest rate is $r_f = 0.004853$ per period. Each option price estimate $C(S_i)$ for initial stock price S_i is followed by its error estimate ϵ and the exact value from Black-Scholes formula C. N_t denotes the number of time periods to maturity.

N_t	$C(95)$	ϵ	C	$C(99)$	ϵ	C	$C(101)$	ϵ	C	$C(105)$	ϵ	C
1	0.4632	0.0006	0.4629	1.7364	0.0047	1.7325	2.8358	0.0117	2.8287	5.8448	0.0572	5.8540
2	1.1827	0.0051	1.1790	2.7907	0.0080	2.7757	3.9362	0.0154	3.9121	6.8103	0.0639	6.7915
3	1.8600	0.0098	1.8613	3.6494	0.0107	3.6390	4.8268	0.0183	4.8059	7.6560	0.0714	7.6377
4	2.5060	0.0145	2.5049	4.4087	0.0130	4.4080	5.6044	0.0207	5.6004	8.4129	0.0774	8.4160
5	3.1353	0.0192	3.1165	5.1283	0.0151	5.1164	6.3470	0.0230	6.3310	9.1655	0.0826	9.1444
6	3.7185	0.0232	3.7024	5.8019	0.0170	5.7813	7.0462	0.0251	7.0160	9.8878	0.0877	9.8342
7	4.2810	0.0269	4.2671	6.4267	0.0188	6.4133	7.6846	0.0271	7.6662	10.5269	0.0926	10.4933
8	4.8267	0.0312	4.8140	7.0248	0.0205	7.0191	8.2981	0.0288	8.2888	11.1462	0.0968	11.1270
9	5.3438	0.0345	5.3457	7.5966	0.0220	7.6032	8.8850	0.0305	8.8887	11.7508	0.1010	11.7393
10	5.8616	0.0380	5.8641	8.1616	0.0235	8.1691	9.4637	0.0321	9.4693	12.3372	0.1052	12.3334
11	6.3681	0.0415	6.3710	8.7055	0.0249	8.7194	0.0198	0.0336	10.0335	12.9117	0.1086	12.9115
12	6.8597	0.0450	6.8676	9.2435	0.0264	9.2561	0.5745	0.0352	10.5834	13.5007	0.1139	13.4755

TABLE 4. Call option price and sensitivities for a jump diffusion process. This table lists results for the option price and its input parameter sensitivities when a jump process is superimposed on the continuous process of the Black-Scholes model. Initial stock value is set to $S = 100$ and the strike price is $X = 100$. Number of Monte Carlo steps is 1×10^5. Each Monte Carlo result is immediately followed by its error estimate ϵ. Jump rate per period is set to $k_P = 0.1$. Riskless interest rate per period is $r_f = 0.004853$ and variance per period is $\sigma^2 = 0.001875$. Jump sizes are uniformly distributed in the interval $(-\Delta, +\Delta)$. δ is the stock price sensitivity ($\delta = \partial C/\partial S$), κ is the volatility sensitivity ($\kappa = \partial C/\partial \sigma$), and ρ is the interest rate sensitivity ($\rho = \partial C/\partial r_f$).

N_t	Δ	C	ϵ	δ	ϵ	κ	ϵ	ρ	ϵ
1	0.02	2.12671	0.01505	0.55233	0.00121	0.12487	0.00108	0.04426	0.00010
2	0.02	3.16259	0.02458	0.57517	0.00141	0.17626	0.00159	0.09059	0.00024
3	0.02	3.97857	0.03243	0.59134	0.00151	0.21151	0.00202	0.13789	0.00040
4	0.02	4.77069	0.03896	0.60583	0.00159	0.24485	0.00244	0.18604	0.00057
5	0.02	5.47055	0.04516	0.61457	0.00161	0.27364	0.00281	0.23328	0.00072
6	0.02	6.19997	0.05057	0.62528	0.00165	0.30257	0.00316	0.28164	0.00090
7	0.02	6.85696	0.05560	0.63553	0.00168	0.32708	0.00349	0.33073	0.00108
8	0.02	7.50037	0.06024	0.64449	0.00170	0.34980	0.00382	0.37966	0.00126
9	0.02	8.10141	0.06485	0.65231	0.00172	0.36877	0.00412	0.42847	0.00145
10	0.02	8.68738	0.06891	0.66075	0.00173	0.39014	0.00443	0.47823	0.00164
11	0.02	9.23218	0.07319	0.66640	0.00174	0.40439	0.00472	0.52623	0.00182
12	0.02	9.78240	0.07923	0.67459	0.00175	0.42227	0.00503	0.57677	0.00204
1	0.05	3.05180	0.02526	0.55565	0.00110	0.19220	0.00192	0.04376	0.00009
2	0.05	4.60908	0.04117	0.57536	0.00120	0.28671	0.00285	0.08821	0.00021
3	0.05	5.81127	0.05421	0.59300	0.00131	0.35139	0.00363	0.13372	0.00035
4	0.05	6.94802	0.06491	0.60581	0.00136	0.41691	0.00432	0.17878	0.00049
5	0.05	7.97130	0.07494	0.61854	0.00141	0.47293	0.00502	0.22451	0.00064
6	0.05	9.01019	0.08419	0.62963	0.00144	0.53323	0.00569	0.26977	0.00080
7	0.05	9.97512	0.09320	0.63949	0.00146	0.58649	0.00627	0.31485	0.00096
8	0.05	10.87057	0.10119	0.64944	0.00149	0.63476	0.00693	0.36049	0.00113
9	0.05	11.75562	0.10962	0.65942	0.00153	0.67822	0.00754	0.40639	0.00132
10	0.05	12.61027	0.11694	0.66944	0.00156	0.73089	0.00826	0.45278	0.00150
11	0.05	13.38559	0.12495	0.67883	0.00159	0.76241	0.00883	0.49956	0.00171
12	0.05	14.21032	0.13614	0.68505	0.00160	0.81260	0.00942	0.54295	0.00190

References

F. Black and M. J. Scholes, 1973, The Pricing of options and corporate liabilities, *Journal of Political Economy* **81**, 637-659.

P. Boyle, 1977, Options: A Monte Carlo approach, *Journal of Financial Economics* **4**, 323-338.

J. C. Cox and S. A. Ross, 1976, The valuation of options for alternative stochastic processes, *Journal of Financial Economics* **3**, 145-166.

A. M. Ferrenberg and R. H. Swendsen, New Monte Carlo technique for studying phase transitions, *Physical Review Letters* **61**, 2635-2638.

G. C. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon and D. Walker, 1988, *Solving Problems on Concurrent Processors*, (Prentice Hill, Englewood Cliffs, New Jersey).

J. M. Hammersley and D. C. Handscomb, 1964, *Monte Carlo Methods* (Methuen, London).

High Performance Fortran Forum (HPFF), 1993, High Performance Fortran Language Specification, *Scientific Programming* **2**, No. 1. Also available by anonymous ftp from ftp.npac.syr.edu (cd /HPFF).

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, E. H. Teller and E. Teller, 1953, Equation of state calculations by fast computing machines, *Journal of Chemical Physics* **21**, 1087-1091.

J. W. Negele and H. Orland, 1988, *Quantum Many-Particle Systems* (Addison Wesley, New York).

A. Papoulis, 1984, *Probability, Random Variables and Stochastic Processes* (McGraw-Hill).