RESEARCH ARTICLE | MARCH 01 1990

Recursive Stratified Sampling for Multidimensional Monte Carlo Integration FREE

William H. Press; Glennys R. Farrar

Comput. Phys. 4, 190–195 (1990)
https://doi.org/10.1063/1.4822899

NUMERICAL RECIPES

Recursive Stratified Sampling For Multidimensional Monte Carlo Integration

William H. Press and Glennys R. Farrar

In this column, we return to the topic of multidimensional Monte Carlo integration. Given a function $f(\mathbf{x})$ and a d-dimensional region of volume V, the problem at its simplest is to find a good statistical estimator, denoted $\langle \tilde{f} \rangle$, of the mean value of f, denoted $\langle f \rangle$, in terms of some number N of randomly sampled function values $f_i \equiv f(\mathbf{x}_i)$, with $\mathbf{x}_i \in V$. (Once we have done this, the integral of f over V is, of course, $\langle f \rangle V$.)

The simplest Monte Carlo estimator is the mean of the N randomly sampled function values,

$$\langle \tilde{f} \rangle = \frac{1}{N} \sum_i f_i. \tag{1}$$

The variance of this estimator, $\mathrm{Var}(\langle \tilde{f} \rangle)$, in the simplest case an indicator of the square of the error of the Monte Carlo integration, is asymptotically related to the variance of the function, $\mathrm{Var}(f) \equiv \langle f^2 \rangle - \langle f \rangle^2$, by the familiar relation

$$\mathrm{Var}(\langle \tilde{f} \rangle) = \mathrm{Var}(f)/N. \tag{2}$$

In other words, the expected error decreases with increased sample size as $N^{-1/2}$.

There are a number of techniques that improve on the error implied by Eq. (2). One such is "importance sampling," where the function f is divided by some known, positive function g, and the volume V is sampled not uniformly but rather with probability density proportional to g. To the extent that f/g can be made approximately constant, then $\mathrm{Var}(f/g)$ [which replaces $\mathrm{Var}(f)$ in Eq. (2)] is correspondingly decreased. It can be shown, in fact, that the optimal choice for g is $g \propto |f|$, valid even if f takes on both signs. Importance sampling is discussed in the *Numerical Recipes* books[1–3] and standard references.[4,5]

A technique that improves on Eq. (2) in quite a different way is the use of "quasirandom numbers," as was discussed previously in this column.[6] With quasirandom numbers, one can improve the *exponent* of N in Eq. (2). Importance sampling and quasirandom numbers are not mutually exclusive—both techniques can be applied simultaneously. In this column we discuss a third technique, which we call *recursive stratified sampling* (RSS), that can be used *in addition* to either or both of the techniques already discussed.

The idea of "stratified sampling," on which RSS is based, is a standard one in the literature[4] and easily illustrated: Suppose we divide the volume V into two equal, disjoint subvolumes, denoted a and b, and sample $N/2$ points in each subvolume. Then another estimator for $\langle f \rangle$, different from Eq. (1) and denoted $\langle \tilde{f} \rangle'$, is

$$\langle \tilde{f} \rangle' \equiv \tfrac{1}{2}(\langle \tilde{f} \rangle_a + \langle \tilde{f} \rangle_b), \tag{3}$$

in other words, the mean of the sample averages in the two half-regions. The variance of estimator (3) is given by

$$\begin{aligned} \mathrm{Var}(\langle \tilde{f} \rangle') &= \tfrac{1}{4}[\mathrm{Var}(\langle \tilde{f} \rangle_a) + \mathrm{Var}(\langle \tilde{f} \rangle_b)] \\ &= \frac{1}{4}\left(\frac{\mathrm{Var}_a(f)}{N/2} + \frac{\mathrm{Var}_b(f)}{N/2}\right) \\ &= \frac{1}{2N}[\mathrm{Var}_a(f) + \mathrm{Var}_b(f)]. \end{aligned} \tag{4}$$

Here, $\mathrm{Var}_a(f)$ denotes the variance of f in subregion a, that is, $\langle f^2 \rangle_a - \langle f \rangle_a^2$, and correspondingly for b.

From the definitions already given, it is not difficult to prove the relation

$$\begin{aligned} \mathrm{Var}(f) &= \tfrac{1}{2}[\mathrm{Var}_a(f) + \mathrm{Var}_b(f)] \\ &\quad + \tfrac{1}{4}(\langle f \rangle_a - \langle f \rangle_b)^2. \end{aligned} \tag{5}$$

(In physics, this formula for combining second moments is the "parallel axis theorem.") Comparing Eqs. (2), (4), and (5), one sees that the stratified (into two subvolumes) sampling gives a variance that is never larger than the simple Monte Carlo case—and smaller whenever the means of the stratified samples, $\langle f \rangle_a$ and $\langle f \rangle_b$, are different.

We have not yet exploited the possibility of sampling the two subvolumes with *different numbers* of points, say, N_a in subregion a and $N_b \equiv N - N_a$ in subregion b. Let us do so now. Then the variance of the estimator is

$$\mathrm{Var}(\langle \tilde{f} \rangle') = \frac{1}{4}\left(\frac{\mathrm{Var}_a(f)}{N_a} + \frac{\mathrm{Var}_b(f)}{N - N_a}\right), \tag{6}$$

which is minimized (one can easily verify) when

$$N_a/N = \sigma_a/(\sigma_a + \sigma_b). \tag{7}$$

Here, we have adopted the shorthand notation $\sigma_a \equiv \sqrt{\mathrm{Var}_a(f)}$, and correspondingly for b. If N_a satisfies Eq. (7), then Eq. (6) reduces to

$$\mathrm{Var}(\langle \tilde{f} \rangle') = (\sigma_a + \sigma_b)^2/4N. \tag{8}$$

Equation (8) reduces to Eq. (2) if $\mathrm{Var}(f) = \mathrm{Var}_a(f) = \mathrm{Var}_b(f)$, in which case stratifying the sample makes no difference.

William H. Press is professor of astronomy and physics at Harvard University. Glennys R. Farrar is professor of physics at Rutgers University.

A standard way to generalize the above result is to consider the volume V divided into more than two subvolumes. One can readily obtain the result that the optimal allocation of sample points among the regions is to have the number of points in each region j proportional to σ_j, that is, the square root of the variance of the function f in that subregion. In spaces of high dimensionality (say, $d \gtrsim 4$) this is not in practice very useful, however. Dividing a volume into K segments along each dimension implies K^d subvolumes, typically much too large a number when one contemplates estimating all the corresponding σ_j's. Instead, we will describe an adaptive (more precisely, recursive) scheme that subdivides the volume V in a more manageable fashion, only where that subdivision is most needed. There are other ways of avoiding the K^d "explosion," however; this is a good point, therefore, to digress briefly on the subject of another adaptive Monte Carlo method.

The VEGAS algorithm[7,8] is widely used for multidimensional integrals that occur in elementary particle physics. VEGAS is based on importance sampling, not stratified sampling. Its basic technique is to construct, adaptively, a multidimensional weight function g (see above) that is *separable*,

$$g(x,y,z,\ldots) = g_x(x)g_y(y)g_z(z)\ldots . \tag{9}$$

Such a function avoids the K^d explosion in two ways: (i) It can be stored in the computer as d separate one-dimensional functions, each defined by K tabulated values, say—so that $K \times d$ replaces K^d; (ii) It can be sampled as a probability density by consecutively sampling the d one-dimensional functions to obtain coordinate vector components $(x,y,z,\ldots)$.

The optimal separable weight function can be shown to be[7]

$$g_z(x) \propto \left(\int dy \int dz \cdots \frac{f^2(x,y,z,\ldots)}{g_y(y)g_z(z)\cdots} \right)^{1/2} \tag{10}$$

(and correspondingly for $y,z,\ldots$). Notice that this reduces to $g \propto |f|$ in one dimension. Equation (10) immediately suggests VEGAS' adaptive strategy: Given a set of g functions (initially all constant, say), one samples the function f, accumulating not only the overall estimator of the integral, but also the Kd estimators (K subdivisions of the independent variable in each of d dimensions) of the right-hand side of Eq. (10). These then determine improved g functions for the next iteration.

When the integrand f is concentrated in one, or at most a few, regions in d space, then the weight function g's quickly become large at coordinate values that are the projections of these regions onto the coordinate axes. The accuracy of the Monte Carlo integration is then enormously enhanced over what simple Monte Carlo would give.

The weakness of VEGAS is the obvious one: To the extent that the projection of the function f onto individual coordinate directions is uniform, VEGAS gives no concentration of sample points in those dimensions. The worst case for VEGAS, e.g., is an integrand that is concentrated close to a body diagonal line, e.g., one from $(0,0,0,\ldots)$ to $(1,1,1,\ldots)$. Since this geometry is completely nonseparable, VEGAS can give no advantage at all. More generally, VEGAS will do badly when the integrand is concentrated along one-dimensional (or higher) curved trajectories (or hypersurfaces), unless these happen to be oriented close to the coordinate directions.

With RSS, we pursue a different approach, starting with Eqs. (3) and (6). Suppose that we have a quota of N evaluations of the function f and want to evaluate $\langle \tilde{f} \rangle'$ in the rectangular parallelepiped region $R = (\mathbf{x}_a, \mathbf{x}_b)$. (We denote such a region by the two coordinate vectors of its diagonally opposite corners.) First, we allocate a fraction p of N toward exploring the variance of f in R: We sample pN function values uniformly in R and accumulate the sums that will give the d different pairs of variances corresponding to the d different coordinate directions along which R can be bisected. In other words, in pN samples, we estimate $\mathrm{Var}(f)$ in each of the regions resulting from a possible bisection of R,

$$R_{ai} \equiv \left[\mathbf{x}_a,\ \mathbf{x}_b - \tfrac{1}{2}\mathbf{e}_i \cdot (\mathbf{x}_b - \mathbf{x}_a)\mathbf{e}_i \right],$$

$$R_{bi} \equiv \left[\mathbf{x}_a + \tfrac{1}{2}\mathbf{e}_i \cdot (\mathbf{x}_b - \mathbf{x}_a)\mathbf{e}_i,\ \mathbf{x}_b \right], \qquad i = 1,2,\ldots d. \tag{11}$$

Here, $\mathbf{e}_i$ is the unit vector in the ith coordinate direction.

Second, we inspect the variances to find the most favorable dimension i to bisect. By Eq. (8), we could, for example, choose that i for which the sum of the square roots of the variance estimators in regions R_{ai} and R_{bi} is minimized. (Actually, as we will explain, we do something slightly different.)

Third, we allocate the remaining $(1-p)N$ function evaluations between the regions R_{ai} and R_{bi}. If we used Eq. (8) to choose i, we should do this allocation according to Eq. (7).

We now have two parallelepipeds each with its own allocation of function evaluations for estimating the mean of f. Our RSS algorithm now shows itself to be *recursive*: To evaluate the mean in each region, we go back to the sentence beginning "First,..." in the paragraph above Eq. (11). (Of course, when the allocation of points to a region falls below some number, we resort to simple Monte Carlo rather than continue with the recursion.)

Finally, we combine the means, and also estimate variances of the two subvolumes, using Eq. (3) and the first line of Eq. (4).

This completes the RSS algorithm in its simplest form. Before we describe some additional tricks under the general rubric of "implementation details," we need to return briefly to Eqs. (6)–(8) and derive the equations that we actually use instead of these. The right-hand side of Eq. (6) applies the familiar scaling law of Eq. (2) twice, once to a and again to b. This would be correct if the estimates $\langle \tilde{f} \rangle_a$ and $\langle \tilde{f} \rangle_b$ were each made by simple Monte Carlo, with uniformly random sample points.

Box 1.

```
      SUBROUTINE miser(func,region,ndim,npts,dith,ave,var)
      INTEGER ndim,npts,MNPT,MNBS,MAXD,NSTACK,NSTF
      REAL func,region(2*ndim),dith,ave,var,TINY,BIG,PFAC
      PARAMETER (MNPT=15,MNBS=4*MNPT,MAXD=10,TINY=1.e-30,BIG=1.e30,
     *     NSTACK=1000,NSTF=9,PFAC=0.1)
C     USES func,ranpt
        Monte Carlo samples a user-supplied ndim-dimensional function func in a rectangular volume
        specified by region, a 2×ndim vector consisting of ndim "lower-left" coordinates of the
        region followed by ndim "upper-right" coordinates. The function is sampled a total of npts
        times, at locations determined by the method of recursive stratified sampling. The mean value
        of the function in the region is returned as ave; an estimate of the statistical variance of ave
        (square of standard deviation) is returned as var. The input parameter dith should normally
        be set to zero, but can be set to (e.g.) 0.1 if func's active region falls on the boundary of a
        power-of-two subdivision of region.
        Parameters: PFAC is the fraction of remaining function evaluations used at each stage to
        explore the variance of func. At least MNPT function evaluations are performed in any terminal
        sub-region; a sub-region is further bisected only if at least MNBS function evaluations are
        available. MAXD is the largest value of ndim. NSTF is the size of the "stack frame", and
        NSTACK is the total size of the stack.
      INTEGER iran,j,jb,jstack,n,naddr,np,npre,nptl,nptr,nptt
      REAL avel,fracl,fval,rgl,rgm,rgr,s,sigl,siglb,sigr,sigrb,sum,
     *     sumb,summ,summ2,varl,fmaxl(MAXD),fmaxr(MAXD),fminl(MAXD),
     *     fminr(MAXD),pt(MAXD),rmid(MAXD),stack(NSTACK),stf(NSTF)
      EQUIVALENCE (stf(1),avel),(stf(2),varl),(stf(3),jb),
     *     (stf(4),nptr),(stf(5),naddr),(stf(6),rgl),(stf(7),rgm),
     *     (stf(8),rgr),(stf(9),fracl)
      SAVE iran
      DATA iran /0/
      jstack=0
      nptt=npts
1     continue
      if (nptt.lt.MNBS) then          Too few points to bisect; do straight Monte Carlo.
        np=abs(nptt)
        summ=0.
        summ2=0.
        do 11 n=1,np
          call ranpt(pt,region,ndim)
          fval=func(pt)
          summ=summ+fval
          summ2=summ2+fval**2
        enddo 11
        ave=summ/np
        var=max(TINY,(summ2-summ**2/np)/np**2)
      else                            Do the preliminary (uniform) sampling.
        npre=max(int(nptt*PFAC),MNPT)
        do 12 j=1,ndim                Initialize the left and right bounds for each dimension.
          iran=mod(iran*2661+36979,175000)
          s=sign(dith,float(iran-87500))
          rmid(j)=(0.5+s)*region(j)+(0.5-s)*region(j+ndim)
          fminl(j)=BIG
          fminr(j)=BIG
          fmaxl(j)=-BIG
          fmaxr(j)=-BIG
        enddo 12
        do 14 n=1,npre                Loop over the points in the sample.
          call ranpt(pt,region,ndim)
          fval=func(pt)
          do 13 j=1,ndim              Find the left and right bounds for each dimension.
            if(pt(j).le.rmid(j))then
              fminl(j)=min(fminl(j),fval)
              fmaxl(j)=max(fmaxl(j),fval)
            else
              fminr(j)=min(fminr(j),fval)
              fmaxr(j)=max(fmaxr(j),fval)
            endif
          enddo 13
        enddo 14
        sumb=BIG                      Choose which dimension jb to bisect.
        jb=0
        siglb=1.
        sigrb=1.
        do 15 j=1,ndim
          if(fmaxl(j).gt.fminl(j).and.fmaxr(j).gt.fminr(j))then
            sigl=max(TINY,(fmaxl(j)-fminl(j))**0.6666)
            sigr=max(TINY,(fmaxr(j)-fminr(j))**0.6666)
            sum=sigl+sigr             Equation 14, see text.
            if (sum.le.sumb) then
              sumb=sum
              jb=j
              siglb=sigl
              sigrb=sigr
            endif
          endif
        enddo 15
        if (jb.eq.0) jb=1+(ndim*iran)/175000    MNPT may be too small.
        rgl=region(jb)                Apportion the remaining points between left and right.
        rgm=rmid(jb)
        rgr=region(jb+ndim)
        fracl=abs((rgm-rgl)/(rgr-rgl))
        nptl=MNPT+(nptt-npre-2*MNPT)
     *       *fracl*siglb/(fracl*siglb+(1.-fracl)*sigrb)      Equation 13.
        nptr=nptt-npre-nptl
        region(jb+ndim)=rgm           Set region to left.
        naddr=1                       Push the stack.
        do 16 j=1,NSTF
          stack(jstack+j)=stf(j)
        enddo 16
        jstack=jstack+NSTF
        nptt=nptl
        goto 1                        Dispatch recursive call; will return back here eventually.
10      continue
        avel=ave                      Save left estimates on stack variable.
        varl=var
        region(jb)=rgm                Set region to right.
        region(jb+ndim)=rgr
        naddr=2                       Push the stack.
        do 17 j=1,NSTF
          stack(jstack+j)=stf(j)
        enddo 17
        jstack=jstack+NSTF
        nptt=nptr
        goto 1                        Dispatch recursive call; will return back here eventually.
20      continue
        region(jb)=rgl                Restore region to original value (so that we don't need to
        ave=fracl*avel+(1.-fracl)*ave   include it on the stack).
        var=fracl**2*varl+(1.-fracl)**2*var   Combine left and right regions by Equa-
      endif                           tion 4 (1st line).
      if (jstack.ne.0) then           Pop the stack.
        jstack=jstack-NSTF
        do 18 j=1,NSTF
          stf(j)=stack(jstack+j)
        enddo 18
        goto (10,20),naddr
        pause 'Never get here.'
      endif
      return
      END
```

However, the two estimates of the mean are in fact made recursively. Thus there is no reason to expect Eq. (2) to hold. Rather, we might substitute for Eq. (6) the relation,

$$\mathrm{Var}(\langle \tilde{f} \rangle') = \frac{1}{4}\left(\frac{\mathrm{Var}_a(f)}{{N_a}^{\alpha}} + \frac{\mathrm{Var}_b(f)}{(N-N_a)^{\alpha}}\right), \quad (12)$$

where α is an unknown constant $\geqslant 1$ (the case of equality corresponding to simple Monte Carlo). In that case, a short calculation shows that Var $(\langle \tilde{f} \rangle')$ is minimized when

$$\frac{N_a}{N} = \frac{\mathrm{Var}_a(f)^{1/(1+\alpha)}}{\mathrm{Var}_a(f)^{1/(1+\alpha)} + \mathrm{Var}_b(f)^{1/(1+\alpha)}} \quad (13)$$

and that its minimum value is

$$\mathrm{Var}(\langle \tilde{f} \rangle') \propto \left[\mathrm{Var}_a(f)^{1/(1+\alpha)} + \mathrm{Var}_b(f)^{1/(1+\alpha)}\right]^{1+\alpha}. \quad (14)$$

Equations (12)–(14) reduce to Eqs. (6)–(8) when $\alpha = 1$. We have done numerical experiments to find a self-consistent value for α. We find that $\alpha \approx 2$. That is, when Eq. (13) with $\alpha = 2$ is used recursively to allocate sample opportunities, the observed variance of the RSS algorithm goes approximately as N^{-2}, while any other value of α in Eq. (13) appears to give a poorer fall-off. The sensitivity to α is, however, not very great; we do not know if $\alpha = 2$ is an analytically justifiable result, or only a useful heuristic.

Box 2.

```
      SUBROUTINE ranpt(pt,region,n)
      INTEGER n
      REAL pt(n),region(2*n),ran5
      USES ran5
        Returns a uniformly random point pt in an n-dimensional rectangular region. Used by
        miser; calls ran5 for uniform deviates.
      INTEGER idum,j
      SAVE idum
      DATA idum /-7/
      do 11 j=1,n
          pt(j)=region(j)+(region(j+n)-region(j))*ran5(idum)
      enddo 11
      return
      END
```

Box 3.

```
      FUNCTION ran5(idum)
      INTEGER idum,IA,IM,IQ,IR,NTAB
      REAL ran5,AM,ATAB
      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
     *     NTAB=32,ATAB=NTAB-1)
        Park and Miller's "Minimal Standard" random number generator (Comm. ACM, 31, 1192,
        1988) with Bays-Durham shuffle. Call with idum negative to initialize. Be sure to preserve
        idum between calls.
      INTEGER j,k
      REAL v(NTAB),y
      SAVE v,y
      DATA v /NTAB*0./, y /0.5/
      if (idum.le.0) then
          idum=max(-idum,1)
          do 12 j=NTAB,1,-1
              k=idum/IQ
              idum=IA*(idum-k*IQ)-IR*k
              if (idum.lt.0) idum=idum+IM
              v(j)=AM*idum
          enddo 12
          y=v(1)
      endif
1     continue
          k=idum/IQ
          idum=IA*(idum-k*IQ)-IR*k
          if (idum.lt.0) idum=idum+IM
          j=1+int(ATAB*y)
          y=v(j)
          ran5=y
          v(j)=AM*idum
      if (ran5.eq.0..or.ran5.eq.1.)goto 1
      return
      END
```

Turn now to the routine, miser, which implements our RSS method. A bit of FORTRAN wizardry is its implementation of the required recursion. This is done by dimensioning an array stack, and a shorter "stack frame" stf; the latter has components that are equivalenced to variables that need to be preserved during the recursion, including a flag indicating where program control should return. A recursive call then consists of copying the stack frame onto the stack, incrementing the stack pointer jstack, and transferring control. A recursive return analogously pops the stack and transfers control to the saved location. Stack growth in miser is only linear (that is, logarithmic), since at each bifurcation one of the subvolumes can be processed immediately.

The principal difference between miser's implementation and the algorithm as described thus far lies in how the variances on the right-hand side of Eq. (13) are estimated. We find empirically that it is somewhat more robust to use the square of the difference of maximum and minimum sampled function values, instead of the genuine second moment of the samples. This estimator is, of course, increasingly biased with increasing sample size; however, Eq. (13) uses it only to compare two subvolumes (a and b) having approximately equal numbers of samples. The "max minus min" estimator proves its worth when the preliminary sampling yields only a single point, or small number of points, in active regions of the integrand. In many realistic cases, these are indicators of nearby regions of even greater importance, and it is useful to let them attract the greater sampling weight that "max minus min" provides.

A second modification embodied in the code is the introduction of a "dithering parameter," dith, whose nonzero value causes subvolumes to be divided not exactly down the middle, but rather into fractions $0.5 \pm$ dith, with the sign of the $\pm$ randomly chosen by a built-in (toy) random number routine. Normally dith can be set to zero. However, there is a large advantage in taking dith to be nonzero if some special symmetry of the integrand puts the active region exactly at the midpoint of the region, or at the center of some power-of-two submultiple of the region. One wants to avoid the extreme case of the active region being evenly divided into 2^d abutting corners of a d-dimensional space. A typical nonzero value of dith, on those rare occasions when it is useful, might be 0.1. Of course, when the dithering parameter is nonzero, we must take the differing sizes of the subvolumes into account; the code does this through the variable fracl.

One final feature in the code deserves mention. Our RSS algorithm uses a single set of sample points to evaluate Eq. (13) in all d directions. At bottom levels of the recursion, the number of sample points can be quite small. Although rare, it can happen that in one direction all the samples are in one half of the volume; in that case, that direction is ignored as a candidate for bifurcation. Even more rare is the possibility that all of the samples are in one half of the volume in *all* directions. In this case, a random direction is chosen. If this happens too often in your application, then you should increase MNPT (see commented statement in code).

The miser routine calls a short subroutine ranpt to get a random point within a specified d-dimensional region. Box 2 gives code for ranpt that makes consecutive calls to a uniform random number generator and does the obvious scaling. (For portability, Box 3 lists a good standard random number generator.) Actually, we use a different version of ranpt, one that makes consecutive d-dimensional calls to the quasirandom routine SOBSEQ, as listed in a previous column.[6] We find that miser with SOBSEQ can be considerably more accurate than miser with uniform random deviates. Since the use of RSS and the use of quasirandom numbers are completely separable, however, we have not made the code given here dependent on SOBSEQ. A similar remark might be made regarding importance sampling, which could in principle be combined with RSS. (One could in principle combine VEGAS and miser, although the programming would be intricate.)

We have compared miser to VEGAS on several test integrands, all selected to have strongly localized active regions, but with different geometries. Since we expect differences to be most evident in higher-dimensional problems, we did four-dimensional integrals on the following functions in the unit cube $0 < x_i < 1$:

- A narrow Gaussian, with peak inside the domain of integration. This is separable in the sense of Eq. (9).
- $1/\sqrt{x_1 x_2 x_3 x_4}$. This is also separable, with a singularity at one corner of the integration region.
- $1/(x_1 + x_2 + x_3 + x_4 - 2 + i\epsilon)$, with $\epsilon \sim 0.001$. This is nonseparable, with a near-singularity on a hyperplane that intersects the center of the integration region.

As expected, miser clearly dominated VEGAS for the third (nonseparable) case; with 10^5 function evaluations, its errors were 50 times smaller than those of VEGAS. Equally important, the variance reported by miser coincided nicely with the actual variance of miser's results, determined in 100 different runs, whereas the observed variance of VEGAS was generally a factor of 10–150 larger than the square of its reported standard deviation.

VEGAS and miser performed about equally well on the Gaussian; but miser accurately reflected, or slightly overestimated, its variance, while the square of VEGAS's standard deviation was consistently a factor of 10–100 smaller than the true variance of its results.

VEGAS was more accurate than miser on the square-root singularity, by a factor of 2–5. Again, miser's reported variance was in acceptable agreement with the distribution of its actual results whereas VEGAS underreported its variance (although in this case only by a factor of 2–4).

Of course, the space of "all possible integrands" is much too large for us to sample in any meaningful way. Nevertheless, we think that miser should be robust in a variety of circumstances. We welcome user reports.

We conclude with a puzzle that indicates the possibility of additional improvements in the RSS algorithm: The astute reader will have noticed that we make no use of the function values acquired in the preliminary, variance-exploring, sampling of a subvolume, except for deciding how to allocate remaining samples. Since allocated samples are in turn used to probe the variance at the next level of recursion, a consequence is that, in aggregate, relatively few function evaluations are directly used in evaluating the desired mean. That seems odd! It would seem reasonable to use the variance-exploring function values to form a preliminary estimate of a subvolume's mean value, and then to combine statistically that estimate with the (more accurate) estimate returned by the recursion. We have spent considerable effort trying to do this; in all variants tried we find, empirically, that inclusion of the extra function values results in poorer, not better, performance. The problem is not with the estimate of the mean, but with the estimate of the weight that it should have when it is combined. Quite subtle statistical biases seem to be present. We commend this problem to interested readers.

In our next column: Hypergeometric functions by direct path integration. ■

References

1. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes* (Cambridge U. P., New York, 1986).
2. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C* (Cambridge U. P., New York, 1988).
3. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Pascal* (Cambridge U. P., New York, 1989).
4. J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods* (Methuen, London, 1964).
5. M. H. Kalos and P. A. Whitlock, *Monte Carlo Methods* (Wiley, New York, 1986).
6. W. H. Press and S. A. Teukolsky, Comput. Phys. **3**(6), 76 (1989).
7. G. P. Lepage, J. Comput. Phys. **27**, 192 (1978).
8. G. P. Lepage, "VEGAS: An Adaptive Multidimensional Integration Program," Publication CLNS-80/447, Cornell University (1980).