

Parallel Computation of Multivariate Normal Probabilities

Elise deDoncker
Dept. of Computer Science
Western Michigan Univ.
Kalamazoo, MI 49008

Alan Genz
Dept. of Mathematics
Washington State Univ.
Pullman, WA 99164

Manuel Ciobanu
Dept. of Computer Science
Western Michigan Univ.
Kalamazoo, MI 49008

Abstract

We present methods for the computation of multivariate normal probabilities on parallel/ distributed systems. After a transformation of the initial integral, an approximation can be obtained using Monte-Carlo or quasi-random methods. We propose a meta-algorithm for asynchronous sampling methods and derive efficient parallel algorithms for the computation of MVN distribution functions, including a method based on randomized Koroobov and Richtmyer sequences. Timing results of the implementations using the MPI parallel environment are given.

1 Introduction

The computation of the multivariate normal distribution function

$$F(\mathbf{a}, \mathbf{b}) = |\Sigma|^{-\frac{1}{2}} (2\pi)^{-\frac{n}{2}} \int_{\mathbf{a}}^{\mathbf{b}} e^{-\frac{1}{2} \mathbf{x}^T \Sigma^{-1} \mathbf{x}} d\mathbf{x}. \quad (1)$$

often leads to computational-intensive integration problems. Here Σ is an $n \times n$ symmetric positive definite covariance matrix; furthermore one of the limits in each integration variable may be infinite. Genz [5] performs a sequence of transformations resulting in

$$I = (e_1 - d_1) \int_0^1 (e_2 - d_2) \dots \int_0^1 (e_n - d_n) \int_0^1 d\mathbf{w}$$

with $d_i = \Phi((a_i - \sum_{j=1}^{i-1} c_{ij} \Phi^{-1}(z_j))/c_{ii})$, $e_i = \Phi((b_i - \sum_{j=1}^{i-1} c_{ij} \Phi^{-1}(z_j))/c_{ii})$, where $\Phi(y)$ is the standard normal distribution function, C is the Cholesky factor of the covariance matrix Σ and $z_j = d_j + w_j(e_j - d_j)$.

This transformation maps the original region to a unit hypercube and reduces the dimension of the problem by one since the transformed integrand does not depend on w_n . It also imposes a priority ordering on the variables $(w_1, w_2, \dots, w_n$, with w_1 the most important) since

d_i and e_i depend on all of w_1, w_2, \dots, w_{i-1} . A comparison of (sequential) techniques (Genz [6]) using the transformation followed by an adaptive method (AM), a crude Monte-Carlo (MC) and a Quasi Monte-Carlo (QMC) method indicate that all provide robust ways to solve the problem, QMC and AM can be used for high accuracy work and QMC becomes faster than AM for moderately high dimensions.

We have done extensive work on the parallelization of adaptive methods in the past (see, e.g., [3, 4]), which has led to the parallel software package ParInt). For moderately low dimensions, the adaptive method used in ParInt is (as AM) particularly suited for the above transformed integral, since it bisects each selected region in a priority coordinate direction. However, the dimensional effect exhibited by generating an adequate partitioning level for a high-dimensional domain can only be reduced linearly (with respect to the number of processes) in a parallel execution. In this paper, we will therefore focus on the parallelization of sampling methods. A parallel meta-algorithm which can be implemented asynchronously will be presented in Section 2. Convergence properties of QMC methods are discussed in Section 3. Sections 4 and 5 give results and conclusions, respectively.

2 Asynchronous Sampling

Sampling methods are sometimes called “embarrassingly parallelizable”. This refers to assigning about equal numbers of evaluation points to all processes and collecting their results. The latter can be done via a single-node-accumulation which, however, introduces synchronization. Good sequential algorithms for integration by sampling evaluate a sequence of sample sets of increasing size N_k , while attempting to attain a prescribed accuracy; in some cases they allow starting from the results obtained so far rather than restarting. Thus, synchronization points would be required at every step in the sequence. On a heterogeneous network this synchroniza-

```

Asynchronous_sampling() {
  if(myid == 0) { /* I am the controller */
    controller_init();
    repeat {
      if(sample results from worker arrived)
        receive sample results (Q,S,N);
      else {
        set next N;
        compute sample results (Q,S,N);
      }
      apply Q,S,N to update F,E,Ntotal;
    } until (E < eps || Ntotal > Nmax);
    broadcast done msg. to workers;
  }
  else { /* I am a worker */
    worker_init();
    repeat {
      check for done msg. from controller;
      set next N;
      compute sample results (Q,S,N);
      send (Q,S,N) to controller;
    } until done;
  }
}

```

Figure 1: Asynchronous sampling meta-algorithm

tion requires faster processors to wait for the slower ones. Furthermore, also on a homogeneous network, there is often a significant lag between the creation of the first and the last of the parallel processes, so that processes which started earlier have to wait for the later ones. This suggests an approach using asynchronous processes.

Let $Q_k = \frac{1}{N_k} \sum_{i=1}^{N_k} f(\mathbf{x}_i)$ denote the mean of a sample set of size N_k with sample variance $\sigma_k^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} f^2(\mathbf{x}_i) - Q_k^2$, and standard error $S_k^2 = \sigma_k^2/N_k$. Then a weighted average [5] over the sample sets can be obtained by

$$F = \bar{Q} = \frac{\sum_j \frac{Q_j}{S_j^2}}{\sum_j \frac{1}{S_j^2}}, \quad (2)$$

with error estimate

$$E = \frac{\alpha}{\left(\sum_j \frac{1}{E_j^2}\right)^{\frac{1}{2}}}. \quad (3)$$

In our asynchronous parallelization, one of the processes assumes the role of a “controller” even though it also participates in sample evaluation. The controller’s additional function is to average over the sample sets until the error tolerance `eps` is achieved or the total number of evaluations `Ntotal` reaches a prescribed maximum

`Nmax`. Thus the updating as a global operation plays a significant role in the parallel algorithm.

The general structure of the algorithm is given as a meta-algorithm description in Figure 1.

3 QMC methods

3.1 Lattice Rules

Let $I = I(f) = K_N + E_N$ be the n -variate integral of f over the unit hypercube $0 \leq x_j \leq 1$, $j = 1, 2, \dots, n$, with

$$K_N = \frac{1}{N} \sum_{i=1}^N f\left(\left\{\frac{i}{N}\mathbf{v}\right\}\right), \quad (4)$$

where $\{x\}$ = the fractional part of x and \mathbf{v} is a pre-determined vector with integer coefficients. The vector \mathbf{v} determines a good lattice if the error E_N is small in the following sense [2]. Define \mathcal{E}^k , $k > 1$ as the class of functions f which are periodic with period 1 in each variable and for which the coefficients $c_{\mathbf{m}}$ in the Fourier series expansion of f satisfy

$$|c_{\mathbf{m}}| \leq Cr(\mathbf{m})^{-k} \quad (5)$$

for all $\mathbf{m} \neq \mathbf{0}$, a constant $C > 0$ and $r(\mathbf{m}) = \prod_{j=1}^n \max\{1, |m_j|\}$. Then

$$E_N = \sum_{\mathbf{m} \neq \mathbf{0}} \delta_N(\mathbf{m}, \mathbf{v}) c_{\mathbf{m}}, \quad (6)$$

where

$$\delta_N(\mathbf{m}, \mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{m} \cdot \mathbf{v} = 0 \pmod{N} \\ 0, & \text{otherwise.} \end{cases}$$

Thus the choice of \mathbf{v} and N in the definition of the Koroobov filter (4) removes from the error (6) all terms with coefficients $c_{\mathbf{m}}$ for which $\mathbf{m} \cdot \mathbf{v}$ is not a multiple of N .

Furthermore, consider a sequence of lattice rules (4) with increasing $N = N_0, N_1, \dots$, for which there exist $A > 0$ and γ such that, for all i , every non-zero \mathbf{m} for which $\mathbf{m} \cdot \mathbf{v}$ is a multiple of N_i satisfies

$$r(\mathbf{m}) > \frac{AN_i}{\log^\gamma N_i}. \quad (7)$$

Then for $f \in \mathcal{E}^k$, the error satisfies

$$E_N = \mathcal{O}\left(\frac{(\log N)^{k\gamma^*}}{N^k}\right),$$

where γ^* is the smallest γ for which (7) holds. Thus under these conditions, error bounds are obtained which

are better than the $\mathcal{O}(N^{-1/2})$ bound of MC. The periodicity conditions on f can be alleviated by “periodizing” the function, i.e., find a function $h \in \mathcal{E}^k$ for $k > 1$ such that $I(h) = I(f)$. An example of $h \in \mathcal{E}^2$ is given by

$$h(x_1, \dots, x_n) = f(1 - 2|x_1 - \frac{1}{2}|, \dots, 1 - 2|x_n - \frac{1}{2}|). \quad (8)$$

Cranley and Patterson [1] randomize (4) to obtain a stochastic family of rules. Let

$$K_N(\beta) = \frac{1}{N} \sum_{i=1}^N f(\frac{i}{N}\mathbf{v} + \beta,) \quad (9)$$

where β is a uniformly distributed random vector. Using a random sample set of size q ,

$$\bar{K}_N = \frac{1}{q} \sum_{j=1}^q K_N(\beta_j) \quad (10)$$

preserves the integration properties of (4) and allows for an estimation of the standard error by $\frac{1}{q(q-1)} \sum_{j=1}^q (K_N(\beta_j) - \bar{K}_N)^2$.

This allows for a parallelization where the controller averages over the sample sets.

3.2 Equidistributed sequences

A (deterministic) sequence of points x_1, x_2, \dots in $[a, b]$ is *equidistributed* in $[a, b]$ if

$$\lim_{N \rightarrow \infty} \frac{b-a}{N} \sum_{i=1}^N f(x_i) = \int_a^b f(x) dx$$

for all bounded Riemann-integrable functions $f(x)$. The following can then be shown [2].

- If θ is an irrational number, then the sequence $x_i = \{i\theta\}$, $i = 1, 2, \dots$ is equidistributed in $[0, 1]$.
- Let $\theta_1, \theta_2, \dots, \theta_n$ be n irrational numbers such that $1, \theta_1, \theta_2, \dots, \theta_n$ are linearly independent over the rational numbers, i.e., $\lambda_0 + \lambda_1\theta_1 + \dots + \lambda_n\theta_n \neq 0$ for rational λ coefficients not all zero. Then the points $P_i : (\{i\theta_1\}, \{i\theta_2\}, \dots, \{i\theta_n\})$ are equidistributed over the unit hypercube.

Consequently,

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N f(\{i\theta_1\}, \dots, \{i\theta_n\}) = \int_0^1 \dots \int_0^1 f(\mathbf{x}) dx_1 dx_2 \dots dx_n$$

for all bounded Riemann-integrable functions $f(\mathbf{x})$.

Error bounds can be obtained which are better than $\mathcal{O}(N^{-\frac{1}{2}})$. Richtmyer obtained a bound of order $\mathcal{O}(N^{-1})$ for $f \in \mathcal{E}^k$.

Alternative methods for generating equidistributed sequences (of n -tuples) include the low-discrepancy methods of Halton and of Sobol.

4 Results

We built two methods conforming to the meta-algorithm of Section 2, a crude Monte-Carlo method described in Section 4.1 below; and a QMC method described in 4.2.

4.1 Asynchronous MC

A crude Monte-Carlo method is called by workers and controller to compute the sample results (Q,S,N); this code was used mainly with the goal of experimentation in the derivation of the parallel structure and as a reference for later results. As the sequence $\{N_k\}$ we used $\{N_k = N_0 \rho^k \mid k = 0, 1, \dots\}$ with $N_0 = 100 + 10n^2$ and $\rho = 1.2$. The sample set sizes N_k are incorporated by the processes in round-robin fashion, i.e., process i generates sample sets of sizes $N_i, N_{i+p}, N_{i+2p}, \dots$, where p is the number of processes. However, when the cumulative (global) number of evaluation points exceeds $N_{\max}/2$, then the sample set sizes are no longer increased (in order to alleviate that the total number of points would exceed N_{\max} significantly at termination).

The program was implemented in C using the MPI parallel environment. Figure 2 gives timing results from this method run on a network (LAN) of Ultra-10 Sun Sparc workstations, for a problem of the form (1) of dimension $n = 5$. The integration limits and correlation matrix are $\mathbf{a} = (0, 0, 1.7817, 1.4755, -\infty)$, $\mathbf{b} = (\infty, 1.5198, \infty, \infty, 1.5949)$ and

$$\Sigma = \begin{pmatrix} 1 & -0.70711 & 0 & 0 & 0 \\ -0.70711 & 1 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 1 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 & 1 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 & 1 \end{pmatrix}.$$

In order to compare equivalent amounts of work each time we forced termination with $N_{\max} = 2M$ (and tolerated error $\mathbf{eps} = 0$). Note that one process is used per workstation (processor). Good speedups are observed for up to 8 processes; for larger numbers of processes no further improvement is observed for this relatively small example, due to the communication overhead on the ethernet. For the results and error estimates computed according to (2) and (3), respectively, the ac-

tual errors range between $1.3 \cdot 10^{-8}$ and $1.0 \cdot 10^{-6}$ (for $I \approx 2.863088 \cdot 10^{-3}$) and error bounds are returned ranging from $6.5 \cdot 10^{-6}$ to $6.8 \cdot 10^{-6}$.

4.2 Asynchronous QMC

A Korobov/ Richtmyer method (from Genz' MVNDSTPACK [7]) is used to calculate the sample results. The algorithm calculates an array of results $K_{N_k}(\beta_j)$ given by (9) where $N_k, k = 0, 1, \dots$ is the number of points in lattice rule k and $\beta_j, 1 \leq j \leq q$ are uniformly distributed random n -vectors. In our parallel algorithm, q is a parameter of the implementation which can be varied for testing purposes. If, for the sake of simplicity, we assume that q is a multiple of p , then $s = p/q$ is the number of sample sets each process will evaluate at a time. Each process computes a block column of width s of the array; the controller averages over each row of the array. The controller thus keeps a vector of the current $\bar{K}_{N_k}, k = 0, 1, \dots$ defined in (10), which is averaged element-wise over all processes which contributed an update.

All processes start by computing \bar{K}_{N_0} and its corresponding standard error estimate based on their samples. The sample results are sent to the controller, which updates the global \bar{K}_{N_0} and corresponding global standard error. All processes then go through a sequence of computing $\bar{K}_{N_k}, k = 1, 2, \dots$ based on their samples. Upon receiving a contribution for lattice rule k , the controller updates the current \bar{K}_{N_k} .

MVNDSTPACK incorporates a sequence of 25 rules in DKBVRC, with numbers of points N_k varying from $N_0 = 31$ to $N_{24} = 601942$. The lattice rules are used exclusively for problems of up to dimension 20. For dimensions $n > 20$, a Richtmyer subvector is used in the components beyond the 20th (consisting of the square roots of successive prime numbers). The periodizing transformation (8) is incorporated.

Figure 2 also gives timing curves obtained with the parallel Korobov method for the above listed 5-dimensional problem. One set of results is obtained on our workstation network of Ultra-10 Sparcs. For this run the number of samples s was set to 1 for each process. The other curve depicts results of a run on the IBM-SP machine at Argonne National Laboratory; for this run $s = 7$ on each process. Both tests terminate with $N_{max} = 2M$. The observed speedups are good. The actual errors for the runs with 1 to 14 processes range between $2 \cdot 10^{-9}$ and $2.8 \cdot 10^{-8}$.

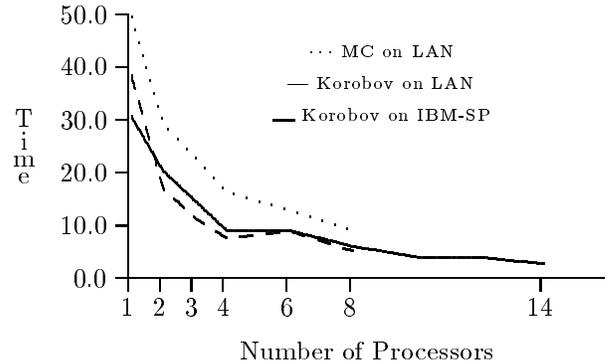


Figure 2: Times (s) for parallel methods

5 Concluding Remarks and Future Work

In this paper we presented a meta-algorithm for asynchronous sampling methods. Based on the meta-algorithm we derived efficient parallel methods for the computation of multivariate normal probabilities, including a Monte-Carlo type method and a method based on a randomized Korobov and a Richtmyer sequence.

Use of the following parallel methods is advised under the indicated circumstances:

- Adaptive method with load balancing: for moderately low dimensions;
- Lattice rules (randomized Korobov type): for moderately high dimensions;
- Equidistributed sequences: for high dimensions.

Future work will include a detailed study of the execution time, accuracy and error estimates obtained with the parallel Korobov method, as a function of the number of samples per process and the number of processes. We also plan a comparative study with alternative strategies. The resulting method is a candidate for inclusion in the ParInt software package for parallel multivariate integration.

ACKNOWLEDGEMENT. We acknowledge the support of the Mathematics and Computer Science Division at Argonne National Laboratory, for our use of the IBM-SP system.

References

- [1] CRANLEY, R., AND PATTERSON, T. N. L. Randomization of number theoretic methods for multi-

- ple integration. *SIAM J. Numer. Anal.* 13 (1976), 904–914.
- [2] DAVIS, P. J., AND RABINOWITZ, P. *Methods of Numerical Integration*. Academic Press, New York, 1984.
- [3] DE DONCKER, E., GUPTA, A., BALL, J., EALY, P., AND GENZ, A. ParInt: A software package for parallel integration. In *10th ACM International Conference on Supercomputing* (1996), Kluwer Academic Publishers, pp. 149–156.
- [4] DE DONCKER, E., GUPTA, A., GENZ, A., EALY, P., LIU, J., AND SUREKA, A. Use of ParInt for the parallel computation of statistics integrals. *Computing Science and Statistics* 27 (1996).
- [5] GENZ, A. Numerical computation of multivariate normal probabilities. *J. Comp. Graph. Stat.* 1 (1992), 141–149.
- [6] GENZ, A. Comparison of methods for the computation of multivariate normal probabilities. *Computing Science and Statistics* 25 (1993), 400–405.
- [7] GENZ, A. MVNDST: Software for the numerical computation of multivariate normal probabilities, 1998. Available from the website with url “<http://www.sci.wsu.edu/math/faculty/genz/homepage>”.