JOURNAL OF COMPUTATIONAL PHYSICS 27, 192–203 (1978)

A New Algorithm for Adaptive Multidimensional Integration

G. PETER LEPAGE

Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305

Received November 10, 1976; revised June 15, 1977

A new general purpose algorithm for multidimensional integration is described. It is an iterative and adaptive Monte Carlo scheme. The new algorithm is compared with several others currently in use, and shown to be considerably more efficient than all of these for a number of sample integrals of high dimension ($n \gtrsim 4$).

1. INTRODUCTION

One of the most important computational problems facing physicists today is the evaluation of multidimensional integrals with complicated and sometimes very poorly behaved integrands. To cite but a single example drawn from elementary particle theory, integrals over four or more variables are routinely encountered in the computation of scattering amplitudes using Feynman perturbation theory [1]. In this note a new algorithm for multidimensional integration is described. It employs an iterative and adaptive Monte Carlo scheme.

Among the characteristics of this algorithm are:

(a) A reliable error estimate for the integral is readily computed.

(b) The integrand need not be continuous for the algorithm to function and, in particular, step functions pose no problem. Thus integration over hypervolumes of irregular shape is straightforward.

(c) The convergence rate is independent of the dimension of the integral.

(d) The algorithm is adaptive. It automatically concentrates evaluations of the integrand in those regions where the integrand is largest in magnitude.

Characteristics (a)–(c) are common to all Monte Carlo methods [2–4]. Characteristic (d) is the single most important feature of this algorithm. The major problem in multidimensional integration is the exponential growth with increasing dimension of the integration volume over which the integrand must be sampled. For example, applying any five-point integration rule (e.g., Gauss–Legendre integration) along each coordinate axis of a nine-dimensional integral requires $5^9 \sim 2 \times 10^6$ evaluations of the integrand—a formidable number for all but the simplest integrands. Thus any

* Work supported by the Energy Research and Development Administration.

0021-9991/78/0272-0192$02.00/0

general purpose integration algorithm for use in high dimensions should be adaptive.

In Section 2 we briefly review the general features of Monte Carlo integration [2–4] and previous work using this approach. In Section 3 we describe the new algorithm for integration over a single variable. The method is generalized for use in n dimensions in Section 4. Finally, in Section 5, we compare the performance of this algorithm with a number of others currently in use. It is shown to be considerably more efficient for several sample integrals of high dimension ($n \gtrsim 4$). In the Appendix we discuss some simple modifications of the basic approach which may improve it in particular applications.

2. Monte Carlo Integration

Consider the integral of a function of n variables $\mathbf{x} = (x_1, \ldots, x_n)$ over a volume Ω:

$$I = \int_\Omega d\mathbf{x}\, f(\mathbf{x}).$$

If M points $(\mathbf{x})$ are randomly selected from a distribution of points in Ω with density $p(\mathbf{x})$, it is easily shown that the integral is approximated by

$$\begin{aligned} S^{(1)} &= \frac{1}{M} \sum_{\mathbf{x}} \frac{f(\mathbf{x})}{p(\mathbf{x})} \\ &\to I \qquad \text{as} \quad M \to \infty, \end{aligned} \tag{1}$$

where the probability density function is normalized to unity:

$$\int_\Omega d\mathbf{x}\, p(\mathbf{x}) = 1.$$

The quantity $S^{(1)}$ is expected to fluctuate about the true value of the integral as different sets of M random points are chosen. The variance of this fluctuation is given by

$$\sigma^2 = \left\{ \int_\Omega d\mathbf{x}\, \frac{f^2(\mathbf{x})}{p(\mathbf{x})} - \left[\int_\Omega d\mathbf{x}\, f(\mathbf{x}) \right]^2 \right\} M^{-1}.$$

For M large, this quantity is approximated by

$$\sigma^2 \simeq \frac{S^{(2)} - (S^{(1)})^2}{M - 1}, \tag{2}$$

where

$$S^{(2)} = \frac{1}{M} \sum_{\mathbf{x}} \left(\frac{f(\mathbf{x})}{p(\mathbf{x})} \right)^2.$$

The standard deviation σ indicates the accuracy of $S^{(1)}$ as an estimate of I. Note that reliable estimates of the variance are possible only if the integral

$$\int_\Omega d\mathbf{x}\,\frac{f^2(\mathbf{x})}{p(\mathbf{x})}$$

is finite, though even if it is not, $S^{(1)}$ may still approximate I on the average.

There are a number of techniques used to reduce the variance σ^2 for fixed M. Two of the more popular methods used in multidimensional integration are:

(a) *Importance sampling.* Here the density $p(\mathbf{x})$ is varied to reduce σ^2. As is well known, the optimal variance is obtained when

$$p(\mathbf{x}) = |f(\mathbf{x})| \Big/ \int_\Omega d\mathbf{x}\,|f(\mathbf{x})|. \tag{3}$$

Thus when using importance sampling function evaluations are concentrated where the integrand is largest in magnitude (whether or not it is flat there).

(b) *Stratified sampling.* To reduce the variance, the integration volume can be subdivided into N smaller volumes of varying sizes. Then a Monte Carlo integration is performed in each subvolume using M/N random points. The variance is varied by changing the relative sizes and locations of the subvolumes and is minimized when the contributions to σ^2 from each subvolume are identical ($=\sigma^2/N$). Thus when using stratified sampling function evaluations are concentrated where the potential error is largest—i.e., where the integral is both large and rapidly changing.

These and other methods of variance reduction appear to be inappropriate for general purpose integration algorithms as they require detailed knowledge of the integrand's behavior prior to implementation. However, Sheppey[1] has devised an iterative algorithm which uses information generated about the integrand during a Monte Carlo integration to reduce the variance in subsequent integrations. He employs stratified sampling. Initially the algorithm divides the n-dimensional integration volume (a hypercube) into N^n identical hyprecubes using a uniform rectangular grid. A two-point Monte Carlo integration is performed in each hypercube generating a contribution to the total integral and to the variance. The variances from the hypercubes are then used to define new grid spacings along each axis for use in the next iteration, keeping the total number of hypercubes constant. Thus over a number of iterations hypercubes can be gradually concentrated where the variance was initially largest, and the variance reduced.

This algorithm has enjoyed widespread usage in theoretical physics (see [1], for example) and is quite successful for many applications in two or more dimensions. The procedure's greatest strength lies in its ability to adapt to the integrand being considered. However the extent to which it can adapt is determined by the number of

[1] A computer program using Sheppey's method is described in [5].

grid increments along each axis (N), and N in turn is limited by the total number of integrand evaluations (M) allowed per iteration (n = dimensionality):

$$M = 2N^n.$$

This limitation proves to be a serious handicap in high dimensions (e.g., $M < 10^6$ for $n = 9 \Rightarrow N \leqslant 4$), as will be illustrated in Section 5.

A number of other techniques similar to Monte Carlo methods have also been suggested for multidimensional integration [2, 3, 6, 7]. These quasi-Monte Carlo techniques generally converge faster than $1/M^{1/2}$ for analytic integrands. However they are not adaptive and so may require prohibitively large numbers of function evaluations in high dimensions when the integrand has sharp peaks. Furthermore it is not clear that they are able to integrate functions which are neither analytic nor even continuous. Also no simple estimate of the error is available in most cases.

3. The Algorithm in One Dimension

The restrictive relation between the numbers of increments and of function calls in Sheppey's method can be avoided using importance sampling rather than stratified sampling to reduce the variance. Intuitively importance sampling seems the inferior of the two methods [3]. However, in practice the ability to adapt is the overriding consideration for high dimensions.

Like Sheppey's, the algorithm described here is iterative. To illustrate consider a one-dimensional integration:

$$I = \int_0^1 dx\, f(x).$$

Initially an M-point Monte Carlo integration is preformed with a uniform probability density ($p(x) = 1$). Besides providing estimates of the integral and the possible error (Eqs. (1) and (2)), the M integrand evaluations can also eb used to define an improved probability density for use in the next (M-point) Monte Carlo integration. In this fashion an empirical variance reduction can be gradually introduced over several iterations.

There exist standard numerical techniques for generating lists of evenly distributed "pseudorandom" numbers. It is more difficult, however, to generate numbers from an arbitrary distribution with density $p(x)$. For present purposes it is natural that $p(x)$ be a step function with N steps. The probability of a random number being chosen from any given step is defined to be a constant, equal to $1/N$ for all steps ($0 = x_0 < \cdots < x_N = 1$, $\Delta x_i = x_i - x_{i-1}$):

$$p(x) = 1/N\Delta x_i\,, \qquad x_i - \Delta x_i \leqslant x < x_i\,, \quad i = 1,\ldots, N,$$

where

$$\sum_{i=1}^{N} \Delta x_i = 1.$$

The probability distribution is tailored to particular integrands simply by adjusting the increment sizes Δx_i. In practice N is limited by the computer storage space available and must be held constant from iteration to iteration ($N = 50$ to 100 typically).

Given M integrand evaluations, the probability distribution or, equivalently, the increment density is refined by subdividing each increment Δx_i into $m_i + 1$ subincrements, where

$$m_i = K\left(\bar{f}_i \, \Delta x_i \Big/ \sum_j \bar{f}_j \, \Delta x_j\right) \tag{4}$$

and

$$\bar{f}_i = \sum_{x \in x_i - \Delta x_i}^{x_i} |f(x)|$$

$$\propto \frac{1}{\Delta x_i} \int_{x_i - \Delta x_i}^{x_i} dx \, |f(x)|.$$

Thus each increment is subdivided into as many as $(K + 1)$ subincrements (K is fixed at 1000 typically), and its contribution to the weight function increased in proportion to its contribution to the integral of $|f(x)|$, as required by (3). As it is desirable to restore the number of increments to its original value ($=N$), groups of the new increments must be amalgamated into larger increments, the number of subincrements in each group being constant (to preserve the relative increment density). The net effect is to alter the increment sizes, while keeping the total number constant, so that the smallest increments occur where $|f(x)|$ is largest. The new grid is used and further refined in subsequent iterations until the optimal grid ahs been obtained (i.e., $m_i = m_j$, $i, j = 1, \ldots, N$).

In practice it is best to damp the subdivision algorithm thereby avoiding rapid, destabilizing changes in the grid from iteration to iteration. For example, this can be done by using [5]

$$m_i = K\left\{\left[\frac{\bar{f}_i \, \Delta x_i}{\sum_j f_j \, \Delta x_j} - 1\right] \frac{1}{\log[\bar{f}_i \, \Delta x_i / \sum_j \bar{f}_j \, \Delta x_j]}\right\}^{\alpha}$$

in place of (4). The parameter α determines the rate of convergence and is typically set between 1 and 2.

The values $\bar{f}_i$ must be discarded after each iteration because of storage limitations.

However, a cumulative estimate of the integral and its error can be made which uses every evaluation of the integrand:

$$\bar{I} = \sigma_I^2 \sum_i \frac{I_i}{\sigma_i^2}$$
$$\sigma_I = \left[\sum_i \frac{1}{\sigma_i^2}\right]^{-1/2}. \tag{5}$$

Here I_i and σ_i^2 are the integral and standard deviation estimated in iteration i using Eqs. (1) and (2). When the integrand has high, narrow peaks, I_i and σ_i are sometimes badly underestimated in the earliest iterations (before the algorithm has adapted). To partially correct for this effect, it may be better to replace Eqs. (5) by

$$\bar{I} = \sum_i I_i \left(\frac{I_i^2}{\sigma_i^2}\right) \Big/ \sum_i \frac{I_i^2}{\sigma_i^2},$$
$$\sigma_I = \bar{I}\left[\sum_i \frac{I_i^2}{\sigma_i^2}\right]^{-1/2}. \tag{6}$$

In any event the χ^2

$$\chi^2 \simeq \sum_i \frac{(I_i - \bar{I})^2}{\sigma_i^2} \qquad \text{or} \qquad \sum_i \frac{(I_i - \bar{I})^2}{\bar{I}^2} \frac{I_i^2}{\sigma_i^2} \tag{7}$$

should not greatly exceed the number of iterations (minus one). The algorithm cannot be trusted when it does.

The number of iterations and the number of integrand evaluations per iteration needed clearly depend upon the complexity of the integrand and the accuracy being sought. In general it is best to use as few integrand evaluations per iteration as are required by the algorithm to converge smoothly to the optimal grid, at least until it has converged. Once the optimal grid has been found (approximately), the uncertainty (σ_I) in the integral becomes roughly proportional to $1/M^{1/2}$, M being the number of integrand evaluations.

4. The Algorithm in n Dimensions

The algorithm described above is easily generalized to handle integrals of arbitrary dimensionality. To illustrate the modifications consider

$$I = \int_0^1 dx \int_0^1 dy\, f(x, y).$$

A separable probability density function is adopted to limit data storage requirements:

$$p(x, y) = p_x(x)\, p_y(y).$$

In this case the optimal densities are easily shown to be[2]

$$p_x(x) = \left[\int_0^1 dy\, \frac{f^2(x, y)}{p_y(y)}\right]^{1/2} \Big/ \int_0^1 dx \left[\int_0^1 dy\, \frac{f^2(x, y)}{p_y(y)}\right]^{1/2}$$

with a similar result for $p_y(y)$. Thus the one-dimensional algorithm can be applied along each axis but with $\bar{f}_i$ (in Eq. (4)) defined by

$$(\bar{f}_i)^2 = \sum_{x \in x_i - \Delta x_i}^{x_i} \sum_y \frac{f^2(x, y)}{p_y^2(y)}$$

$$\propto \frac{1}{\Delta x_i} \int_{x_i - \Delta x_i}^{x_i} dx \int_0^1 dy\, \frac{f^2(x, y)}{p_y(y)}$$

for the x axis and with an analogous definition for the y axis. The generalization to arbitrary dimension is obvious.

5. Numerical Examples

We have encoded this algorithm in a Fortran IV program called VEGAS and compared its performance with a number of other multidimensional integration methods. A sampling of these results is presented in Tables I–V.

The test integral for Tables I and II has a spherically symmetric Gaussian placed in the center of the integration region:

$$I_n = \left(\frac{1}{a\pi^{1/2}}\right)^n \int_0^1 d^n x \exp\left(-\sum_{i=1}^n \frac{(x_i - \frac{1}{2})^2}{a^2}\right), \tag{8}$$

where $a = 0.1$ and $n = 4, 9$. Partial results from several iterations of VEGAS are shown, illustrating the convergence of the algorithm ($\alpha = 2.0, 1.0$ for $n = 4, 9$). For comparison estimates of I_n using crude Monte Carlo ($p(x) = 1$) and using Gauss–Legendre formulas along each axis are also exhibited. Of course VEGAS is far superior to crude Monte Carlo. VEGAS is also considerably more efficient than Gauss–Legendre integration (especially when $n = 9$), even though the integrand and

[2] For example, this result can be obtained using functional methods (together with Lagrange multipliers to preserve density normalizations):

$$\frac{\delta}{\delta p_x(x)}\left[\sigma^2[p_x, p_y] + \lambda_x \int_0^1 dx\, p_x(x) + \lambda_y \int_0^1 dy\, p_y(y)\right] = 0,$$

$$\Rightarrow \frac{-1}{p_x^2(x)} \int_0^1 dy\, \frac{f^2(x, y)}{p_y(y)} + \lambda_x = 0,$$

where the definition of σ^2 from Section 2 has been used. The final result follows immediately.

TABLE I

Integration of a Single Gaussian in Four Dimensions[a]

Vegas					
	Per iteration		Cumulative		
Iteration	I_i	σ_i	I	σ_I	No. of integrand evaluations
1	0.790	0.313	0.790	0.313	1,000
3	0.970	0.063	0.977	0.059	3,000
5	0.952	0.023	0.975	0.018	5,000
10	1.008	0.016	0.994	0.007	10,000
Crude Monte Carlo			0.890	0.149	10,000

Gauss–Legendre Integration		
No. of points/axis	Integral	No. of integrand evaluations
5	6.664	625
6	0.164	1,296
10	0.892	10,000
13	1.008	28,561

[a] The exact result is 1.

all of its derivatives are analytic and bounded throughout the integration region. These examples illustrate the importance of using adaptive algorithms for integration in high dimension.

In Table III VEGAS is compared with a code (SHEP) written by Sheppey and employing the algorithm described in Section 2. Here the test integral has two spherically symmetric Gaussians equally spaced along the diagonal of theintegration volume:

$$I_n = \frac{1}{2}\left(\frac{1}{a\pi^{1/2}}\right)^n \int_0^1 d^n x \left\{\exp\left(-\sum_{i=1}^{n} \frac{(x_i - \frac{1}{3})^2}{a^2}\right) + \exp\left(-\sum_{i=1}^{n} \frac{(x_i - \frac{2}{3})^2}{a^2}\right)\right\}. \quad (9)$$

The integration was done for $n = 2, 4, 7, 9$ with $a = 0.1$. The "Optimal standard deviation" quoted in this table is the standard deviation computed on any iteration after the optimal grid has been achieved. Results averaged over 15 iterations (Eqs. (6)) are also presented.

For high dimensions ($n \geqslant 4$), VEGAS converges with far fewer function evaluations than SHEP. The number of function evaluations is independent of the number of

TABLE II

Integration of a Single Gaussian in Nine Dimensions[a]

VEGAS					
	Per iteration		Cumulative		
Iteration	I_i	σ_i	I	σ_I	No. of integrand evaluations
1	0.007	0.005	0.007	0.005	10^4
3	0.643	0.070	0.612	0.064	3×10^4
5	1.009	0.041	0.963	0.034	5×10^4
10	1.003	0.008	1.001	0.005	10^5
Crude Monte Carlo			0.843	0.360	10^5

Gauss–Legendre Integration		
No. of points/axis	Integral	No. of integrand evaluations
5	71.364	2.0×10^6
6	0.017	1.0×10^7
10	0.774	10^9
15	1.002	3.8×10^{10}

[a] The exact result is 1.

TABLE III

Integration of a Double Gaussian in n Dimensions[a]

	VEGAS				
	$n = 2$	$n = 4$	$n = 7$	$n = 7$	$n = 9$
No. function evaluations/iteration	20,000	20,000	32,000	160,000	100,000
No. increments/axis	50	50	50	50	50
Optimal standard deviation	0.007	0.02	0.05	0.02	0.06
Cumulative result after 15 iterations	0.999	1.003	1.015	0.991	0.96
	±0.002	±0.006	±0.015	±0.007	±0.04
	SHEP				
	$n = 2$	$n = 4$	$n = 7$	$n = 7$	$n = 9$
No. function evaluations/iteration	20,000	20,000	32,768	170,000	3,906,250
No. increments/axis	100	10	4	5	5
Optimal standard deviation	0.0008	0.03	1.0[b]	0.25	**[c]
Cumulative result after 15 iterations	1.0001	1.004	*	0.90	**
	±0.0002	±0.008	*	±0.04	**

[a] The exact result is 1.
[b] Algorithm would not converge.
[c] Too large to be tried.

increments per axis in VEGAS, and so it is better able to adapt to the integrand in high dimension than is SHEP. Note that when the algorithms are equally adaptive ($n \leqslant 4$), SHEP performs as well as or better than VEGAS—as expected.

In Table IV we compare VEGAS with the quasi-Monte Carlo methods suggested by Tsuda [7] and by Haselgrove [6]. The test integral is the most difficult considered by Tsuda [7]:

$$I = \int_0^1 dy_1 \cdots \int_0^1 dy_8 f(y_1 \cdots y_8),$$
$$f(y_1 \cdots y_8) = \prod_{i=1}^{8} \frac{c}{c+1}\left(\frac{c+1}{c+y_i}\right)^2, \tag{10}$$

where c is chosen such that $f(0 \cdots 0) = 10^4$. Neither of the quasi-Monte Carlo methods allows a simple error estimate and so this must be inferred from the convergence of the estimates of I. VEGAS is the superior algorithm though Tsuda's method is almost as efficient when many function evaluations are employed.

Finally, we present the results of a seventh order calculation (in Feynman perturbation theory) of part of the decay rate of orthopositronium into three photons.

TABLE IV

Integrand (10) Using VEGAS, Tsuda's Algorithm [7], and Haselgrove's Algorithm [6][a]

No. of integrand evaluations	VEGAS	TSUDA ($n = 40$)	HASELGROVE (s_8)
1,000	1.083 ± 0.085	1.248	11.019
5,000	1.003 ± 0.004	0.982	2.946
10,000	1.000 ± 0.002	1.006	1.974
20,000	1.000 ± 0.001	1.002	1.484

[a] The exact result is 1. The numbers for Tsuda's method are from Table I in Ref. [7].

TABLE V

Contribution of Ladder Graph to Decay Rate of Orthopositronium[a]

	VEGAS ($n = 5$)	SHEP ($n = 5$)
No. function evaluations/iteration	33,000	33,000
No. increments/axis	50	7
Optimal standard deviation	0.73	1.21
Cumulative result after 15 iterations	-5.82 ± 0.22	-6.34 ± 0.35

[a] In units of $\alpha\Gamma_0/\pi$.

The contribution presented here is from the ladder kernel ($\Gamma_{g'}^{S}$ in Ref. [8]). No analytic result is known; the best numerical estimate is $(-5.82 \pm .01)\,\alpha\Gamma_0/\pi$. Note that the integral contains a step function $\theta(1 - x_1 - x_2)$ and is therefore zero throughout half the integration volume.

6. Conclusions

It is clear that any general purpose multidimensional integration method should be adaptive. We have demonstrated that adaptive Monte Carlo methods are quiet effective, especially in high dimensions or with nonanalytic integrands where simple generalizations of the well-known one-dimensional methods are not so generally applicable. Of course, if the integrand is analytic and smooth, techniques such as Gauss–Legendre or quasi-Monte Carlo integration may frequently be superior to adaptive Monte Carlo integration, though for such integrands the latter is more than sufficient to obtain the three or four significant digits usually required in scientific applications.

Note that this algorithm's overhead and storage requirements grow only linearly with dimension, because separable distribution densities are employed. Thus it should be effective even in very high dimension ($n = 20, 30, 40,...$).

Appendix

The use of importance sampling as described in Section 3 is formally equivalent to making the variable transformation $x = g(y)$

$$I = \int_0^1 dx\, f(x) = \int_0^1 dy\, f(g(y))\, g'(y), \tag{11}$$

with $g'(y) = p(g(y))^{-1}$ and performing a Monte Carlo integration with uniformly distributed points $\{y_i\}$. The algorithm presented in Section 3 adjusts $g'(y)$ such that $g'(y) \propto |f(g(y))|^{-1}$ thereby improving the Monte Carlo estimate of the integral. For some integrands it is possible to further reduce the variance through use of other well-known methods (stratified sampling, antithetic variates, quasi-random numbers,... [2]) applied to the smoothed integrand $f(g(y))\, g'(y)$ in (11). For example, rather than choosing M random points uniformly distributed on $0 \leqslant y \leqslant 1$, the sampling can be stratified by selecting two points in each of $M/2$ subintervals of $[0, 1]$.

The optimal definition of $g'(y)$ may differ from that in Section 3 when additional variance reduction is used. Thus for stratified sampling, $g'(y)$ should perhaps be chosen such that the variation (rather than the value itself) of $g'(y) f(g(y))$ in each subinterval is uniform over all subintervals.

Trial runs of a modified VEGAS, incorporating stratified sampling, suggest considerable gains in efficiency in low dimensions (where the modified algorithm is very similar to Sheppey's) but very little improvement in higher dimensions. Work is still in progress on this and similar modifications of VEGAS.

ACKNOWLEDGMENTS

The author is indebted to Dr. S. J. Brodsky for his comments and suggestions. He also thanks Drs. K. J. F. Gaemers, W. E. Caswell, and J. Friedman for many discussions.

REFERENCES

1. J. ALDINS *et al.*, *Phys. Rev. D* **1** (1970), 2378; P. PASCUAL AND E. DE RAFAEL, *Nuovo Cimento Lett.* **4** (1970), 1144; G. GRAMMAR, JR. AND T. KINOSHITA, *Nucl. Phys. B* **80** (1974), 461.
2. J. M. HAMMERSLEY AND D. C. HANDSCOMB, "Monte Carlo Methods," Chaps. 3, 5, Methuen, London, 1964.
3. A. H. STROUD, "Approximate Calculation of Multiple Integrals," Chap. 6, Prentice–Hall, Englewood Cliffs, N. J., 1971.
4. J. H. HALTON, *SIAM Rev.* **12** (1970), 1.
5. A. DUFNER, *in* "Proceedings of the Colloquium on Computational Physics, Marseille, 1970," p. I-46 (unpublished); B. LAUTRUP, *in* "Proceedings of the Colloquium on Computational Physics, Marseille, 1971," p. I-57 (unpublished).
6. C. B. HASELGROVE, *Math. Comp.* **15** (1961), 323.
7. T. TSUDA, *Numer. Math.* **20** (1973), 377.
8. W. E. CASWELL *et al.*, *Phys. Rev. Lett.* **38** (1977), 488.