CLNS-80/447
March,1980

VEGAS - An Adaptive Multi-dimensional Integration Program

G. Peter Lepage
Newman Laboratory of Nuclear Studies
Cornell University
Ithaca, New York 14853

Abstract

This note describes the use of VEGAS - a new program for multi-dimensional integration. The theoretical considerations leading to VEGAS are briefly described. A Fortran listing of the program is included together with detailed directions on its use.

Index

I. Understanding VEGAS

VEGAS estimates the integral (Ω = rectangular integration volume)

$$I = \int_{\Omega} d^n x \; f(\vec{x}) \tag{1}$$

by computing the integrand at N random points, $\{\vec{x}_i\}_1^N$, in Ω and forming the weighted average

$$I \simeq S = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\vec{x}_i)}{p(\vec{x}_i)} \quad . \tag{2}$$

The random points are chosen in Ω with density $p(\vec{x})$, which will be discussed below ($\int_{\Omega} d^n x \; p(\vec{x}) = 1$). VEGAS makes m estimates of the integral, $\{S_\alpha\}_{\alpha=1}^m$, each using N evaluations of the integrand Eq. (2)). These m estimates are combined to give a cumulative estimate $\bar{S}$:

$$I \simeq \bar{S} = \bar{\sigma}^2 \sum_{\alpha} \frac{S_\alpha}{\sigma_\alpha^2} \tag{3}$$

where σ_α is the approximate uncertainty in S_α as an estimate of I:

$$\sigma_\alpha^2 = \frac{1}{N-1} \; \frac{1}{N} \sum_{i=1}^{N} \frac{f^2(\vec{x}_i)}{p(\vec{x}_i)} - S_\alpha^2 \quad , \tag{4}$$

and where $\bar{\sigma}$ is the approximate uncertainty in $\bar{S}$:

$$\frac{1}{\bar{\sigma}^2} = \sum_{\alpha} \frac{1}{\sigma_\alpha^2} \quad . \tag{5}$$

VEGAS also determines whether or not the various estimates are consistent, one with the other, by computing the χ^2 per degree of freedom (χ^2/dof):

$$\chi^2/\text{dof} = \frac{1}{m-1} \sum_{\alpha=1}^{m} \frac{(S_\alpha - \bar{S})^2}{\sigma_\alpha^2} \quad . \tag{6}$$

When the algorithm is working properly, one expects a χ^2/dof not much greater than one, since $(S_\alpha - \bar{S})^2 \sim O(\sigma_\alpha^2)$. Otherwise, the various S_α do not agree "within errors".

If N is made sufficiently large and if $f(\vec{x})$ is square integrable, the Central Limit Theorem implies that the distribution of S_α's about I becomes Gaussian. Then Eqs. (3) and (5) are valid, and $\bar{S}$ is a reliable estimate of I--i.e.

$$\begin{aligned} |\bar{S}-I| &\leq \bar{\sigma} \quad \text{with 68\% confidence} \\ &\leq 2\bar{\sigma} \quad \text{with 95\% confidence} \\ &\ldots \quad \text{etc.} \end{aligned} \tag{7}$$

The χ^2/dof (Eq. (6)) tests to some extent whether the distribution is Gaussian--χ^2/dof >> 1 implies a decidedly non-Gaussian distribution of S_α's, in which case $|\bar{S}-I|$ may be substantially larger than $\bar{\sigma}$. The minimum number of points (N) required per iteration is highly dependent upon the integrand, and is usually determined empirically. Smooth integrands require few points; integrands with high, narrow peaks or with many fluctuatuions in sign require more.

When $f(\vec{x})$ is integrable, but not square integrable, $\bar{S}$ may still converge to I. However the error estimates are completely unreliable, in general being too small.

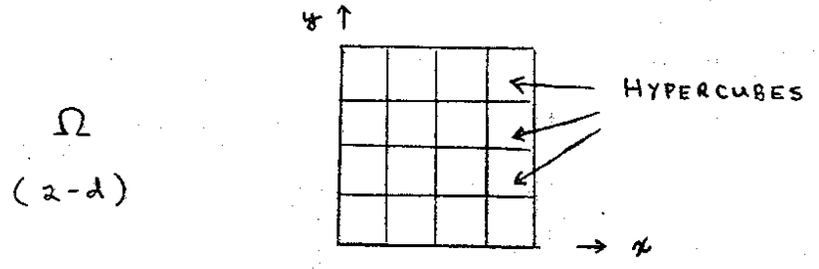
In the simplest form of Monte Carlo integration, the random integration points are uniformly distributed--i.e., $p(\vec{x})$ = constant. In VEGAS, the density $p(\vec{x})$ is modified so as to minimize σ_α^2 (Eq. (4)). Uniformly distributed random points are employed in the first iteration of the integration algorithm (i.e., in determining S_1). The information gained about $f(\vec{x})$ in this first sampling is used to define a new density $p'(\vec{x})$ which reduces σ_α^2 in the next iteration (for S_2). After each subsequent iteration, $p(\vec{x})$ is again refined for use in the next. In this fashion, σ_α^2 is gradually reduced over several iterations (even though N = constant), and the

estimates $\bar{S}$ of the integral are progressively improved.

Theoretically, σ_α^2 is minimized when

$$p(\vec{x}) = \frac{|f(\vec{x})|}{\int_\Omega d^n x |f(\vec{x})|} \tag{8}$$

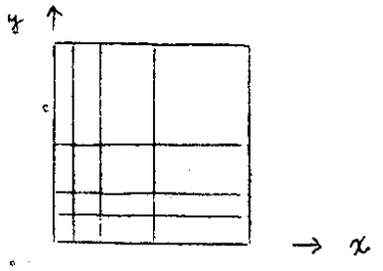
--that is when sample points $\{\vec{x}_i\}$ are concentrated where the integrand is largest in magnitude. In VEGAS, this ideal is approximated by dividing the integration volume into hypercubes using a rectangular grid:



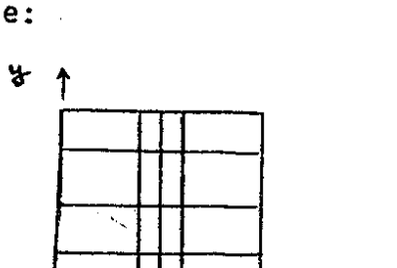
Ω

(2-d)

Random points are distributed so that the average number of points falling in any given hypercube is the same as in any other. (In high dimensions this average is much less one, since the number of hypercubes ($\sim 50^n$) is much greater than N). From iteration to iteration, the increment sizes on each axis are adjusted so as to concentrate hypercubes (and therefore sample points) in the regions where $|f(\vec{x})|$ is largest. For example, if f(x,y) has a high peak in x and y at the origin, the optimal grid might be:



while if the integrand peaks in x at x = 1/2 but is smooth in y, the most efficient grid might resemble:



Thus over several iterations, VEGAS adjusts the density of points, to suit the integrand, by varying the increment sizes along each axis. This readjustment continues until the optimal grid is obtained (i.e., the one most similar to (8)), beyond which point σ_α ceases to decrease with each iteration (though $\bar{\sigma}$ continues to decrease, now falling like $\sim 1/\sqrt{Nm}$ as better statistics accumulate).

A more detailed description of the algorithm used in VEGAS and the theory behind it is given in G. P. Lepage, J. Comp. Phys. 27, 192 (1978). Note that the modification discussed in the Appendix of this reference has been incorporated into VEGAS, thereby substantially improving its efficiency in low dimensions (n=1,...,4).

II. Using VEGAS

A. Input

Subroutine VEGAS is written in standard Fortran (see Appendix). VEGAS is called by the statements:

```
      EXTERNAL FXN
        .
        .
        .
      CALL VEGAS(NDIM,FXN,AVGI,SD,CHI2A)
```

where the input variables are

NDIM - n, the number of dimensions (≤ 10);

FXN - Fortran name of the function subprogram which computes the integrand $f(\vec{x})$.

The output variables are (double precision)

AVGI - $\overline{S}$, the cumulative estimate of the integral (Eq. (3));

SD - $\overline{\sigma}$, the standard deviation of $\overline{S}$ from I (Eq. (5));

CHI2A - χ^2 per degree of freedom (Eq. (6)).

Several other parameters can be set before calling VEGAS. These are contained in common block BVEG1:

```
COMMON/BVEG1/NCALL,ITMX,NPRN,NDEV,XL(10),XU(10),ACC
```

Each of these variables has a default value (set in the BLOCK DATA subprogram). If any of these defaults is to be overridden, this COMMON card must be included in the calling program. The paramenters are defined as follows (defaults are in []):

NCALL - N, the approximate number of integrand evaluations per iteration [5000];

ITMX - m, the maximum number of iterations [5];

NPRN > 0 integral, standard deviation, χ^2 and grid information are printed (on unit NDEV) for each iteration;

= 0 integral, standard deviation and χ^2 are printed for each iteration;

< 0 nothing is printed by VEGAS;

(see Section II.B) [5];

NDEV - Fortran device number for output from VEGAS [6];

XL(I) - lower integration limit on I-th axis [0.];

XU(I) - upper integration limit on I-th axis [1.];

ACC - algorithm stops when the relative accuracy, |SD/AVGI|, is less than ACC; accuracy is not checked when ACC < 0; [-1.];

The integrand is encoded as a function subprogram:

```
FUNCTION FXN(X,WGT)
DOUBLE PRECISION X(10), WGT, FXN
    .
    .
    .
FXN = ... Integrand value at point (X(I); I=1, NDIM)
RETURN
END
```

Note that integration over regions Ω which are not rectangular is made possible by embedding Ω in a rectangular region (specified by XL,XU) and setting FXN to zero for points X(I) lying outside Ω.

The user must also provide a random number generator:

```
SUBROUTINE RANDA(N,RAND)
```

where (RAND(I),I=1,N) are random numbers (single precision) between 0 and 1.

B. Output

An estimate of the integral averaged over all iterations (AVGI), its standard deviation (SD) from the exact value, and the χ^2/dof (CHI2A) are always returned to the calling program. In addition, if NPRN $\geq$ 0, the following information is printed out for each iteraction (on the Fortran unit specified by NDEV):

INTEGRAL - the integral as estimated from that iteration alone (S_α, Eq. (2)), and the estimate averaged over all iterations up to that point ($\bar{S}$, Eq. (3));

STD DEV - the standard deviations for these estimates of the integral (σ_α, $\bar{\sigma}$; Eqs. (4), (5));

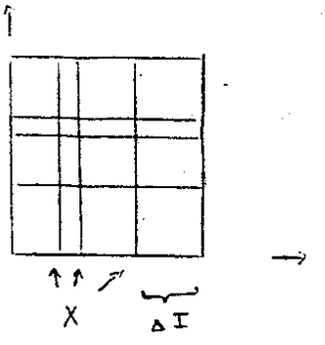
CHI**2 PER IT'N - the χ^2/dof for all iterations up to that point (Eq. (6)).

The input parameters are also listed, at the beginning. When NPRN $\geq$ 1, information about the grid used is also printed for each iteration. The following details are tabulated for each axis:

X - locations along the axis of axis divisions defining the grid; these are normalized to lie between 0 and 1;

DELT I - contribution ΔI to the total integral coming from the increment to the left of X; thus $\sum \Delta I = I$ when the sum is over all increments on any single axis.

This information is printed for every NPRN-th increment on an axis; NPRN=1 implies information is printed for all increments.



Finally, note that integration information from the last iteration of VEGAS is always contained in common block BVEG4:

```
COMMON/BVEG4/CALLS,TI,TSI
```

where

CALLS = the number of integration points used in that iteration;

TI = the integral as estimated from that iteration alone;

TSI = the standard deviation of this estimate from the exact value.

C. Multiple Entry Points

The integration algorithm can be called in four ways, each differing from the other in the grid used for the first iteration (i.e., in the distribution of random points) and/or in the initial values of the cumulative variables (i.e., iteration number, cumulative estimate of the integral, etc.). Each is called in the same fashion:

```
CALL   VEGAS    (NDIM,FXN,AVGI,SD,CHI2A)
       VEGAS1
       VEGAS2
       VEGAS3
```

The function of each is as follows:

VEGAS - sets all cumulative variables to zero before beginning the first iteration, and uses a uniform grid for that iteration. This is the routine normally called (see Section II.A).

VEGAS1 - sets all cumulative variables to zero before the first iteration, but does not initialize the grid. The grid last used by VEGAS is employed in the first iteration. For example, consider:

```
COMMON/BVEG1/NCALL,ITMX,...
EXTERNAL F1, F2
   .
   .
   .
ITMX = 5
CALL VEGAS (NDIM,F1,ANS1,ERR1,CHIS1)
ITMX = 2
CALL VEGAS1(NDIM,F2,ANS2,ERR2,CHIS2)
```

Here the integral of F1 is evaluated in the usual fashion; the grid adapts to the integrand over five iterations. The grid generated for F1 in the last iteration (i.e., the fifth) of VEGAS is used in the first iteration of VEGAS1, which estimates the integral of F2. If F2 is similar in structure

to F1, far fewer iterations are required for VEGAS1 since the starting grid is close to optimal. As another example, consider:

```
      .
      .
      .
ITMX = 10
NCALL = 2000
CALL VEGAS (NDIM,FXN,AVGI,SD,CHI2A)
ITMX = 5
NCALL = 100000
CALL VEGAS1(NDIM,FXN,AVGI,SD,CHI2A)
```

Here the integral of FXN is estimated in fifteen iterations--the first ten use 2000 integration points per iteration; the last five use 100,000 points per iteration. Since errors tend to be large until the optimal grid is found, it is desirable to use as few integration points as possible in finding it. In this example, the hope is that the optimal grid can be determined in the first ten iterations, using a total of only 20,000 integrand evaluations. Then NCALL is increased to obtain the desired precision in the last five iterations. The final estimate of the integral (AVGI) from VEGAS1 depends only upon the last five iterations (in this example); aside from the grid, all information generated in the first ten iterations of VEGAS is discarded.

Finally, note that any of the parameters discussed in Section II.A can be modified before calling VEGAS1.

VEGAS2 - Initializes neither the grid nor the cumulative variables. In the following example

```
      COMMON/BVEG1/NCALL,ITMX,NPRN,...
      EXTERNAL F
        .
        .
        .
      NPRN = -1
      ITMX = 5
      NCALL = 1000
      CALL VEGAS(NDIM,F,AVGI,SD,CHI2A)
      NPRN = 1
      ITMX = 3
      NCALL = 10**6
      CALL VEGAS2(NDIM,F,AVGI,SD,CHI2A)
```

the integral of F is determined in eight iterations--five with 1000 points, three with 1,000,000 points. As in the examples above, the final grid from VEGAS is used as a starting point in VEGAS2. The difference here is that the final estimate of the integral (AVGI from VEGAS2) is an average over all eight iterations; estimates from the first five iterations are not discarded, as they would be were VEGAS1 used in place of VEGAS2. In VEGAS2, the integration algorithm picks up where VEGAS left off and continues as if there had been no interruption. Finally, any of the parameters discussed in Section II.A can again be modified before calling VEGAS2.

VEGAS3 - No initialization; main integration loop only. VEGAS3 can be used if the calling program is to examine results of each single iteration, as they are generated. For example, the following code

```
      .
      .
      .
      ITMX = 1
      CALL VEGAS (NDIM,F,AVGI,SD,CHI2A)
      DO 1 I = 1,9
         .
         .
         .
      CALL VEGAS3(NDIM,F,AVGI,DS,CHI2A)
    1 CONTINUE
```

is equivalent to a single call to VEGAS with ITMX = 10, except that control is returned to the calling program after each iteraction. Control parameters (Section II.A) should not be modified after VEGAS (and before VEGAS3) is called.

D. Saving Grid Information

Complete information about the grid and about cumulative variables is contained in common block BVEG2:

```
COMMON/BVEG2/IT,NDO,SI,SWGT,SCHI,XI(50,10)
```

where

IT - number of iterations completed;

NDO - number of subdivisions on an axis;

SI - $\sum_{1}^{IT} \frac{S_\alpha}{\sigma_\alpha^2}$

SWGT - $\sum_{1}^{IT} \frac{1}{\sigma_\alpha^2}$

SCHI - $\sum_{1}^{IT} \frac{S_\alpha^2}{\sigma_\alpha^2}$

XI(I,J) - location of the I-th division on the J-th axis, normalized to lie between 0 and 1.

This information can be stored for later use. For example, if the contents of BVEG2 have been stored on Fortran unit 9, VEGAS can be restarted as follows:

```
READ(9)IT,NDO,SI,SWGT,SCHI,XI
ITMX = IT + 5
NCALL = 50000
CALL VEGAS2(NDIM,FXN,AVGI,SD,CHI2A)
```

Here five additional iterations are combined with the earlier results to provide a new and hopefully improved estimate of the integral. If only the grid is needed in future, NDO and XI(I,J) should be saved. The grid is then used as the starting point for the first iteration in VEGAS as follows (for example):

```
READ(9)NDO,XI
ITMX = 5
NCALL = 1000
CALL VEGAS1(NDIM,FXN,AVGI,SD,CHI2A)
```

E. Computing Distributions; Integrating Vectors

VEGAS can be employed to simultaneously compute any number of arbitrary distributions of the sort

$$\frac{dI}{dy} = \int_\Omega d^n x \; f(\vec{x})\,\delta(y-g(\vec{x}))$$

where

$$I = \int dy \; \frac{dI}{dy} \quad .$$

These are computed in the subprogram which generates the integrand (function FXN in Section II.A; supplied by the user). With each integration point supplied to FXN, the probabilistic weight w_i = WGT assigned that point is also supplied. These weights are defined so that the total integral is

$$I \simeq \sum_{i=1}^{N} w_i \; f(\vec{x}_i) \quad .$$

To estimate dI/dy for different values y, the range of $y = g(\vec{x})$ ($\vec{x}$ in Ω) is divided into M increments Δy_j with centers y_j. Then the contribution to I coming from increment Δy_j is just

$$\Delta I_j \simeq \sum_{g(\vec{x})\varepsilon\Delta y_j} w_i \; f(\vec{x}_i)$$

where the sum is over all points $\vec{x}_i$ such that $g(\vec{x}_i)$ lies within the increment. Then dI/dy at $y = y_j$ is approximated by

$$\frac{dI(y_j)}{dy} \simeq \frac{\Delta I_j}{\Delta y_j} \qquad j=1,\ldots,M$$

As an example, consider the integral

$$I = \int_0^{\sqrt{1/2}} dx \int_0^{\sqrt{1/2}} dy \; \cos(x^2+y)$$

for which the distribution dI/dr with $r^2 = x^2 + y^2$ is desired. Variable r ranges from 0 to 1. Dividing this range into 20 equal bins ($\Delta r_j = 1/20$), an appropriate code for the integrand is

```
      FUNCTION FXN(X,WGT)
      DOUBLE PRECISION X(2),WGT,FXN,DI
      COMMON DI(20)
      FXN = DCOS(X(1)**2 + X(2))
      R = DSQRT(X(1)**2 + X(2)**2)
      J = R*20 + 1
      DI(J) = DI(J) + WGT*FXN
      RETURN
      END.
```

Upon completion of IT iterations, dI/dr at $r = (J - 1/2)/20$ is approximately

$$\frac{dI}{dr} \simeq \frac{1}{IT} \frac{DI(J)}{.05} \qquad J=1,\ldots,20$$

VEGAS can also be used to simultaneously integrate any number of functions in addition to $f(\vec{x})$. The integral of any $\tilde{f}(\vec{x})$ is estimated by

$$\int_\Omega d^n x \; \tilde{f}(\vec{x}) \simeq \sum_{i=1}^{N} w_i \; \tilde{f}(\vec{x}_i).$$

This sum can be accumulated in subprogram FXN in much the same manner distributions are accumulated.

F. Modifying Internal Parameters

Some additional parameters controlling VEGAS are contained in common block BVEG3,

```
COMMON/BVEG3/ALPH,NDMX,MDS
```

where (default values are in [ ]):

ALPH - controls the rate at which the grid is modified from iteration to iteration; decreasing ALPH slows modification of the grid (ALPH=0 implies no modification); [1.5];

NDMX - determines the maximum number of increments along each axis; the actual number used varies between NDMX/2 and NDMX; [50];

MDS = 0 VEGAS uses importance sampling only;

≠ 0 VEGAS uses importance sampling + stratified sampling; increments are concentrated either where the integrand is largest in magnitude (MDS=1), or where the contribution to the error is largest (MDS=-1). The program chooses between these two strategies--the latter being used only when

$$\left(\frac{NCALL}{2}\right)^{1/NDIM} > \frac{NDMX}{2} . \qquad [1]$$

The array sizes used in VEGAS can be modified, if desired. The program, as presented in the Appendix, will integrate over as many as ten variables. To increase (or decrease) this maximum dimension, change every 10 to the new maximum in each of the COMMON, DIMENSION and REAL statements. Similarly the maximum number of increments (NDMX) can be increased beyond 50 by replacing 50 in the COMMON/BVEG2/ and DIMENSION statements. Note however that increasing NDMX beyond 50 rarely results in significant gains in accuracy.

III. Interpreting VEGAS

When the algorithm is working well, $\bar{S} \pm \bar{\sigma}$ (i.e., AVGI±SD) is a reliable estimate of the integral (Eq. (7)). However in certain circumstances care is required in interpreting output from VEGAS. Among the more common situations calling for extra care are the following:

a) $|S_\alpha|$ grows steadily from iteration to iteration with the χ^2/dof increasing and much larger than one - If S_α grows by orders of magnitude and fails to level off, the integral is probably divergent. When the S_α tends to a finite number, the integral may be finite but not square integrable. In this last case, the S_α may tend to the exact answer, but the error estimates will be unreliable.

b) χ^2/dof (i.e., CHI2A, or 'CHI**2 PER IT'N') much larger than one - Different iterations are inconsistent, one with the other. Generally either NCALL must be increased or the integration variables transformed so as to smooth the integrand before VEGAS can be trusted. However, when the integrand has high narrow peaks, S_α and σ_α are sometimes badly underestimated in the earliest iterations, before the algorithm has adapted. If χ^2/dof is small when these iterations are omitted, then the estimate $\bar{S} \pm \bar{\sigma}$ as determined from the later iterations alone is reliable. VEGAS1 can be employed to omit the early iterations when determining $\bar{S}$, $\bar{\sigma}$ and χ^2 (see Section II.C).

c) Large oscillations in the grid spacings from iteration to iteration, frequently accompanied by large χ^2/dof - Upon occasion, VEGAS badly overcorrects the grid for a sharp narrow peak in the integrand. Then if it continues to overcorrect for the overcorrections, the grid can oscillate back and forth for several iterations. This problem can be

alleviated by reducing ALPH (Section II.F), thereby requiring VEGAS to be less precipitous in changing the grid, and further damping any oscillations which do occur.

d) The estimates S_α repeatedly oscillate in sign - This usually indicates that the integrand has two or more large cancelling peaks (i.e., $\int d^n x\ |f(\vec{x})| >> \int d^n x\ f(\vec{x})$). The situation is improved by increasing NCALL, or sometimes by transforming variables so that large cancelling peaks become small cancelling peaks (e.g., integration region can be folded over on itself so that much of the cancellation between peaks is local or point-wise). Needless to say, VEGAS cannot compute $\int d^n x\ f(\vec{x})$ when $\int d^n x\ |f(\vec{x})| = \infty$ (e.g., principal value integrations, ...).

Aside from these specific problems, there is the general problem of improving the estimate of the integral. Once the optimal grid has been found by VEGAS, accuracy improves as the square root of the number of integration points. So increasing NCALL or ITMX will increase the accuracy of the estimate. As mentioned above, another procedure which frequently helps is to change integration variables so as to smooth out the integrand. The grid information printed after each iteration can be used to locate peaks in the integrand and 'simple' transformations introduced to reduce them. For example, the replacement

$$\int_0^1 dx\ f(x) \rightarrow \int_0^1 dy \left.\frac{f(x)}{\beta x^{\beta-1}}\right|_{y=x^\beta} \qquad \beta > 1$$

may help when f(x) has a large peak at x = 1.

Appendix A. FORTRAN Listing of VEGAS

```
      BLOCK DATA
C
C   MAKES DEFAULT PARAMETER ASSIGNMENTS FOR VEGAS
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/BVEG1/NCALL,ITMX,NPRN,NDEV,XL(10),XU(10),ACC
      COMMON/BVEG2/IT,NDO,SI,SWGT,SCHI,XI(50,10)
      COMMON/BVEG3/ALPH,NDMX,MDS
      DATA NCALL/5000/,ITMX/5/,NPRN/5/,ACC/-1./,
     1     XL/0.,0.,0.,0.,0.,0.,0.,0.,0.,0./,
     2     XU/1.,1.,1.,1.,1.,1.,1.,1.,1.,1./,
     3     ALPH/1.5/,NDMX/50/,MDS/1/,NDEV/6/,
     4     NDO/1/,XI/500*1./,IT/0/,SI,SWGT,SCHI/3*0./
      END
      SUBROUTINE VEGAS(NDIM,FXN,AVGI,SD,CHI2A)
C
C   SUBROUTINE PERFORMS NDIM-DIMENSIONAL MONTE CARLO INTEG'N
C     - BY G.P. LEPAGE    SEPT 1976/(REV)AUG 1979
C     - ALGORITHM DESCRIBED IN J COMP PHYS 27,192(1978)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/BVEG1/NCALL,ITMX,NPRN,NDEV,XL(10),XU(10),ACC
      COMMON/BVEG2/IT,NDO,SI,SWGT,SCHI,XI(50,10)
      COMMON/BVEG3/ALPH,NDMX,MDS
      COMMON/BVEG4/CALLS,TI,TSI
      DIMENSION D(50,10),DI(50,10),XIN(50),R(50),DX(10),IA(10),
     1          KG(10),DT(10),X(10)
      REAL RAND(10)
      DATA ONE/1./
      SQRT(A)=DSQRT(A)
      ALOG(A)=DLOG(A)
      ABS(A)=DABS(A)
C
      NDO=1
      DO 1 J=1,NDIM
1     XI(1,J)=ONE
C
      ENTRY VEGAS1(NDIM,FXN,AVGI,SD,CHI2A)
C       - INITIALIZES CUMMULATIVE VARIABLES, BUT NOT GRID
      IT=0
      SI=0.
      SWGT=SI
      SCHI=SI
C
      ENTRY VEGAS2(NDIM,FXN,AVGI,SD,CHI2A)
C       - NO INITIALIZATION
      ND=NDMX
      NG=1
      IF(MDS.EQ.0) GO TO 2
      NG=(NCALL/2.)**(1./NDIM)
      MDS=1
      IF((2*NG-NDMX).LT.0) GO TO 2
      MDS=-1
```

```
      NPG=NG/NDMX+1
      ND=NG/NPG
      NG=NPG*ND
    2 K=NG**NDIM
      NPG=NCALL/K
      IF(NPG.LT.2) NPG=2
      CALLS=NPG*K
      DXG=ONE/NG
      DV2G=(CALLS*DXG**NDIM)**2/NPG/NPG/(NPG-ONE)
      XND=ND
      NDM=ND-1
      DXG=DXG*XND
      XJAC=ONE/CALLS
      DO 3 J=1,NDIM
      DX(J)=XU(J)-XL(J)
    3 XJAC=XJAC*DX(J)
C
C   REBIN, PRESERVING BIN DENSITY
      IF(ND.EQ.NDO) GO TO 8
      RC=NDO/XND
      DO 7 J=1,NDIM
      K=0
      XN=0.
      DR=XN
      I=K
    4 K=K+1
      DR=DR+ONE
      XO=XN
      XN=XI(K,J)
    5 IF(RC.GT.DR) GO TO 4
      I=I+1
      DR=DR-RC
      XIN(I)=XN-(XN-XO)*DR
      IF(I.LT.NDM) GO TO 5
      DO 6 I=1,NDM
    6 XI(I,J)=XIN(I)
    7 XI(ND,J)=ONE
      NDO=ND

      IF(NPRN.GE.0) WRITE(NDEV,200) NDIM,CALLS,IT,ITMX,ACC,NPRN,
     1                    ALPH,MDS,ND,(XL(J),XU(J),J=1,NDIM)

      ENTRY VEGAS3(NDIM,FXN,AVGI,SD,CHI2A)
          - MAIN INTEGRATION LOOP
      IT=IT+1
      TI=0.
      TSI=TI
      DO 10 J=1,NDIM
      KG(J)=1
      DO 10 I=1,ND
      D(I,J)=TI
   10 DI(I,J)=TI
```

```
C
11      FB=0.
        F2B=FB
        K=0
12      K=K+1
        CALL RANDA(NDIM,RAND)
        WGT=XJAC
        DO 15 J=1,NDIM
        XN=(KG(J)-RAND(J))*DXG+ONE
        IA(J)=XN
        IF(IA(J).GT.1) GO TO 13
        XO=XI(IA(J),J)
        RC=(XN-IA(J))*XO
        GO TO 14
13      XO=XI(IA(J),J)-XI(IA(J)-1,J)
        RC=XI(IA(J)-1,J)+(XN-IA(J))*XO
14      X(J)=XL(J)+RC*DX(J)
15      WGT=WGT*XO*XND
C
        F=WGT
        F=F*FXN(X,WGT)
        F2=F*F
        FB=FB+F
        F2B=F2B+F2
        DO 16 J=1,NDIM
        DI(IA(J),J)=DI(IA(J),J)+F
16      IF(MDS.GE.0) D(IA(J),J)=D(IA(J),J)+F2
        IF(K.LT.NPG) GO TO 12
C
        F2B=SQRT(F2B*NPG)
        F2B=(F2B-FB)*(F2B+FB)
        TI=TI+FB
        TSI=TSI+F2B
        IF(MDS.GE.0) GO TO 18
        DO 17 J=1,NDIM
17      D(IA(J),J)=D(IA(J),J)+F2B
18      K=NDIM
19      KG(K)=MOD(KG(K),NG)+1
        IF(KG(K).NE.1) GO TO 11
        K=K-1
        IF(K.GT.0) GO TO 19
C
C   COMPUTE FINAL RESULTS FOR THIS ITERATION
        TSI=TSI*DV2G
        TI2=TI*TI
        WGT=ONE/TSI
        SI=SI+TI*WGT
        SWGT=SWGT+WGT
        SCHI=SCHI+TI2*WGT
        AVGI=SI/SWGT
        CHI2A=(SCHI-SI*AVGI)/(IT-.9999)
        SD=SQRT(ONE/SWGT)
```

```
      IF(NPRN.LT.0) GO TO 21
      TSI=SQRT(TSI)
      WRITE(NDEV,201) IT,TI,TSI,AVGI,SD,CHI2A
      IF(NPRN.EQ.0) GO TO 21
      DO 20 J=1,NDIM
0     WRITE(NDEV,202) J,(XI(I,J),DI(I,J),I=1+NPRN/2,ND,NPRN)

    REFINE GRID
1     DO 23 J=1,NDIM
      XO=D(1,J)
      XN=D(2,J)
      D(1,J)=(XO+XN)/2.
      DT(J)=D(1,J)
      DO 22 I=2,NDM
      D(I,J)=XO+XN
      XO=XN
      XN=D(I+1,J)
      D(I,J)=(D(I,J)+XN)/3.
2     DT(J)=DT(J)+D(I,J)
      D(ND,J)=(XN+XO)/2.
3     DT(J)=DT(J)+D(ND,J)

      DO 28 J=1,NDIM
      RC=0.
      DO 24 I=1,ND
      R(I)=0.
      IF(D(I,J).LE.0.) GO TO 24
      XO=DT(J)/D(I,J)
      R(I)=((XO-ONE)/XO/ALOG(XO))**ALPH
      RC=RC+R(I)
      RC=RC/XND
      K=0
      XN=0.
      DR=XN
      I=K
      K=K+1
      DR=DR+R(K)
      XO=XN
      XN=XI(K,J)
      IF(RC.GT.DR) GO TO 25
      I=I+1
      DR=DR-RC
      XIN(I)=XN-(XN-XO)*DR/R(K)
      IF(I.LT.NDM) GO TO 26
      DO 27 I=1,NDM
      XI(I,J)=XIN(I)
      XI(ND,J)=ONE

      IF(IT.LT.ITMX.AND.ACC*ABS(AVGI).LT.SD) GO TO 9
0     FORMAT(/35H INPUT PARAMETERS FOR VEGAS:  NDIM=I3,8H  NCALL=F8.0
     1  /28X,5H  IT=I5,7H  ITMX=I5/28X,6H  ACC=G9.3
```

```
     2  /28X,7H  NPRN=I3,7H  ALPH=F5.2/28X,6H  MDS=I3,6H   ND=I4
     3  /28X,10H  (XL,XU)=(T40,2H( G12.6,3H , G12.6,2H )))
 201  FORMAT(///21H INTEGRATION BY VEGAS//14H ITERATION NO.I3,
     1  14H:   INTEGRAL =G14.8/21X,10HSTD DEV  =G10.4/
     2  34H ACCUMULATED RESULTS:   INTEGRAL =G14.8/
     3  24X,10HSTD DEV  =G10.4/24X,17HCHI**2 PER IT'N =G10.4)
 202  FORMAT(/15H DATA FOR AXIS I2/25H    X        DELTA I       ,
     1  24H    X        DELTA I       ,18H    X        DELTA I
     2  /(1H F7.6,1X,G11.4,5X,F7.6,1X,G11.4,5X,F7.6,1X,G11.4))
      RETURN
      END
      SUBROUTINE RANDA(N,RAND)
C
C    SUBROUTINE GENERATES UNIFORMLY DISTRIBUTED RANDOM NO'S X(I),I=1,N
      DIMENSION RAND(N)
      DO 1 I=1,N
 1    RAND(I)=RAN(1234567)
      RETURN
      END
```

Appendix B. An Example

In this Appendix, the use of VEGAS is illustrated in the computation of

$$I = \frac{100}{\pi} \int_0^1 dx_1 \int_{-1}^1 dx_2 \; e^{-100(x_1^2 + (x_2-1)^2)} = \frac{1}{4}$$

The integrand's subprogram is

```
      FUNCTION F(X,WGT)
      DOUBLE PRECISION X(2),WGT,F
      F = DEXP(-100.*(X(1)**2 + (X(2) - 1)**2))*100./3.141592654 D0
      RETURN
      END
```

The calling program is

```
      DOUBLE PRECISION XL,XU,ACC,F,AVGI,SD,CHI2A
      COMMON/BVEG1/NCALL,ITMX,NPRN,NDEV,XL(10),XU(10),ACC
      EXTERNAL F
      XL(2) = -1
      CALL VEGAS( 2,F,AVGI,SD,CHI2A)
      RETURN
      END
```

Most of the control parameters have their default values in this example. Thus VEGAS will estimate the integral in five iterations using approximately 5000 evaluations of the integrand per iteration (in fact 4802 are used per iteration). The output from VEGAS is presented in the succeeding pages. Since NPRN=5, information for only one out of every five axis subdivisions is printed - i.e., for only ten increments on each axis (while in fact VEGAS used ND=49 increments per axis). The results returned to the calling program were

```
AVGI = .249984
  SD = .000059
CHI2A = 1.116
```

and thus the integral is estimated to be

$.24998 \pm .00006$ with 68% confidence

$\pm .00012$ with 95% confidence

... etc.

As the χ^2/iteration is 1.1, the various iterations are consistent with each other and thus the error estimates are probably statistically meaningful.

```
INPUT PARAMETERS FOR VEGAS:  NDIM=  2  NCALL=   4802.
                             IT=    0  ITMX=    5
                             ACC=-.100D+01
                             NPRN=  5  ALPH= 1.50
                             MDS= -1   ND=  49
                             (XL,XU)= (  .000000D+00 ,  .100000D+01 )
                                      ( -.100000D+01 ,  .100000D+01 )

INTEGRATION BY VEGAS

ITERATION NO.   1:    INTEGRAL = .25088284D+00
                      STD DEV  = .4816D-02
ACCUMULATED RESULTS:    INTEGRAL = .25088284D+00
                        STD DEV  = .4816D-02
                        CHI**2 PER IT'N = .0000D+00

DATA FOR AXIS  1
    X        DELTA I          X        DELTA I          X        DELTA I
.020408   0.5580D-01      .122449   0.1762D-01      .224490   0.5239D-03
.326531   0.2957D-05      .428571   0.1479D-08      .530612   0.1171D-12
.632653   0.9579D-18      .734694   0.1026D-23      .836735   0.2292D-30
.938776   0.0000D+00

DATA FOR AXIS  2
    X        DELTA I          X        DELTA I          X        DELTA I
.020408   0.0000D+00      .122449   0.0000D+00      .224490   0.0000D+00
.326531   0.0000D+00      .428571   0.0000D+00      .530612   0.0000D+00
.632653   0.1537D-24      .734694   0.1428D-13      .836735   0.1191D-05
.938776   0.1399D-01

INTEGRATION BY VEGAS

ITERATION NO.   2:    INTEGRAL = .24977193D+00
                      STD DEV  = .1431D-03
ACCUMULATED RESULTS:    INTEGRAL = .24977291D+00
                        STD DEV  = .1430D-03
                        CHI**2 PER IT'N = .5315D-01

DATA FOR AXIS  1
    X        DELTA I          X        DELTA I          X        DELTA I
.000000   0.5575D-02      .011891   0.5515D-02      .022112   0.6176D-02
.034293   0.6129D-02      .047390   0.6566D-02      .062173   0.6374D-02
.087179   0.7377D-02      .115528   0.4869D-02      .155384   0.2700D-02
.249067   0.2836D-03

DATA FOR AXIS  2
    X        DELTA I          X        DELTA I          X        DELTA I
.221415   0.6393D-07      .902915   0.7042D-03      .924613   0.1246D-02
```

```
.935991   0.2344D-02      .945854   0.3144D-02      .955225   0.4611D-02
.964178   0.5683D-02      .972827   0.7116D-02      .982037   0.1098D-01
.993264   0.1232D-01


INTEGRATION BY VEGAS

ITERATION NO.   3:    INTEGRAL = .25013271D+00
                      STD DEV  = .1069D-03
ACCUMULATED RESULTS:    INTEGRAL = .25000378D+00
                        STD DEV  = .8561D-04
                        CHI**2 PER IT'N = .2057D+01

DATA FOR AXIS  1
    X         DELTA I          X         DELTA I          X         DELTA I
.002848   0.8037D-02      .017867   0.8188D-02      .033773   0.8344D-02
.051907   0.8650D-02      .071086   0.6884D-02      .090364   0.5588D-02
.112597   0.3986D-02      .134260   0.2216D-02      .162196   0.1434D-02
.212546   0.6073D-03

DATA FOR AXIS  2
    X         DELTA I          X         DELTA I          X         DELTA I
.865896   0.1949D-04      .917791   0.1738D-02      .936020   0.3219D-02
.949088   0.4346D-02      .959171   0.5128D-02      .967603   0.5491D-02
.975327   0.6674D-02      .982623   0.7356D-02      .989472   0.7211D-02
.995797   0.7105D-02


INTEGRATION BY VEGAS

ITERATION NO.   4:    INTEGRAL = .24992221D+00
                      STD DEV  = .1202D-03
ACCUMULATED RESULTS:    INTEGRAL = .24997633D+00
                        STD DEV  = .6974D-04
                        CHI**2 PER IT'N = .1473D+01

DATA FOR AXIS  1
    X         DELTA I          X         DELTA I          X         DELTA I
.002524   0.7292D-02      .015906   0.7751D-02      .029970   0.7797D-02
.045519   0.7724D-02      .061624   0.6324D-02      .078364   0.5279D-02
.096047   0.4412D-02      .121749   0.2811D-02      .152193   0.2274D-02
.201994   0.7764D-03

DATA FOR AXIS  2
    X         DELTA I          X         DELTA I          X         DELTA I
.734757   0.1670D-13      .911899   0.1029D-02      .929993   0.2262D-02
.948656   0.3888D-02      .954333   0.4589D-02      .963891   0.6192D-02
.972197   0.6027D-02      .979992   0.7292D-02      .987937   0.8022D-02
.995328   0.8316D-02
```

```
INTEGRATION BY VEGAS

ITERATION NO.   5:    INTEGRAL = .25000366D+00
                      STD DEV  = .1125D-03
ACCUMULATED RESULTS:    INTEGRAL = .24998392D+00
                        STD DEV  = .5927D-04
                        CHI**2 PER IT'N = .1116D+01

DATA FOR AXIS  1
   X        DELTA I          X        DELTA I          X        DELTA I
.003114   0.8786D-02      .015250   0.4685D-02      .026525   0.8191D-02
.042111   0.8995D-02      .060036   0.6663D-02      .077718   0.5409D-02
.098793   0.4723D-02      .123348   0.3467D-02      .154156   0.1942D-02
.208158   0.7767D-03

DATA FOR AXIS  2
   X        DELTA I          X        DELTA I          X        DELTA I
.330214   0.0000D+00      .893072   0.2844D-03      .923040   0.1913D-02
.939590   0.3392D-02      .952160   0.5016D-02      .962694   0.6426D-02
.971621   0.6587D-02      .980018   0.7600D-02      .988117   0.8616D-02
.995636   0.8450D-02
```