OFFSET REQUEST

Name Harriet Canfield

Group Title CG Phone 2574

Date March 24, '77

Remarks or special instructions

Number of originals 15

Number of copies of each 50

Obsolete
see CGTM 188

SLAC Computation Group
Stanford, California

CGTM NO. 184
March 1977

DIVONNE

A Program for Multiple Integration and Adaptive Importance Sampling

Jerome H. Friedman

DIVONNE

DIVONNE numerically integrates a bounded function of several variables over a finite region. An integral over an infinite region can be transformed to one over a finite region. It also allows random sampling in multidimensional coordinate spaces with sample points approximately distributed according to a user specified density that need not be normalized.

Structure

Subroutine subprogram

User Entry Names: DIVONE, FUN, USRTRM, FUNINT, GENPNT, TREDMP, BUKDMP

Files Referenced: OUTPUT, TAPE NFILE: optional user defined external file (see below)

External Routines Referenced: ARAN9(X,N)

Common Blocks: /DATE/, /PRINT/, /ISTRGE/, /RSTRGE/, /LIMITS/, /SOLSVE/, /STRTGY/, /QUADRE/, /START/, /EXFILE/, /DISPOS/, /DEPTHS/, /MXMZER/, /SAMPLE/, /SPCNTL/, /BXCNTL/, /MLIMIT/, /TRESZE/, /EXMCTL/, /BUKSZE/, /GENINL/, /MINSPL/, /FUNN/, /ANSWER/, /MAXERR/, /SIGSPL/, /STRAND/, /WITCH/

Usage

In this write-up, only the more straightforward uses are discussed. More sophisticated applications are discussed in the long write-up.

The integrand is defined by a user coded function subprogram:

```
FUNCTION FUN(D,X)
Integer D
Real X(D)
  .
  .
  .
FUN=
  .
  .
  .
RETURN
END
```

Both arguments are input to the subprogram. The first, D, is the number of integration variables (dimensionality of integration space) and the second is a vector containing the coordinates of a point in the integration volume. The function subprogram must return (in FUN) the value of the integrand for the coordinate values stored in X.

The integration is invoked by a subroutine call

```
CALL DIVONE (ND, ERROR, MAXPTS)
```

All arguments are input to the subroutine and must be specified by the user.

ND = number of integration variables
(dimensionality of integration space)

ERROR = accuracy desired for convergence

If ERROR $\geq$ 0, then $\frac{\Delta I}{ABS(I)} \leq$ ERROR defines convergence

If ERROR $<$ 0, then $\Delta I \leq |ERROR|$ defines convergence

MAXPTS = maximum number of integrand evaluations. Control is returned to the calling program at the end of the first iteration for which the number of integrand evaluations (calls to FUN) exceed MAXPTS.

For multiple integration, the procedure described above is the minimum required for invocation of the program. For random sampling according to a user provided density, one proceeds identically as above with the user provided density coded in the function subroutine FUN. In addition, the labeled common

```
COMMON /SOLSVE/ SVESOL
LOGICAL SVESOL
```

must be declared in the calling program, and the statement

```
SVESOL = .TRUE.
```

must be executed before the call to DIVONE.

After control returns from DIVONE, repeated subroutine calls

```
CALL GENPNT(ND,X,WT)
```

will each store the ND coordinates of a new random point in the vector X, and a corresponding weight in WT. Applying this weight with the point X, results in distributions that are statistically equivalent to a sample of points distributed with the density coded in FUN. The user density (as coded in FUN) need not be normalized. However, it must be bounded within the region of generation.

The above procedure assumes that the region of integration (or generation) is enclosed by the ND dimensional unit hypercube. If this is not the case, the user can declare the labeled common

```
COMMON /LIMITS/ XMINUS(MAXDIMEN), XPLUS(MAXDIMEN)
```

and store the lower and upper limit of each integration variable in the corresponding location of XPLUS and XMINUS. MAXDIMEN is the maximum allowed dimensionality defined when the DIVONNE package is compiled (currently MAXDIMEN=10).

Method

The technique used is an adaptive stratified random sampling. The d-dimensional hyperrectangle containing the integration region is divided into mutually exclusive subregions. Within each subregion, R_i, an approximation $g_i(\vec{x})$ is made to the integrand function $f(\vec{x})$. This approximation is used as a control variate in a Monte Carlo integration of $f(\vec{x})$ within the region. That is,

$$F_i = \int_{R_i} f(\vec{x})\, d\vec{x} = \int_{R_i} [f(\vec{x}) - g_i(\vec{x})]\, dx + G_i$$

where $G_i = \int_{R_i} g_i(\vec{x})\, d\vec{x}$ is known exactly.

An estimate of F_i $(\hat{F}_i)$ is obtained by sampling n points randomly within the region and taking

$$\hat{F}_i = \frac{V_i}{n} \sum_{j=1}^{n} [f(\vec{x}_j) - g_i(\vec{x}_j)] + G_i$$

with an associated uncertainty

$$(\delta\hat{F}_i)^2 = \frac{V_i^2}{n} \sum_{j=1}^{n} [f(\vec{x}_j) - g_i(\vec{x}_j)]^2 - (\hat{F}_i - G_i)^2 \quad .$$

V_i is the volume of the ith subregion. The integral and its associated uncertainty estimate over the entire region are taken as the sum over all of the M- subregions

$$\hat{F} = \sum_{i=1}^{M} \hat{F}_i \quad \text{and} \quad (\delta\hat{F})^2 = \sum_{i=1}^{M} (\delta\hat{F}_i)^2 \quad .$$

The form of the approximation function $g_i(\vec{x})$ within each region is specified

by the user from two choices:

$$\text{constant:} \qquad g_i(\vec{x}) = b_i$$

$$\text{linear:} \qquad g_i(\vec{x}) = \vec{a}_i \cdot \vec{x} + b_i \ .$$

The values of the parameters $(\vec{a}_i, b)$ are the least-squares solutions to the set of linear equations

$$f(\vec{x}_j) = g_i(\vec{x}_j) \qquad\qquad 1 \le j \le n \ .$$

The n values $\vec{x}_j$ are chosen randomly within the subregion. The resulting approximation to the integrand over the entire region of integration is then either piecewise constant or piecewise linear.

A crucial aspect of this procedure is the choice of the particular subregions. That is, how to divide the region of integration into subregions so as to yield the best possible piecewise linear (or constant) approximation, for as few subregions (and hence, as few integrand evaluations) as possible. The method utilized in DIVONNE is recursive or nested partitioning. At each stage (recursive invocation), one has a particular subregion of the region of integration. (The subregion for the first invocation is the entire region of integration.) For this subregion, the approximation function $g_i(\vec{x})$, the estimate of the subintegral $\hat{F}_i$, and its associated uncertainty $\delta\hat{F}_i$, are determined. The values of $\hat{F}_i$ and $(\delta\hat{F}_i)^2$ are used to update the current estimate of $\hat{F}$ and $(\delta\hat{F})^2$. If $\delta\hat{F}_i$ is small enough, the subregion becomes a terminal subregion and is not processed further. If not, the subregion is divided into two daughter subregions by a d-1 dimensional hyperplane parallel to one of the coordinate axes. The object is to divide the parent region into two subregions such that the residual mean squared error (sum of variances of the two daughters $(\delta F_+)^2 + (\delta F_-)^2$) is as small as possible. The recursive procedure (described above) is then applied to each daughter subregion.

The result of applying the recursive partitioning procedure is a set of nested, mutually exclusive subregions of the full integration volume. These subregions are collectively oriented so that within each (ith) the integrand $f(\vec{x})$ can be closely approximated by its corresponding control function $g_i(\vec{x})$. Associated with each subregion is the estimated value of the integral, $\hat{F}_i$, and an associated uncertainty estimate, $(\delta F_i)^2$. These are summed incoherently to form the global estimate of the integral and uncertainty. The partitioning stops when the uncertainty estimate is below that specified by the user, or when the number of integrand evaluations exceeds the user specified maximum.

A problem with the recursive partitioning strategy is that convergence to the integral value is not continuous or uniform. Each subregion is recursively partitioned until the desired accuracy is reached within it. This is no problem if convergence is reached. However, if the program is terminated before convergence (by exceeding the maximum number of integrand evaluations), then it is desirable that the resulting uncertainty ($\delta\hat{F}$) represent the best answer that is obtainable with the method for the specified number of integrand evaluations. This will not be the case for recursive partitioning because the strategy always chooses for the next partitioning, the current subregion (until it is good enough) rather than the subregion with the largest uncertainty ($\delta\hat{F}_i$) in the whole integration region. This can be remedied by implementing a global <u>iterative</u> strategy rather than the local recursive strategy; that is, one chooses the subregion with the largest error estimate as the next candidate for partitioning. This procedure causes increased overhead computation but has the desired convergence properties. The compromise chosen here is to apply the recursive strategy iteratively. That is, at each iteration the recursive strategy is applied to the subregion with the largest error estimate. The recursive partitioning of that region proceeds until each of its daughter sub-

regions has an estimated uncertainty less than the one with the second largest uncertainty, or until a maximum number of partitions have been made. At that time, the new subregions are merged with the others and another iteration begins.

For any particular subregion, the procedure is effective only if the randomly drawn itegrand evaluation points adequately sample the variation of the integrand within the region. For a highly structured function in high dimensionality, this may not be the case. This is a fundamental shortcoming of all numerical integration methods, quadrature or Monte Carlo, adaptive or not. This is a problem for adaptive procedures because convergence is based on the error estimate which is strongly based toward low values with inadequate sampling. For example, if the integrand is relatively smooth except for a few narrow but high peaks, it is quite likely that the sampling will not detect the peaks, especially for problems of high dimensionality. The procedure will then converge to a small apparent uncertainty, but with an incorrect value for the integral.

To help overcome this problem, the recursive partitioning strategy is augmented (at the option of the user) with an optimization strategy. In addition to the random sampling within an (ith) subregion at each partitioning stage, the maximum of the function $D_i(\vec{x}) = |f(\vec{x}) - g_i(\vec{x})|$ is found within the subregion. That is,

$$D_i(\vec{x}^*) = \max_{\vec{x}} |f(\vec{x}) - g_i(\vec{x})| \ .$$

The value of $D_i(\vec{x}^*)$ is compared with the root mean squared variation of $f(\vec{x}) - g_i(\vec{x})$ as determined from the random sampling. If the ratio of these two quantities is below a prespecified value (say 20), then the random sampling is considered to be adequate and the usual partitioning strategy is invoked.

If not, a special strategy designed to handle these anomolous situations is invoked.

The advantage of employing the optimization strategy is the strong robustness of its error estimate against badly behaved integrands. It can seldom be fooled into thinking that it has an accurate estimate of the integral when it does not. The price paid for this assurance is increased number of integrand evaluations (usually 20% to 30%) and an upward bias of the error estimate (usually a factor of 1.5 to 2) for those cases when the integrand happens to be well behaved.

Random sampling according to a user provided density function proceeds as above with the density function as the integrand. The result is a collection of subregions and a separate linear (or constant) approximation to the density function within each. The sampling is accomplished by choosing a subregion randomly (with probability proportional to the estimated integral within the region) and then using the approximation function as a density for sampling within the subregion. Weighting the sample points so obtained by $f(\vec{x})/g_i(\vec{x})$ results in a distribution that is statistically equivalent to sampling directly from $f(\vec{x})$ with only a slight increase in variance.

Additional Options (Bells and Whistles)

Although the minimum necessary to use the program has been described above, there are several user options that enhance the flexibility and usefulness of the program. These options are invoked by changing the value of an internal parameter stored in a labeled common. Default values for these parameters are set via a BLOCK DATA subprogram internal to the DIVONNE package at compilation time. They can be changed by declaring the appropriate labeled common in a user routine and reset via an executable statement before executing the call to DIVONE.

Algorithmic Options

COMMON/STRTGY/LEVEL

This parameter sets the basic approximation strategy for the recursive partitioning.

LEVEL = 1 : piecewise constant approximation, no optimization

LEVEL = 2 : piecewise constant approximation with optimization

LEVEL = 3 : piecewise linear approximation, no optimization

LEVEL = 4 : piecewise linear approximation with optimization

default = 3

COMMON/SAMPLE/NPOINT

Controls the number of random points sampled in each subregion.

$$2 \leq \text{NPOINT} \leq \text{MAXPOINTS}$$

where MAXPOINTS is the maximum number defined when the DIVONNE package is compiled (currently = 100).

default = 50

COMMON/QUADRE/IDEG

In addition to the control variate Monte Carlo estimate of the integral within each subregion, numerical quadrature formulas may be applied. A numerical quadrature formula of degree IDEG will yield the exact result if the integrand can be exactly represented by a polynomial of degree IDEG within the region.

IDEG = 0 : no numerical quadrature

IDEG = 2 : 2nd degree quadrature within each subregion

IDEG = 3 : 2nd and 3rd degree quadrature within each subregion

IDEG = 5 : 2nd, 3rd and 5th degree quadrature within each region

default = 0

Control Options

COMMON/PRINT/IPRINT

Controls printed output. Iteration information is printed when the iteration number is an exact multiple of IPRINT and upon termination.

deafult = 1 (maximum printing)

COMMON/EXFILE/NFILE

Defines FORTRAN file number (TAPE NFILE) for external stroage to save intermediate results between runs (see below).

default = 1

COMMON/DISPOS/IDISP

Controls the storage on an external file of information necessary to continue iterating with a future job (or job step) at the last iteration of the present one.

IDISP = 0 : no action

IDISP ≠ 0 just before DIVONE returns control to calling program, TAPE NFILE is rewound and the information is written onto TAPE NFILE for continuing the algorithm at that point at which it left off.

default = o.

COMMON/START/ISTART

Defines initialization state at time of call to DIVONE.

ISTART = 1 : new integration, begin partitioning

ISTART = 2 : subsequent call, continue iterating from last iteration of previous call to DIVONE in this job step. This value (ISTART=2) is not valid for the first call to DIVONE within a job step

ISTART = 3 : continue iterating from last iteration of previous job (or job step) by reading information from TAPE NFILE

default = 1

Output

During the execution of the program, the user can have access to intermediate results by declaring the appropriate labeled commons. These values are the current estimates of the various quantities and are updated with each iteration. For the correct functioning of the algorithm, these quantities should never be reset by the user.

COMMON/FUNN/NFUN

NFUN = number of integrand evaluations

COMMON/ANSWER/NRGN,FINTGL,ERROR,ERRMAX,MAXRGN,QUAD2,
QUAD3,QUAD5

NRGN = number of subregions
FINTGL = control variate Monte Carlo estimate of integral
ERROR = associated uncertainty estimate
ERRMAX = largest subregion uncertainty
MAXRGN = subregion number with largest uncertainty
QUAD2 = 2nd degree quadrature estimate of integral
QUAD3 = 3rd degree quadrature estimate of integral
QUAD5 = 5th degree quadrature estimate of integral

The quadrature estimates only exist if IDEG has been set appropriately before the call to DIVONNE.

User Defined Termination

When invoked, DIVONNE returns control to the calling program only if the specified accuracy has been reached or the specified number of integrand evaluations have been exceeded. Based on his own logic and the values of the various output quantities defined above (for example, consistency among the various quadrature formulae and the Monte Carlo estimate), the user may wish to define other convergence criteria. This can be done by coding a user defined function subroutine.

```
LOGICAL FUNCTION USRTRM (ITER)
   .
   .
   .
Return
End
```

The argument is input to the routine and contains, as a value, the iteration number. Returning a value .FALSE. for USRTRM will cause continued iteration subject to the usual convergence criteria. Returning a value of .TRUE. will cause immediate termination of the iteration procedure.

Auxiliary Integration

It often happens that one needs to evaluate several integrals for which the integrands and regions of integration are quite similar. For these cases, it is efficient to perform the recursive partitioning forming a piecewise linear (or constant) approximation only once for one of the integrands. This approximation can then be used repeatedly as a control variate from the other similar integrations.

This is accomplished by first invoking DIVONE in the normal manner with

the nominal (training) integrand. After this, repeated execution of the statement

```
FINT = FUNINT(ND,IDEG,ERROR)
```

will cause an integration to take place using the previously defined terminal subregions and associated control functions. The integrand for each integration is obtained from the user coded function subroutine FUN in the same manner as with DIVONE. Therefore, if the integrand is to change between calls, there must be user coded logic in both the calling program and in FUN to ensure that the correct integrand is evaluated. User communication between the calling program and FUN can be accomplished via labeled commons.

The quantities defined above have the following meanings:

input:

ND:	number of integration variables as above
IDEG:	signals type of integration strategy
IDEG=0,1:	control variate Monte Carlo integration

For this case, the labeled common

```
COMMON/SOLSVE/SVESOL
LOGICAL SVESOL
```

must be declared in the calling program and the statement

```
SVESOL = .TRUE.
```

must be executed before the call to DIVONE.

IDEG=0:	causes the random number generator to be re-initialized before the integration is performed
IDEG=1:	does not.

For the case of LEVEL=1 or LEVEL=3 (no optimization), the invocation of FUNINT causes the integral and error estimate for each terminal subregion to be reset internally to the value determined by FUNINT. This allows one to use FUNINT as part of a strategy to help guard

against the effects of inadequate sampling in one or several terminal regions, as follows: after the control returns from DIVONE, FUNINT is invoked (one or several times) with IDEG=1 to re-perform the same integration, but with different random samplings. If the results from FUNDIM and DIVONE are consistent, one has some confidence in the result. If not, DIVONE can be re-invoked for further partitioning. However, since the error estimates in the subregions have been re-evaluated, the partitioning sequence will be altered, increasing chances for convergence to the correct result.

IDEG=2:	Integral performed with 2nd degree quadrature formula in each subregion.
IDEG=3:	Integral performed with 3rd degree quadrature formula in each subregion.
IDEG=5:	Integral performed with 5th degree quadrature formula in each subregion.

Output:

FINT:	stores value of estimated integral
ERROR:	associate error estimate (meaningful for IDEG=0,1 only)

ACKNOWLEDGMENT

The author wishes to thank David Carey and Fred James for helpful suggestions