95-255

E11-95-255

T.A.Merkulova, G.G.Takhtamyshev

ADAPTIVE RANDOM NUMBER GENERATOR SMART

1995

Introduction. Monte Carlo method is well-known universal tool for integration of a function over multidimensional space. In general, the method can be applied for integration any kind of function, in any number of dimensions space and for any shape of integration volume.

Applying the Monte Carlo method one has to drop points randomly into multidimensional volume and to calculate value of the function at the each point. Repeating the procedure many times and summing the values of the function, one can estimate the average of the function and then find the value of the integral. For the simplicity reasons everywhere in this paper we will assume the integration volume to be unit hypercube - in this case estimation of integral just coincide with estimation of average value of the function. The random points can be dropped in the hypercube uniformly, or in accordance with some pre-defined density probability. Different kinds of variance reduction techniques can be used in difficult cases, when a substancial amount of time is requested for the integration. (See, e.g. Ref. [1] for a description of such methods.)

An adaptive approach to the problem can also be recommended, as one of such methods. Basic idea of adaprtive approach is quite simple. During the process of points dropping and estimating of average of the function one can start to gather the information about the function behavior, and, after some period of time, one can start to drop more points into region (or regions), where the function has maximum (or maxima). Such a non-uniformity random points in the volume should be compensated by some additional weight, assigned to each point. Such procedure, when law of distribution of random numbers in the volume is changing in accordance with a specific shape of the integrand function, usually called as adaptive Monte Carlo method. One of such algorithms was developed by B.Lautrup [2]. Realizations of this algorithm are known as program RIWIAD in the CERN Library [3] and program D01GBF in NAG Library [4]. One more algorithm was suggested by G.P.Lepage (Ref. [5]).

In this paper we present another way of realization of adaptive method. In following sections of the paper short description of the algorithm will be given, as well as results of tests and comparison with results, obtained when RIWIAD and D01GBF were used.

The first version of our algorithm was published earlier (See Ref. [6]).

Algorithm. In short words, the procedure arranged in the following way. All k intervals $[0,1]$ for each dimension of the k-dimensional hypercube are divided by some definite numbers of subintervals. Lengths of all subintervals are all equal at the beginning. The process of dropping random point into the whole interval consists of two steps. First, the number of subinterval is randomly chosen and then the point is randomly dropped at this specific subinterval. The numbers of subintervals here and then are chosen with equal probability. Also the points always are dropped into subinterval uniformly. But the length of subinterval is matter of change during the process of calculation. Such changing all the lengths all the intervals is performed after some certain portion of points is dropped. Typical number of points for such set is 1000.

So, there are k intervals $[0,1]$ each of them consists of m subintervals with length $l(i,j)(i=1,...,m, j=1,...,k)$. Respectively, $m*k$ summators $s(i,j)$ and equal number of counters $n(i,j)$ are being stored during each set of N points. It means, when point is dropped, and the calculated value of the function is f, then the value of product $f*m*l(i,j)$ is added to k summators $s(i,j)$, and each of k counters $n(i,j)(j=1,...,k)$ is incremented by 1, assuming, that the point belongs to subinterval number i for dimension number j. Also the weighted averages of coordinate $x(j)$ within subinterval number i are calculeted and stored in the array $v(i,j)$.

After end of set (N points are dropped) the program starts to analyze the summators $s(i,j)$ together with counters $n(i,j)$. The analysis is being performed independently for each number of dimension j, so the subjects of analysis are just $2*k$ one-dimensional arrays $s(i)$ and $n(i)$. The essense of the analysis for each given j is as follows.

On the first step the program finds the maximum and minimum among all $s(i)$ and calculate the average value of all $s(i)$. We will use notations $s1$, $s2$ and $s3$ for these values. Also, sum of all $s(i)$, except the maximum one is calculated and we will call this value $s4$.

At the next step of the analysis a "level of difficultiness" (LOD) is assigned to this dimension number j. The assignment is done on base of the values $s1, s2, s3, s4$ and some constants $c1, c2, c3$, as well. The rules for the assignment are the following. The initial value of LOD is 0. Then, if $s1/s2$ is larger than c1, LOD is set to be equal 1. After that, if at least one of the values $s1/s3$ or $s3/s2$ is larger than $c2$, LOD becames equal 2. And, finally, if $s1$ is greater than $s4*c3$, the LOD is equal 3. Usually, the values of the constants $c1, c2, c3$ were taken as 3, 5 and 2.

After all LOD's are defined, the rearranging of the lengths of all the intervals $[0,1]$ is being performed. This process is going differently for each given value of LOD. In the procedure of the lengths of subintervals rearranging an important role plays the subinterval, containing the maximum value of s. We will say, that the bin, containing this value has number h.

For $LOD = 3$ the procedure is slightly different in case, if $h = 1$, or $h = m$ (case A), or, if $1 < h < m$ (case B). In the case A all $m-1$ subintervals, except the subinterval number h, are united in one subinterval, whereas subinterval number h is subdivided into $m-1$ equal new subintervals. In the case B all subintervals with numbers less than h are united into one, subinterval number h is subdivided into $m-2$ equal subintervals and all subintervals with numbers greater than h are also united into one subinterval.

For $LOD = 2$ the subinterval number h is divided into two equal subintervals. Then the program is looking for two neighbors subintervals, sum of values of $s(i)$ in which is the least. When found, two such neighbors are united into one subinterval.

And for the case $LOD = 1$ bounds of all the subintervals are moved. The basic idea for such movements is to set value of the weighted average of the coordinate $x(i)$ for given subinterval j closer to the center of the new subinterval. As was said earlier, the averages, which are needed for such a procedure are stored in the array $v(i,j)$.

Tests and results. This algorithm was realized as a subroutine, written in FORTRAN, and set of multidimensional integration tests was performed. The tests were made for the several kinds of functions and the obtained results were compared with results, obtained with RIWIAD and D01GBF. As a main characteristic of program's work we used convergency speed, i.e. the dependence of relative accuracy ε on the total number N of multidimensional points, used for calculation.

Since results, obtained with different functions were found to be more or less similar, here we present results of integration only one, Gauss-like function F in the unit hypercube of dimension k, for $k = 5, 7, 10$.

$$F(\vec{x}) = \frac{1}{\sqrt{2\pi D}} \exp[-\frac{1}{2D}\sum_{i=1}^{k}(x_i - a_i)^2],$$

where $\vec{x} = (x_1, \cdots, x_k)$ is a vector of coordinates in the hypercube, $\vec{a} = (a_1, \cdots, a_k)$ is a randomly generated vector. Results, presented on Fig.1 and Fig.1a were obtained with SMART and correspond to the cases $D = 0.01$ and $D = 0.04$, respectively.

For the comparison, the dependence $\varepsilon(N)$, obtained at the calculation of the same integral by program D01GBF from NAG Program Library, is presented on the Fig.2.

One can see that in all these examples speed of convergency for SMART close to $1/N$. Moreover, comparing data from Fig.1 and Fig.2 one can say, for case $k = 5$ SMART and D01GBF give approximately the same accuracy, but results of D01GBF are substantially worse in case $k = 7$ and are quite poor for $k = 10$.

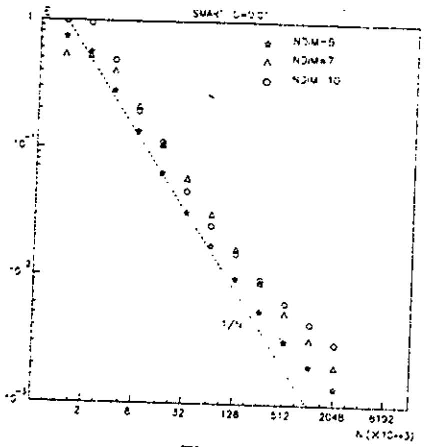

Fig.1.

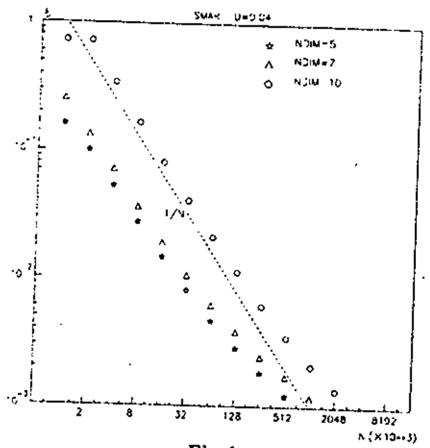

Fig.1a.

On the Fig.3 there is the dependence $\varepsilon(N)$ for the similar calculations produced with the program RIWIAD at $k = 5$. The speed of convergency in this case also close to $1/N$. Unfortunately, we didn't manage to get any reasonable result for higher dimensionalities ($k = 7$, $k = 10$).

The calculations were performed on computers VAX, CONVEX and SUN. The purpose of this work was to test the quality of the algorithm, its feasibility to the calculations of multidimensional integrals. Due to this, we didn't try to measure and compare the time, needed for each program to get result. But the rough estimation showed that the calculations with SMART generally were executed faster then with RIWIAD, or with D01GBF.

One of the possible applications of fast convergency integration methods in physics of elementary particles is integration the matrix element squared over the phase space. Such integration should be performed in order to obtain decay rate of a particle, or cross-section of a reaction. Generator SMART was used for calculations of cross-section of production Higgs boson in electron-positron collision. The matrix element squared for this three-body reaction can be written in a rather simple form:

$$T = F\frac{s_{12}}{((t_1 - M_W^2)(t_2 - M_W^2))^2}.$$

Here s_{12} — scalar product of 4-momenta of electron and positron after interaction, $t_1(t_2)$ — transferred momentum squared between electrons (positrons) before and after interaction, M_W — mass of the W-boson, and F — some constant, which has no effect on convergency rate. Integration has to be performed

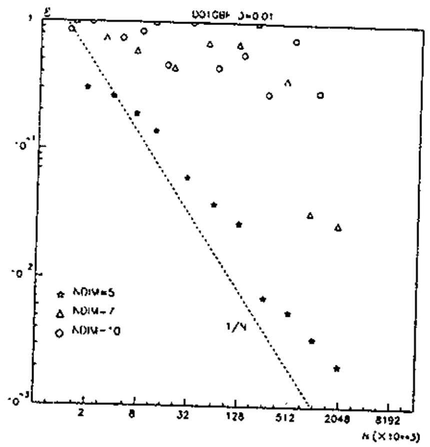

Fig.2.

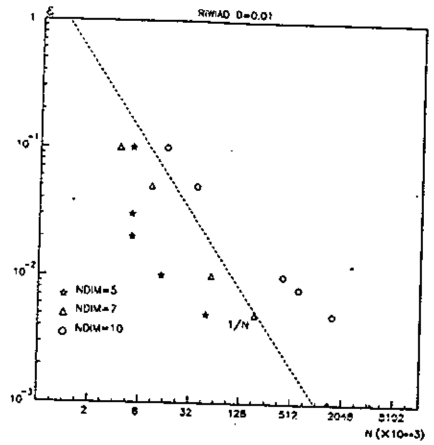

Fig.3.

over three body phase space, and, if being done with usual method, request rather long time for achievment of reasonable accuracy. On the Fig.4 we again present the dependence $\varepsilon(N)$ obtained with regular random number generator, and with SMART. Again, as one can see, SMART provides speed of convergency close enough to $1/N$.

Another field in high energy physics, where such generator can be successfully applied is simulation of an experiment, where acceptance (or efficiency of registration) is rather small value. One of example of such experiment is proposed measurement of elastic neutron-neutron scattering in the colliding beams of nuclear pulse reactor (See Ref. [7]). In the experiment two scattered neutrons should be detected at rather large distance from the point of collision. Due to this fact, acceptance for the registration is pretty small, and, as a result, simulation of the experiment with reasonable accuracy seems to be difficult problem. Effectively, this simulation was equivalent to calculation of 9-dimensional integral into the some volume V. Attempts to calculate this integral with help of program DGB01F didn't give any reasonable result. The integrand can be written in the following way.

$$F = \varphi(\vec{v},\vec{r},t)\varphi(\vec{w},\vec{r},t)\xi(t,r_2)\eta(\vec{v},\vec{w})\zeta(\vec{v}),$$

where $\vec{v} = (v_1, v_2, v_3)$, $\vec{w} = (w_1, w_2, w_3)$, $\vec{r} = (r_1, r_2, r_3)$. Integration was performed over the variables $v_2, v_3, \vec{w}, \vec{r}, t$. Function v_1 is the following

$$v_1 = [Y_1 w_1 + Y_2(v_2 + w_2) + Y_3(v_3 + w_3) - Y_1^2 - Y_2^2 - Y_3^2 - v_2 w_2 - v_3 w_3]/(w_1 - Y_1),$$

where $Y_i = (CY_i - r_i)/(\tau - t), i = 1, 2, 3.$

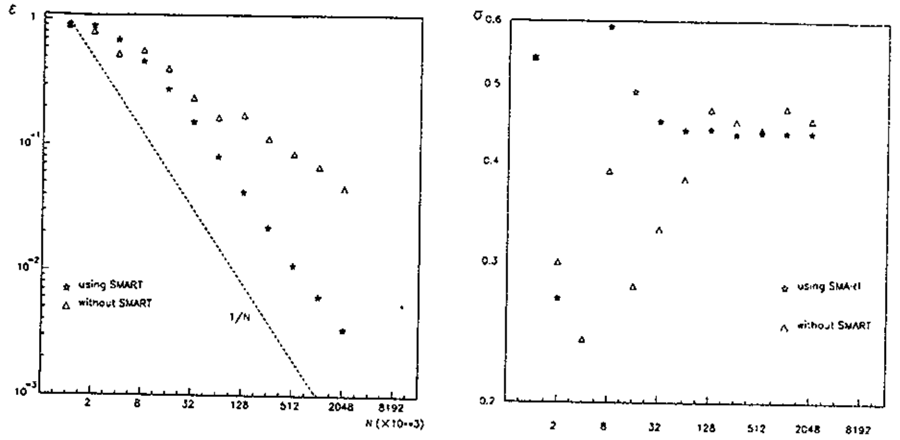

Fig.4.

The other functions, used in the expression, were

$$\varphi(\vec{v},\vec{r},t)=\alpha(t)\beta(\vec{v})\gamma(\vec{r}),$$

where

$$\alpha(t)=\frac{t}{t_0^2}e^{-\frac{t}{t_0}},\beta(\vec{v})=\frac{1}{(2\pi)^{\frac{3}{2}}}e^{-\frac{|\vec{v}|^2}{2a^2}},\gamma(\vec{r})=\frac{1}{(2\pi)^{\frac{3}{2}}}\exp(-\frac{r_1^2}{2b^2}-\frac{r_2^2}{2c^2}-\frac{r_3^2}{2d^2}),$$

$$\xi(t,r_2)=\frac{(CY_2-r_2)}{(\tau-t)^4},\eta(\vec{v},\vec{w})=[(v_1-w_1)^2+(v_2-w_2)^2+(v_3-w_3)^2]^{\frac{1}{2}};$$

$$\zeta(\vec{v})=(v_1^2+v_2^2+v_3^2)^{-\frac{1}{2}}.$$

Here CY_i,τ,t_0,a,b,c,d — are the constants.

On the Fig.5 we present the results of calculations of this integral with help of SMART. One can see, the problem is rather difficult, because only after 15 mln. points the result got some stability.

Conclusions. Rather simple adaptive algorithm was realized as a FORTRAN subroutine, which can be used either as a random number generator, or as a multidimensional integration routine.

Many tests, which have been performed, shown, that the program provides in difficult cases speed of convergency close to $1/N$ and in many cases works better, than other known routines.

Authors are grateful to Dr. G.Ososkov for many valuable discussions on the subject of this paper.

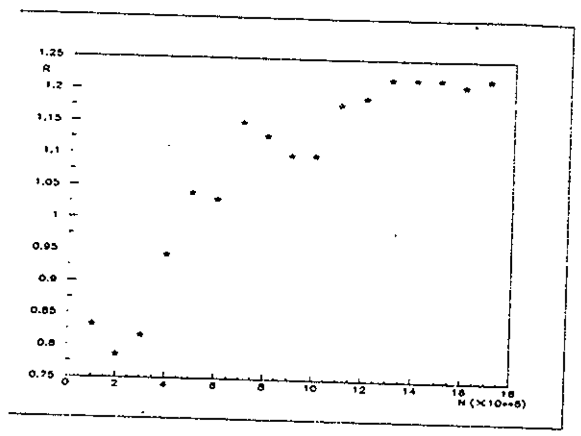

Fig.5.

References

[1] J. M. Hammersley and D. C. Handscomb, Monte Carlo Methods, Methuen,London,1964.

[2] B. Lautrup, An Adaptive Multi-dimensional Integration Procedure. Proc. 2nd Coll. on Advanced Methods in Theoretical Physics. Marseille,1971.

[3] CERN Program Library. D114.

[4] NAG Fortran Library Manual. D01GBF.

[5] G.P.Lepage, Journ. Comp. Phys. v.27, 192 (1978)

[6] G.Takhtamyshev, preprint JINR (in Russian) P11-87-473 (1987)

[7] A.D.Budnik et al., Proc. of the Amer. Nucl. Soc. Topical Meeting "Physics, Safety and Applications of Pulse Reactors", Nov. 13-17, 1994, Washington, DC, p.343-346.

Received by Publishing Department
on June 14, 1995.