L.D. Fosdick and
A.K. Cline, Editors

Algorithm 440

A Multidimensional Monte Carlo Quadrature with Adaptive Stratified Sampling [D1]

L.J. Gallaher (Recd. 10 Dec. 1970, 20 July 1971)
Rich Electronic Computer Center, Georgia Institute of Technology, Atlanta, GA 30319

Key Words and Phrases: Monte Carlo quadrature, stratified sampling, adaptive quadrature, sequential stratification
CR Categories: 5.16, 5.5
Language: Algol

Description

This procedure evaluates the n-dimensional integral

$$\int_{V(a,b)} v(\mathbf{x})\, d\mathbf{x} = \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} v(x_1, x_2, \cdots, x_n)\, dx_n \cdots dx_2\, dx_1$$

by the Monte Carlo method. The variance reduction scheme used here is a form of stratified sampling.

The advantages of stratified sampling are well known [1], and the concept of optimum stratification is discussed in most text books on Monte Carlo methods [2, 3, 4]. The advantages of adaptive quadrature are also well known, and many such algorithms have been published in Communications and elsewhere [5, 6, 7]. Combining adaptive quadrature with stratified sampling is a straightforward process [8, 9].

The workings of this procedure are somewhat similar to Algorithm 303 [6]. Algorithm 303 is one-dimensional, and while it can be used for multidimension integrals by recursive calls, for more than approximately six dimensions the number of evaluations of the integrand becomes intolerable. The goal of the algorithm given here is to try to overcome this defect of Algorithm 303 and other algorithms like it.

The procedure works as follows:
1. A set of samples is taken, uniformly stratified throughout the entire volume being integrated.
2. Based on the variance in these samples, a decision is made as to whether more samples are needed.
3. If more samples are needed, the volume is cut in half and the entire procedure (but with fewer samples) is repeated on each half, recursively, the halvings being repeated as required. The choice of axis for the halving is based on samples of the gradient.

The result of this process is that the overall stratification is not uniform, but approaches optimum as more and more samples are taken, since more halvings (thus more samples) are taken in the regions of high variance.

A certain amount of caution must be used in the choice of the input parameter m ($m + n$ is the number of samples taken initially). If the function being integrated is reasonably smooth, relatively low values of m (say 5 to 10) are satisfactory. If $v(\mathbf{x})$ is known to have sharp peaks, ridges, valleys, or pits, then large values of m will be necessary in order to avoid missing these high and low spots. A rough rule is that m should be inversely proportional to the error tolerance and proportional to the logarithm of volume of anomalous regions. If V_A is the fractional volume of the anomalous regions and E_r is the relative error tolerance, then the empirical rule $m \gtrsim (-2\ln(V_A))/E_r$ has proved satisfactory. For this quadrature algorithm tc be useful, the results should be insensitive to the users choice of m, and this has been observed provided m is not chosen too small. (This difficulty about the occasional need to choose m shrewdly is characteristic of all adaptive quadrature schemes, whether Monte Carlo or "exact" methods such as Romberg, Simpson, or others.)

As a test of this procedure, 100 evaluations were made of the volume of 1/32 of a hypersphere in five dimensions (in rectangular coordinates), i.e.

$$\int_0^R\int_0^R\int_0^R\int_0^R \left\{ \begin{array}{l} \text{if } \sum\limits_{1\le i\le 4} x_i^2 \ge R^2 \text{ then } 0 \\ \text{else } (R^2 - \sum\limits_{1\le i\le 4} x_i^2)^{1/2} \end{array} \right\} dx_1\, dx_2\, dx_3\, dx_4 ,$$

with 3% accuracy requested. A histogram is given below of the values obtained.

Number of occurrences	4	0	8	14	18	16	20	14	5	1	
I_{obs}/I_{exact}	0.94	0.95	0.96	0.97	0.98	0.99	1.00	1.01	1.02	1.03	1.04

Here I_{obs} is the value observed, I_{exact} is the correct value. The initial value of m was 120, and the average number of function evaluations per integral was 1427. The standard error for the 100 evaluations was approximately 2%. For corresponding accuracy, about 4.5 times as many samples would have been needed by unstratified uniform sampling.

Finally it should be pointed out that the results given by adaptive stratification are not entirely unbiased in the usual sense of the Monte Carlo method. There is, in fact, a biasing in favor of regions having low values of the magnitude of the gradient. However, this bias should normally be expected to be much smaller than the requested error tolerance.

Acceptable random number generators for this algorithm may be found in [10].

References

1. Cochran, William G. *Sampling Techniques.* Wiley, New York, 1953 (2nd ed. 1963).
2. Kahn, Herman. Applications of Monte Carlo. RM-1237-AEC, Rand Corp., Santa Monica, Calif., Apr. 1954 (revised Apr. 1956).
3. Hammersley, J.M., and Handscomb, D.C. *Monte Carlo Methods.* J. Wiley, New York, 1964.

4. Spanier, Jerome, and Gedbard, Ely M. *Monte Carlo Principles and Neutron Transport Problems.* Addison-Wesley, Reading, Mass., 1969.
5. McKeeman, William M. Algorithm 145: Adaptive numerical integration by Simpson's rule. *Comm. ACM 5* (Dec. 1962), 604.
6. Gallaher, L.J. Algorithm 303: An adaptive quadrature procedure with random panel sizes. *Comm. ACM 10* (June 1967), 373–374.
7. Lyness, J.N. Algorithm 379: SQUANK (Simpson quadrature used adaptively—noise killed). *Comm. ACM 13* (Apr. 1970), 260–263.
8. Halton, J.H., and Zeidman, E.A. Monte Carlo integration and sequential stratification. Comput. Sci. Tech. Rep. 13, U. of Wisconsin, Madison, Wis. 1968.
9. Zeidman, E.A. The Evaluation of multidimensional integrals by sequential stratification. (to be published).
10. Halton, John H. A retrospective and prospective survey of the Monte Carlo method. *SIAM Rev. 12*, 1 (Jan. 1970), 1–63.

Algorithm

```
real procedure quadmc (n, a, x, b, vx, esq, m, Vab, rn);
  value n, esq, m, Vab;
  integer n, m; real vx, esq, Vab, rn;
  array a, x, b;
comment The procedure parameters are:
  n - number of dimensions, n ≥ 1
  a - array of n lower bounds
  x - array of n position coordinates of which v(x_1, x_2, ..., x_n) is
      a function, x is called by name
  b - array of n upper bounds (it is not required that b_i > a_i)
  vx - function to be integrated, vx must be a function of the array
      x (Jensen's device) and be called by name
  esq - square of the absolute error tolerance for the quadrature
  m - the number of samples to be taken at the first level is m + n,
      m ≥ n
  Vab - volume being integrated, i.e. Vab = ∏_{1≤i≤n} | (b_i − a_i) |
  rn - procedure to give a new random number uniform on the
      open interval zero to one (0 < rn < 1) each time referenced,
      called by name.
  All of these parameters are input parameters to be supplied by
      the user.
  Some of the local variables of this procedure are:
  vbar - average value of v(x) for m + n samples, i.e.
```

$$\bar{v} \equiv \frac{1}{m+n} \sum_{1 \le i \le m+n} v(\mathbf{x}_i)$$

```
  vsqbar - average value of v(x)^2 for m + n samples, i.e.
```

$$\overline{v^2} \equiv \frac{1}{m+n} \sum_{1 \le i \le m+n} v(\mathbf{x}_i)^2$$

```
  ssq - the square of the standard error of the mean (of the integral)
      for m + n samples, i.e.
```

$$\sigma^2 = \frac{(\overline{v^2} - \bar{v}^2)}{(m+n-1)} V_{ab}^2$$

```
  vi - value of v(x) at ith sample, i.e. v(x_i)
  vip - a value of v(x) such that 2 | vip − vi | is a sample of the
      magnitude of the ith component of the average normalized
      gradient, 1 ≤ i ≤ n
  it - vector of shuffled integers 1 to m
  j - array of n different vectors of shuffled integers 1 to m used in
      constructing the (uniform) stratification
  cl - point on the lth axis that divides the volume of integration in
      half for the next recursive level, i.e. cl = (b[l] − a[l])/2,
  l - index of the axis having the largest in magnitude sample of
      the component of the average normalized gradient.
  end of comment;
begin
  integer l; real vbar, ssq;
  if m < n then m := n;
  begin
    real gm, vi, vip, vsqbar;
    integer itemp, ir, k, i;
    array h[1:n];
    integer array j[1:n, 1:m], it[1:m];
    for i := 1 step 1 until m do it[i] := i;
    for k := 1 step 1 until n do
    begin
      h[k] := (b[k] − a[k])/m;
      for i := 1 step 1 until m do
      begin
        ir := entier (rn × m) + 1;
        comment 0 < rn < 1;
        itemp := it[i]; it[i] := it[ir]; it[ir] := itemp;
      end;
      for i := 1 step 1 until m do j[k, i] := it[i];
    end;
    l := 1;
    vsqbar := vbar := gm := 0;
    for i := 1 step 1 until m do
    begin
      for k := 1 step 1 until n do
      x[k] := a[k] + (j[k, i] − rn) × h[k];
      vi := vx;
      vbar := vbar + vi;
      vsqbar := vsqbar + vi ↑ 2;
      if i ≤ n then
      begin
        comment Sample the gradients;
        x[i] := x[i] + abs(b[i] − a[i])/2×
        (if x[i] < (b[i] + a[i])/2 then 1 else −1);
        vip := vx;
        vbar := vbar + vip;
        vsqbar := vsqbar + vip ↑ 2;
        if gm < abs(vip − vi) then
        begin
          l := i; gm := abs(vip − vi);
        end;
      end;
    end;
    vbar := vbar/(m + n);
    vsqbar := vsqbar/(m+n);
    ssq := Vab ↑ 2 × (vsqbar − vbar ↑ 2)/(m+n−1);
  end;
  if ssq ≤ 2 × esq then quadmc := vbar × Vab else
  begin
    real temp, cl, al, bl;
    m := m × 0.707;
    if m < ssq/esq then m := ssq/esq;
    comment The author is indebted to the referee's
      discussions pointing out the significance of maintaining
      m ≳ ssq/esq;
    esq := esq × ssq/(ssq − esq);
    al := a[l]; bl := b[l];
    b[l] := cl := (bl + al)/2;
    temp := quadmc(n, a, x, b, vx, esq/2, m, Vab/2, rn);
    b[l] := bl; a[l] := cl;
    temp := quadmc(n, a, x, b, vx, esq/2, m, Vab/2, rn) + temp;
    a[l] := al;
    quadmc := (temp × ssq + esq × vbar × Vab)/(ssq+esq);
  end;
end of quadmc
```

Algorithm 441

Random Deviates from the Dipole Distribution [G5]

Robert E. Knop [Recd. 12 Jan. 1971, 7 May 1971, 23 Aug. 1971, and 8 Mar. 1972]
Department of Physics, The Florida State University, Tallahassee, FL 32306

Key Words and Phrases: random number, probability density, probability distribution, Dipole distribution, Cauchy distribution, simulation, Monte Carlo
CR Categories: 3.17, 5.5
Language: Fortran

Description

The function subprogram *DIPOLE* returns a random deviate $-\infty < z < \infty$ sampled from the two parameter ($R^2 < 1$, α arbitrary) family of density functions:

$$f(z) = 1/(\pi(1+z^2)) \\ + R^2 \times ((1-z^2) \times cos(2\alpha) + 2 \times z \times sin(2\alpha))/(\pi \times (1+z^2)^2)$$

The cumulative distribution function is:

$$F(z) = (1/2) + (1/\pi) \times tan^{-1}(z) \\ + R^2 \times (z \times co(s(2\alpha - sin(2\alpha))/(\pi \times (1+z^2))$$

Densities of this type commonly occur in the analysis of resonant scattering of elementary particles. We note that when $R = 0$ we have the Cauchy [1] or Breit-Wigner [2] density. When $R = 1$ and $\alpha = 0$ we have the single channel dipole density.[1] The dipole density with free parameters has been proposed to describe multichannel resonant scattering [3].

The algorithm begins by sampling the random vector (x, y) from a density uniform over the unit disk. The center of the unit disk is then displaced from the origin by the transformations $u = x + R \times cos(\alpha)$ and $v = y + R \times sin(\alpha)$. Letting $u = r \times cos(\theta)$ and $v = r \times sin(\theta)$ we can find the marginal density of θ:

$$f(\theta) = 1/(2\pi) \times \left(\int_0^{r_+^2} ds + \int_0^{r_-^2} ds \right)$$

where the limits of integration for r are given by:

$$r_\pm(\theta) = R \times cos(\theta - \alpha) \pm (1 - R^2 \times sin^2(\theta - \alpha))^{\frac{1}{2}}.$$

The marginal density of θ is thus:

$$f(\theta) = (1 + R^2 cos(2 \times (\theta - \alpha)))/\pi$$

for $-\pi/2 < \theta < \pi/2$. The transformation $z = tan(\theta) = v/u$ then yields the dipole density function. Other densities which could be easily sampled by computing rational functions of u and v are suggested by transformations such as $z = tan^2(\theta)$, $sin^2(\theta)$, $sin(2 \times \theta)$, or $1/|sin(2 \times \theta)| - 1$.

Function *DIPOLE* has two arguments which must be calculated by the calling program, $A = R \times cos(\alpha)$ and $B = R \times sin(\alpha)$. *DIPOLE* calls the function $R11(D)$ which must return a random deviate from the uniform distribution over the interval $(-1,1)$. D represents a dummy argument.

This work was supported in part by the U.S. Atomic Energy Commission.

[1] The density is named after the analytic property of having poles of order 2 in the complex plane. See [2].

The author wishes to express his gratitude to Prof. R.G. Glasser of the University of Maryland for comments concerning this algorithm.

References

1. von Neumann, J. Various techniques used in connection with random digits. In Nat. Bur. Standards Appl. Math. Ser. 12, U.S. Gov. Print. Off., Washington, D.C., 1951, p. 36.
2. Goldberger, M.L., and Watson, K.M. *Collision Theory*. J Wiley, New York, 1964, Chap. 8.
3. Rebbi, C., and Slansky, R. Doubled resonances and unitarity. *Phys. Rev. 185*, 1838 (1969).

Algorithm

```
      FUNCTIØN DIPØLE(A,B)
10    X = R11(D)
      Y = R11(D)
      IF(1.0-X*X-Y*Y) 10,10,20
20    DIPØLE = (Y+B)/(X+A)
      RETURN
      END
```

Algorithm 442

Normal Deviate [S14]

G.W. Hill and A.W. Davis [Recd. 20 Jan. 1971 and 2 Aug. 1971]
C.S.I.R.O. Division of Mathematical Statistics, Glen Osmond, Sth. Australia

Key Words and Phrases: normal distribution inverse, probit transform, Taylor series approximation
CR Categories: 5.12, 5.5
Language: Algol

Description

This procedure evaluates the inverse of the cumulative normal distribution, i.e. the normal deviate $u(p)$, corresponding to the probability level p, where

$$p = P(u) = \int_{-\infty}^{u} \phi(t)\, dt, \qquad \phi(t) = \frac{1}{(2\pi)^{\frac{1}{2}}} exp(-t^2/2).$$

An initial approximation to $u(p)$, such as $x(p)$, may be improved by using an expansion of $u(z)$, defined as the inverse of

$$z = p - P(x) = \int_x^u \phi(t)\, dt.$$

$u(z)$ may be developed in a Taylor series about $z=0$, where $u(0)=x$, see ref. [1],

$$u_n = x + \sum_{r=1}^{n} c_r(x) \left(\frac{z}{\phi(x)} \right)^r \Big/ r!,$$

and

$$c_1(x)=1,\ c_2(x)=x,\ c_3(x)=2x^2+1,\ c_4(x)=6x^3+7x, \\ c_{r+1}(x)=(rx+d/dx)c_r(x).$$

An error $\epsilon(x)$ in the initial approximation, $u_0=x(p)$, entails an error $\epsilon_n(x)$ in u_n of the order of $\epsilon^{n+1}c_{n+1}(x)/(n+1)!$ In order to minimize the maximum relative error $R_n = max\,|\epsilon_n/u_n|$ in the result obtained from n terms of the Taylor series, several sets of coefficients in an initial rational approximation styled after Hastings [2]

$$x(p) = s - \frac{a + bs + cs^2}{d + es + fs^2 + s^3}, \qquad s = (-2ln(p))^{\frac{1}{2}}, \qquad 0 < p < 0.5,$$

have been obtained such that $| [\epsilon(x)]^{n+1}c_{n+1}(x)/x |$ is minimax for $|x|<40$. For odd n the minimized expression is an even function of ϵ and x, so that the relative error level may be halved when n is odd by adding $\frac{1}{2}xR_n$. The resulting precision is shown below as S_n, i.e.

$$10^{-S_n} = max \left| \frac{error(result)}{result} \right|$$

	a	b	c	d	e	f	R_n	S_n
u_1	1271.059	450.636	7.45551	500.756	750.365	110.4212	$0.622_{10}-7$	7.50
u_2	1484.397	494.327	7.61067	589.557	855.441	119.4733	$0.644_{10}-10$	10.19
u_3	1251.789	444.751	7.51005	493.187	739.156	109.3967	$0.743_{10}-13$	13.43
u_4	1637.720	494.877	7.47395	659.935	908.401	117.9407	$0.111_{10}-15$	15.95
u_5	1488.369	460.200	7.38458	598.957	831.379	110.7527	$0.940_{10}-19$	19.37
u_6	1269.225	448.718	7.49755	499.171	749.275	110.0194	$0.759_{10}-21$	21.12
u_7	1266.846	448.047	7.49101	498.003	748.189	109.8371	$0.166_{10}-23$	24.07

According to the precision required, one set of coefficients and the corresponding labeled statement, selected from the following list, should be incorporated in the procedure body as illustrated for the case of u_7.

```
u1: normdev := z + x×1.0000000311
u2: normdev := (x×z×0.5 + 1.0)×z + x
u3: normdev := (((s + 0.5)×z/3.0 + x×0.5)×z + 1.0)×z + x
               + 0.3713₁₀−13
u4: normdev := (((((s×0.75 + 0.875)×z + x)×x + 0.5)×z/3.0
               + x×0.5)×z + 1.0)×z + x
u5: normdev := ((((((s×0.6 + 1.15)×s + 0.175)×z
               + (s×0.75 + 0.875)×x)×z + s + 0.5)×z/3.0
               + x×0.5)×z + 1.0)×z + x
               + 0.42₁₀−19×x
u6: normdev := (((((((120×s + 326)×s + 127)×x×z/6
               + (24×s + 46)×s + 7)×z/40 + (0.75×s
               + 0.875)×x)×z
               + s + 0.5)×z/3.0 + x×0.5)×z + 1.0)×z + x
u7: normdev := (((((((((720 ×s + 2556)×s + 1740)×s
               + 127)×z/7
               + ((120×s + 326)×s + 127)×x)×z/6
               + (24×s + 46)×s + 7)×z/40 + (0.75×s
               + 0.875)×x)×z
               + s + 0.5)×z/3.0 + x×0.5)×z + 1.0)×z + x
               + 0.832₁₀−24×x
```

Coefficients in a similar Taylor series in powers of $ln(P(x)/p)$, used in AS Algorithm *24* [*3*], require more computation than the $c_n(x)$ in these approximations.

The real procedure supplied by the user for *normal(x,y)* should return the value of the tail area to the left of x and, via the second parameter, y, should return the value of $\phi(x)$, which is often available in the process of computing the tail area. A procedure based on Algorithm 304 [4] is recommended since other algorithms such as Algorithm 209 [5] and *CDFN* [6] lose precision as p approaches their error levels (about 10^{-7}, 10^{-10} respectively), whereas Algorithm 304 maintains precision until calculations involving $\phi(x)$ exceed the capacity of floating point representation. The similar CJ Algorithm 39 [7] matches the precision of u_2 and may be readily modified to return also the value of $\phi(x)$.

The user-supplied real procedure *extreme* (p) should cater for the cases $p=0$, $p=1$, by returning suitable extreme values dependent on the floating point representation for the processor used, e.g. $extreme(0) = -37$ where binary exponents are ten bits, since $\phi(-37)$ is approximately $2^{-2^{10}}$ and $extreme(1) = +7$ for 36-bit precision, since $P(x>7)$ is approximately $1-2^{-36}$. If p lies outside (0,1) the procedure should provide a diagnostic warning and may terminate or return an extreme value such as +37 as an indication of error to the calling program.

Precision may be extended by using the D decimal digit result from one application of *normal* and the n-term Taylor series as an initial approximation for a second application, thus increasing precision to at least $(n+1)(D\text{-}log_{10}(x^2+1))$ decimal digits (as noted by the referee) or at most the precision of *normal*, e.g. $u_1(u_1)$ as in *CDFN1* [6] would have a relative error $O(10^{-14}(x^2+1))$, if not limited by the use of the lower precision *CDFN* for *normal*. For double precision calculations the more elaborate higher order terms of the Taylor series may be evaluated using single precision operations, enabling achievement of extended precision with relatively little increase in processor time. Calculations to 25 decimal digit precision and independently calculated check values to 18 significant digits [8] confirmed achievement of at least 10 significant decimal digits for Algorithm AS 24 and S_n significant digits for this procedure, except for limitations of representation of p near 1.0.

References

1. Hill, G.W., and Davis, A.W. Generalized asymptotic expansions of Cornish-Fisher type. *Ann. Math. Statist. 39*, 4 (Aug. 1968), 1264–1273.
2. Hastings, C. Jr. *Approximations for Digital Computers.* Princeton U. Press, Princeton, New Jersey, 1955, pp. 191–192.
3. Cunningham, S.W. Algorithm AS 24, From normal integral to deviate. *J.R. Statist. Soc. C. 18*, 3 (1969), 290–293.
4. Hill, I.D., and Joyce, S.A. Algorithm 304, Normal. *Comm. ACM 10*, 6 (June 1967), 374.
5. Ibbetson, D. Algorithm 209, Gauss. *Comm. ACM 6*, 10 (Oct. 1963), 616.
6. Milton, R.C., and Hotchkiss, R. Computer evaluation of the normal and inverse normal distribution functions. *Technometrics 11*, 4 (Nov. 1969), 817–822.
7. Adams, A.G. Algorithm 39. Areas under the normal curve. *Comp. J. 12*, 2 (May 1969), 197–198.
8. Strecok, A.J. The inverse of the error function. *Math. Comp. 22*, 101 (Jan. 1968), 144–158.

Algorithm

```
real procedure normdev(p, normal, extreme);
  value p; real p; real procedure normal, extreme;
  comment Input parameter p is the cumulative normal probability
    defined by
      p = ∫_{-∞}^{u} φ(t) dt,   φ(t) = (1/(2π)^½) exp(−t²/2),
    normal (x,y) is a procedure for evaluating the above integral
    for u=x and which returns y = φ(x), extreme(p) is a procedure
    designed to handle extreme values of p. On completion of ex-
    ecution of this procedure normdev is an approximation for u;
begin
  real s, x, z;
  x := if p > 0.5 then 1.0 − p else p;
  if x < 0.0 then normdev := extreme (p)
  else
  begin
    comment Initial rational approximation;
    s := sqrt(−2.0 × ln(x));
    x := ((−7.49101 × s − 448.047) × s − 1266.846)/
      (((s + 109.8371) × s + 748.189) × s + 498.003) + s;
    if p < 0.5 then x := −x;
    z := p − normal(x,s); z := z/s; s := x ↑ 2;
u7:
    normdev := (((((((((720 × s + 2556) × s + 1740) × s
      + 127) × z/7
      + ((120 × s + 326) × s + 127) × x) × z/6
      + (24 × s + 46) × s + 7) × z/40 + (0.75 × s + 0.875)
      × x) × z
      + s + 0.5) × z/3.0 + x × 0.5) × z + 1.0) × z + x
      + 0.832₁₀−24 × x
  end seven term Taylor series for 24 decimal precision
end normal deviate
```