Numerical integration in many dimensions. II

Charles Schwartz
Department of Physics, University of California, Berkeley, California 94720

(Received 13 June 1984; accepted for publication 21 September 1984)

Two new techniques are presented that appear to be useful in obtaining accurate numerical values for the numerical integration of fairly smooth functions in many dimensions. Both methods start with the idea of a mesh containing n points laid out in each of the d dimensions, then seek strategies that use far less than all n^d points in some systematically improved sequence of approximations.

I. EXTRAPOLATION METHOD

Suppose we have some prescription for the numerical integration of a function $f(x)$ of one variable:

$$\sum_{j=1}^{n} w_j f(x_j) = \int_a^b f(x)dx + E(n). \tag{1}$$

A high-accuracy prescription (quadrature rule) is the set of points x_j and weights w_j such that the error $E(n)$ is a small and rapidly decreasing function of n, the number of mesh points used.

Now suppose we want to integrate a function $F(x_1,x_2,...,x_d) = F(\mathbf{x})$ over the d-dimensional cube. The direct product technique would be to use the rule (1) d times:

$$\sum_{j_1=1}^{n_1}\sum_{j_2=1}^{n_2}\cdots\sum_{j_d=1}^{n_d} w_{j_1}w_{j_2}\cdots w_{j_d}F(x_{j_1},x_{j_2},...,x_{j_d})$$

$$\equiv S(n_1,n_2,...,n_d) = S(\mathbf{n}). \tag{2}$$

This computation will require a large amount of effort, since the total number of evaluations involved is

$$N = \prod_{i=1}^{d} n_i. \tag{3}$$

To see the form of the error, apply the relation (1) d times to $F(\mathbf{x})$:

$$S(\mathbf{n}) = \int\int\cdots\int d^d x\, F(\mathbf{x}) + [E_1(n_1) + E_2(n_2) + \cdots + E_d(n_d)] + \text{higher-order terms}, \tag{4}$$

where the higher-order terms would be of the form of products of two or more "small" terms. This is the main result: If the errors are indeed small in each separate dimension, the leading (first-order) error term for the multidimensional computation is *additive* in contribution from each dimension.

Upon this observation we build a simple technique for eliminating the first-order errors. First, compute S for a given set of numbers n_i; then, one at a time, increase the number of mesh points used in a *single* dimension while keeping all the others fixed, and compute

$$D_i \equiv S(n_1, n_2,...,n_i,...,n_d) - S(n_1, n_2,...,n_i',...,n_d), \quad i = 1, d. \tag{5}$$

Then, from (4), we have

$$D_i \cong E_i(n_i) - E_i(n_i'); \tag{6}$$

and, if n_i' is substantially larger than n_i, we may take

$$D_i \approx E_i(n_i), \tag{7}$$

because each $E(n)$ is assumed to decrease very rapidly as n increases. Thus we computationally determine the first-order error terms and we subtract these terms out from the original computation to get the improved approximation for the integral I of $F(\mathbf{x})$:

$$I \approx S(\mathbf{n}) - \sum_{i=1}^{d} D_i. \tag{8}$$

The saving in computer time by this technique may be considerable: If N_0 is the number of evaluations needed to compute the original $S(\mathbf{n})$, and if we take each $n_i' = 2n_i$, then the additional computing effort for the result (8) is $2dN_0$; this may be compared to $2^d N_0$ which is the amount of effort needed if one doubled all the n_i at once.

This result is an extension of the basic idea in Richardson extrapolation, except that we do not assert a known form for the error function $E(n)$ but only rely upon it being rapidly decreasing.

For numerical examples I took two six-dimensional integrals of complicated form from the book by Davis and Rabinowitz[1]:

$$F_1 = x_1x_2x_3x_4x_5x_6[\log(x_1x_2x_3/x_4x_5x_6)]^2,$$
integrated over the cube $(0,1)^6$, (9a)

$$F_2 = \tfrac{1}{64}\cos(3x_1x_2x_3x_4x_5(1-x_6) + \tfrac{1}{2})$$
integrated over the cube $(-1,1)^6$. (9b)

The points x_j and weights w_j used were those tabulated for Gauss–Legendre numerical quadrature.

Computed results are displayed in Table I. The column headed "Mesh" gives the set of numbers n_i used for the original $S(\mathbf{n})$ (2^6, 3^6, etc.) in each block, followed by the incremental sets ($n^{d-1}n'$) used. The column headed "Number" counts the number of function evaluations needed at each stage of the computation. (In the actual work these numbers were much reduced because of the permutation symmetry of the integrands, but that is not a general feature of the present method.) The columns headed "Error" give the fractional

 0022-2488/85/050955-03$02.50

TABLE I. Numerical results for the six-dimensional integrals (9a) and (9b) using Gauss–Legendre quadrature rules plus the extrapolation technique (8).

Mesh	Error $-F_1$	Number	Error $-F_2$
2^6	0.16	64	0.002 9
2^54	0.028	768	0.002 1
2^56	0.0085	1 152	
2^58	0.0040	1 536	
2^510	0.0024	1 920	
3^6	0.060	729	0.000 27
3^56	0.0076	8 736	0.000 064
3^58	0.0031	11 664	
3^510	0.0015	14 580	
4^6	0.028	4 096	0.000 014
4^56	0.0077	36 864	0.000 000 81
4^58	0.0030	49 152	0.000 000 83
4^510	0.0014	61 440	
5^6	0.014	15 625	0.000 000 56
5^58		25 000	0.000 000 006 9
6^6	0.0076[a]	46 656	0.000 000 01[a]
8^6	0.0031[a]	262 144	

[a] From R. Cranley and T. N. L. Patterson, Numer. Math. **16**, 70 (1970).

error in the numerical value of the integral (for the functions F_1 and F_2) computed.

Looking first at the results for the function F_1, we see that overall the error is not very small and decreases rather slowly: for example, look only at the sequence n^6. This is doubtless due to the logarithmic singularity in the integrand, something which the chosen quadrature rule is ill prepared to accommodate. Yet, given that overall difficulty, the present scheme is seen to be very successful at getting higher accuracy with fewer number of mesh points used: compare the accuracy at 2^510 (1,920 + 64 mesh points) with that at 8^6 (262,144 mesh points.) There is a cost saving here of two orders of magnitude for the same result.

When we turn to the results for F_2 things are different. The overall accuracy is better and the convergence more rapid. This may be attributed to the analytic character of the function F_2. The improvements gained by the present extrapolation technique start out as nil (in the topmost block) but then increase rapidly, reaching almost two orders of accuracy improvement (in the fourth block) at a cost of less than twice the starting number of mesh points.

I do not have a general theory to predict when this technique will work well or how best to implement it strategically. It does appear to be quite promising, however, as a technique which one can readily experiment with, using systematic increases in the numbers n to show whether the convergence seems to be good or poor.

II. FACTORIZATION METHOD

Suppose the function $F(\mathbf{x})$ were given as a product of factors, each involving only a single coordinate,

$$f_1(x_1)f_2(x_2)\cdots f_d(x_d); \tag{10}$$

then the d-dimensional integral of F would be simply the product of d one-dimensional integrals, each one of which could be evaluated by some numerical quadrature rule such as (1). The total cost would be proportional to nd rather than the much larger number n^d.

Suppose that $F(\mathbf{x})$ may be well approximated by a factorized form (10) but the individual functions $f_i(x_i)$ are not known. Then one may construct these functions as follows. Choose some reference point (node) $\mathbf{y} = (y_1, y_2, \ldots, y_d)$, such that $F(\mathbf{y})$ is nonzero. Now tabulate the values of F walking out from this node along each one of the coordinate axes:

$$f_i(x_j) = F(y_1, y_2, \ldots, y_{i-1}, x_j, y_{i+1}, \ldots, y_d)/F(\mathbf{y}), \quad j = 1, n, \tag{11}$$

where we have chosen a normalization for the factor functions f_i such that they are equal to 1 at the node, and the points x_j would be chosen to fit the quadrature rule (1) being used. We have thus constructed the aproximation

$$F(\mathbf{x}) \approx G(\mathbf{x}) = F(\mathbf{y}) \prod_{i=1}^{d} f_i(x_i), \tag{12}$$

and the integration follows easily.

Now, to develop a generally useful method, we need to invent a sequence of approximations, like (12), such that we may approach closer and closer to the given function F. From the discussion above it is clear that we have the freedom of choice of the node point $\mathbf{y}$ from which the construction (11) follows.

A first strategy is to take a sequence of nodes $\mathbf{y}_k$, $k = 1, 2, 3, \ldots$, and then construct a sequence of product functions $G_k(\mathbf{x})$, defined by (11) and (12), where G_1 is built from the original function F, G_2 is built from the residual function $F - G_1$, G_3 is built from $F - G_1 - G_2$, etc. This procedure was tried on the two six-dimensional integrals (9a) and (9b); the results were very poor. Probably what is happening is this: At the k th stage one is fitting exactly at the point $\mathbf{y}_k$ and on the lines passing through this point but at the same time one is messing up the fit achieved at the previous node points and their lines. Thus the error can just bounce around from one region to another without being reduced.

A second strategy involved constructing a set of approximations $G_k(\mathbf{x})$, each constructed to fit the original function $F(\mathbf{x})$ at the point $\mathbf{y}_k$, independent of the others

$$G_k(\mathbf{x}) = F(\mathbf{y}_k) \prod_{i=1}^{d} f_i^k(x_i), \quad k = 1,2,\ldots. \tag{13}$$

Then take a linear combination of these G_k to minimize the expression

$$\sum_k \Big[F(\mathbf{y}_k) - \sum_l C_l G_l(\mathbf{y}_k) \Big]^2. \tag{14}$$

This was also tried on the same two functions (9a) and (9b) for five points; and the results were even worse than with the first strategy.

A third strategy involved a more complicated "cluster decomposition":

$$F(\mathbf{x}) = F(\mathbf{y}) \prod_i f_i(x_i) + \sum_{i_1 < i_2} H^{(2)}_{i_1,i_2}(x_{i_1}, x_{i_2}) \times \prod_{i \neq i_1, i_2} h_i^{(2)}(x_i) + \sum_{i_1 < i_2 < i_3} H^{(3)}_{i_1,i_2,i_3}(x_{i_1}, x_{i_2}, x_{i_3}) \times \prod_{i \neq i_1, i_2, i_3} h_i^{(3)}(x_i) + \cdots. \tag{15}$$

Here only a single node point y is used; the functions $H^{(2)}$, $H^{(3)}$, etc., span larger-dimensional subspaces and are defined to vanish when any of their arguments are on the lines passing through y. This method was tried, through third order, on the same two functions (9a) and (9b) and the results were unsatisfactory once again.

A fourth strategy works the other way: rather than building up correlations between the coordinates from the uncorrelated product (10), we start by taking the full d-dimensional space and decomposing it into a product of two subspaces

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2), \tag{16}$$

where d_1 (the dimension of $\mathbf{x}_1$) and d_2 (the dimension of $\mathbf{x}_2$) add up to d. The original function F is represented by

$$F(\mathbf{x}) = F(\mathbf{x}_1, \mathbf{x}_2) = \sum_k G_1^k(\mathbf{x}_1) G_2^k(\mathbf{x}_2). \tag{17}$$

This arrangement has a special property, which was first noticed to be true in the first strategy above only for the case $d = 2$. There is a freedom of redefinition of the functions G which leaves F unchanged:

$$G_2^k \to G_2^k + AG_2^{k'}, \; G_1^{k'} \to G_1^{k'} - AG_1^k, \tag{18}$$

for any number A. With this, one can choose a series of node points

$$\mathbf{y}^k = (\mathbf{y}_1^k, \mathbf{y}_2^k),$$

and require

$$G_1^{k'}(\mathbf{y}_1^k) = G_2^{k'}(\mathbf{y}_2^k) = 0, \quad \text{for all } k' > k. \tag{19}$$

This means that we can carry out the sequential fitting described as the "first strategy" to evaluate the functions G^k [Eq. (17)]. The new advantage, from (19), is the fact that fitting at the k th node $\mathbf{y}^k$ will *not* disturb the previous fittings obtained at other nodes. The price paid for this advantage is that each G function must be evaluated at a large number of points. Still, the total number of evaluations, $n^{d_1} + n^{d_2}$, for each point $\mathbf{y}^k$ can be significantly less than the full number of mesh points $n^{d_1+d_2}$. Some experiments were carried out using this method. The function (9a) yielded very good results after three node points; the function (9b) gave only fair results with up to six node points. A chief advantage of this method appears to be that the results tend to converge relatively smoothly; while the previous strategies would often give results that jumped around irregularly.

Obviously, the approach of this fourth strategy could be carried further: each subspace $\mathbf{x}_1$ and $\mathbf{x}_2$ could be subdivided into smaller subspaces with consequent savings in the number of evaluations needed.

It is not clear to me when these various strategies will work well and when they will fail. What are the characteristics of the function F which suggest that one or another technique will be most successful? What is the best way to choose a sequence of node points $\mathbf{y}^k$? Perhaps some later analysis or accumulation of experience may shed light on these questions. For the present I believe it is useful to have a variety of strategies which one may simply try out when an expensive multidimensional integral confronts one.

III. SUMMARY

Two new methods have been presented for trying to deal with multidimensional integrals in systematic manners that allow one to judge the accuracy in terms of experimental observations of how the computer outputs converge. The first method is based upon a simple analysis of the error terms when high-accuracy numerical quadrature rules are used. The second method has a geometric conception, with the function being fitted along sets of lines passing through selected node points in the multidimensional space. Several strategies within this second method have been described, with a success rate (at least for the rather difficult test problems studied here) that calls for considerable further work before one would be tempted to market this second method. The numerical success of the first method, on the other hand, is quite encouraging; and the first method is, furthermore, simpler to understand and to implement.

[1] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration* (Academic, New York, 1975). Chapter 5 deals with multidimensional integrals.