Package 'R2Cuba'

February 19, 2015

Title Multidimensional Numerical Integration

Version 1.0-11

Date 2013-04-15

Author The Cuba library has been written by Thomas Hahn (http://wwwth.mppmu.mpg.de/members/hahn); Interface to R was written by Annie Bouvier and Kiên Kiêu

Maintainer Annie Bouvier <Annie.Bouvier@jouy.inra.fr>

Depends methods

Suggests deldir

Description R2Cuba implements four general-purpose multidimensional integration algorithms: Vegas, Suave, Divonne and Cuhre

License GPL (>= 3)

Encoding latin1

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-04-15 17:47:33

R topics documented:

R2Cuba-package *Multidimensional Numerical Integration*

Description

R2Cuba implements four general-purpose multidimensional integration algorithms: Vegas, Suave, Divonne and Cuhre. It is a wrapper around the Cuba-1.6 library by Thomas Hahn available from the URL http://www.feynarts.de/cuba/.

Details

Package:	R2Cuba
Depends:	methods
Suggests:	deldir
License:	GPL (>= 3)

Author(s)

The Cuba library has been written by Thomas Hahn (http://wwwth.mppmu.mpg.de/members/hahn). Interface to R was written by Annie Bouvier and Kiên Kiêu (MIA Lab, INRA, Jouy-en-Josas, France http://www.jouy.inra.fr/mia_eng/).

Maintainer: Annie Bouvier <Annie.Bouvier@jouy.inra.fr>

References

The Cuba library is described at http://www.feynarts.de/cuba/. User documentation is available in *T. Hahn (2005) CUBA-a library for multidimensional numerical integration.* Computer Physics Communications, ***168***, *78-95.* (http://arxiv.org/pdf/hep-ph/0404043).

See Also

The R-package "cubature"

Examples

```
integrand <- function(arg, weight) {
  x <- arg[1]
  y <- arg[2]
  z <- arg[3]
  ff <- sin(x)*cos(y)*exp(z);
return(ff)
} # end integrand
NDIM <-3
NCOMP <- 1
```

```
vegas(NDIM, NCOMP, integrand, rel.tol=1e-3,  abs.tol=1e-12)
```

---

cuhre *Multidimensional numerical integration with a deterministic iterative adaptive algorithm*

Description

Implement a deterministic algorithm for multidimensional numerical integration. Its algorithm uses one of several cubature rules in a globally adaptive subdivision scheme. The subdivision algorithm is similar to suave's.

Usage

```
cuhre(ndim, ncomp, integrand, ...,
     lower=rep(0,ndim), upper=rep(1,ndim),
     rel.tol= 0.001,  abs.tol = 0,
     flags=list(verbose=1, final=1, pseudo.random=0, mersenne.seed=NULL),
      min.eval=0,  max.eval=50000,  key=0)
```

Arguments

ndim	the number of dimensions of the integral. It should be less or equal to 40.
ncomp	the number of components of the integrand. It should be less or equal to 10.
integrand	the R function which computes the integrand. It is expected to be declared as `integrand <- function(x, ...)` or `integrand <- function(x, phw, ...)` where `x` is an input vector of length `ndim`, and `phw` an ignored argument for compatibility with the other 'R2Cuba' functions. `...` denotes optional additional arguments which correspond to those passed to the main function in "...". The value returned by this R function should be a vector of length `ncomp`.
...	optional additional parameters to be passed to `integrand`, if any
lower	the lower bounds of the integration region. Vector of length `ndim`
upper	the upper bounds of the integration region. Vector of length `ndim`
rel.tol	the requested relative accuracy. Default, 0.001.
abs.tol	the requested absolute accuracy. The algorithm stops when either the relative or the absolute accuracies are met. Default 0 (the algorithm stops when the relative accuracy is met).
flags	flags governing the integration. A list with components: - `verbose`: `verbose` encodes the verbosity level, from. 0 to 3. Level 0 does not print any output, level 1 prints "reasonable" information on the progress of the integration, level 2 also echoes the input parameters, and level 3 further prints the subregion results.

- final: when 0, all sets of samples collected on a subregion during the various iterations or phases contribute to the final result. When 1, only the last (largest) set of samples is used in the final result.

- pseudo.random: (*ignored in* cuhre)

when 0, Sobol quasi-random numbers are used for sampling. When 1, Mersenne Twister pseudo-random numbers are used for sampling.

- mersenne.seed: (*ignored in* cuhre)

the seed for the Mersenne Twister algorithm, when pseudo.random=1 and when it would be explicitly set.

`min.eval` the minimum number of integrand evaluations required.

`max.eval` the (approximate) maximum number of integrand evaluations allowed.

`key` chooses the basic integration rule:

key = 7, 9, 11, 13 selects the cubature rule of degree key. Note that the degree-11 rule is available only in 3 dimensions, the degree-13 rule only in 2 dimensions.

For other values, the default rule is taken, which is the degree-13 rule in 2 dimensions, the degree-11 rule in 3 dimensions, and the degree-9 rule otherwise.

Details

See details in the documentation.

Value

A list of the S3-class cuba with components:

`method` here, "cuhre"

`nregions` the actual number of subregions needed.

`neval` the actual number of integrand evaluations needed.

`ifail` an error flag:

ifail = 0 , the desired accuracy was reached,

ifail = -1, dimension out of range,

ifail = 1, the accuracy goal was not met within the allowed maximum number of integrand evaluations.

`value` vector of length ncomp; the integral of integrand over the hypercube.

`abs.error` vector of length ncomp; the presumed absolute error of value.

`prob` vector of length ncomp; the

$$\chi^2$$

-probability (not the

$$\chi^2$$

-value itself!) that abs.error is not a reliable estimate of the true integration error.

`message` "OK" or a character string giving the error message.

`call` The matched call.

References

J. Berntsen, T. O. Espelid (1991) An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions on Mathematical Software*, **17**(4), 437-451.

T. Hahn (2005) CUBA-a library for multidimensional numerical integration. *Computer Physics Communications*, **168**, 78-95.

See Also

vegas, suave, divonne

Examples

```
integrand <- function(arg) {
  x <- arg[1]
  y <- arg[2]
  z <- arg[3]
  ff <- sin(x)*cos(y)*exp(z);
return(ff)
} # End integrand

NDIM <- 3
NCOMP <- 1
cuhre(NDIM, NCOMP,  integrand,
             rel.tol= 1e-3, abs.tol= 1e-12,
             flags= list(verbose=2, final=0))
```

---

divonne *Multidimensional numerical integration by stratified sampling for variance reduction*

Description

Divonne works by stratified sampling, where the partioning of the integration region is aided by methods from numerical optimization.

Usage

```
divonne(ndim, ncomp, integrand, ...,
        lower=rep(0,ndim), upper=rep(1,ndim),
        rel.tol= 0.001,   abs.tol = 0,
        flags=list(verbose=1, final=1, pseudo.random=0, mersenne.seed=NULL),
        min.eval=0,  max.eval=50000,
        key1=47,  key2=1, key3=1,
        max.pass=5, border=0, max.chisq=10,
        min.deviation=0.25,
        xgiven=NULL,  nextra=0, peakfinder=NULL)
```

Arguments

ndim	same as cuhre
ncomp	same as cuhre
integrand	same as cuhre. But, here, the input argument phw indicates the integration phase: 0: sampling of the points in xgiven, 1: partitioning phase, 2: final integration phase, 3: refinement phase. This information might be useful if the integrand takes long to compute and a sufficiently accurate approximation of the integrand is available. The actual value of the integrand is only of minor importance in the partitioning phase, which is instead much more dependent on the peak structure of the integrand to find an appropriate tessellation. An approximation which reproduces the peak structure while leaving out the fine details might hence be a perfectly viable and much faster substitute when phw < 2. In all other instances, phw can be ignored.
...	same as cuhre
lower	same as cuhre
upper	same as cuhre
rel.tol	same as cuhre
abs.tol	same as cuhre
flags	same as cuhre pseudo.random and mersenne.seed are only taken into account when the argument key1 is negative.
min.eval	same as cuhre
max.eval	same as cuhre
key1	integer that determines sampling in the partitioning phase: key1 = 7, 9, 11, 13 selects the cubature rule of degree key1. Note that the degree-11 rule is available only in 3 dimensions, the degree-13 rule only in 2 dimensions. For other values of key1, a quasi-random sample of $n = \|key1\|$ points is used, where the sign of key1 determines the type of sample, key1 = 0, use the default rule. key1 > 0, use a Korobov quasi-random sample, key1 < 0, use a "standard" sample (a Mersenne Twister pseudo-random sample if flags$pseudo.random=1, otherwise a Sobol quasi-random sample).
key2	integer that determines sampling in the final integration phase: same as key1, but here $$n = \|key2\|$$ determines the number of points, $n > 39$, sample n points, $n < 40$, sample n nneed points, where nneed is the number of points needed to reach the prescribed accuracy, as estimated by Divonne from the results of the partitioning phase.

`key3` integer that sets the strategy for the refinement phase:

`key3 = 0`, do not treat the subregion any further.

`key3 = 1`, split the subregion up once more.

Otherwise, the subregion is sampled a third time with `key3` specifying the sampling parameters exactly as `key2` above.

`max.pass` integer that controls the thoroughness of the partitioning phase: The partitioning phase terminates when the estimated total number of integrand evaluations (partitioning plus final integration) does not decrease for `max.pass` successive iterations.

A decrease in points generally indicates that Divonne discovered new structures of the integrand and was able to find a more effective partitioning. `max.pass` can be understood as the number of "safety" iterations that are performed before the partition is accepted as final and counting consequently restarts at zero whenever new structures are found.

`border` the relative width of the border of the integration region. Points falling into the border region will not be sampled directly, but will be extrapolated from two samples from the interior. Use a non-zero `border` if the integrand subroutine cannot produce values directly on the integration boundary. The relative width of the border is identical in all the dimensions. For example, set `border=0.1` for a border of width equal to 10% of the width of the integration region.

`max.chisq` the maximum

$$\chi^2$$

value a single subregion is allowed to have in the final integration phase. Regions which fail this

$$\chi^2$$

test and whose sample averages differ by more than `min.deviation` move on to the refinement phase.

`min.deviation` a bound, given as the fraction of the requested error of the entire integral, which determines whether it is worthwhile further examining a region that failed the

$$\chi^2$$

test. Only if the two sampling averages obtained for the region differ by more than this bound is the region further treated.

`xgiven` a matrix (`ndim`, `ngiven`). A list of `ngiven` points where the integrand might have peaks.

Divonne will consider these points when partitioning the integration region. The idea here is to help the integrator find the extrema of the integrand in the presence of very narrow peaks. Even if only the approximate location of such peaks is known, this can considerably speed up convergence.

`nextra` the maximum number of extra points the peak-finder subroutine will return. If `nextra` is zero, `peakfinder` is not called and an arbitrary object may be passed in its place, e.g. just 0.

`peakfinder` the peak-finder subroutine. This R function is called whenever a region is up for subdivision and is supposed to point out possible peaks lying in the region, thus

acting as the dynamic counterpart of the static list of points supplied in `xgiven`. It is expected to be declared as

`peakfinder <- function(bounds)`

where `bounds` is a matrix of dimension `(ndim, 2)` which contains the upper and lower bounds of the subregion. The names of the columns are `c("lower", "upper")`.

The returned value should be a matrix `(ndim, nx)` where `nx` is the actual number of points (should be less or equal to `nextra`).

Details

Divonne uses stratified sampling for variance reduction, that is, it partitions the integration region such that all subregions have an approximately equal value of a quantity called the spread (volume times half-range).

See details in the documentation.

Value

Idem as cuhre. Here `ifail` may be `>1` when the accuracy goal was not met within the allowed maximum number of integrand evaluations. Divonne can estimate the number of points by which `maxeval` needs to be increased to reach the desired accuracy and returns this value.

References

J. H. Friedman, M. H. Wright (1981) A nested partitioning procedure for numerical multiple integration. *ACM Trans. Math. Software*, **7**(1), 76-92.

J. H. Friedman, M. H. Wright (1981) User's guide for DIVONNE. SLAC Report CGTM-193-REV, CGTM-193, Stanford University.

T. Hahn (2005) CUBA-a library for multidimensional numerical integration. *Computer Physics Communications*, **168**, 78-95.

See Also

cuhre, suave, vegas

Examples

```
NDIM <- 3
NCOMP <- 1
integrand <- function(arg, phase) {
  x <- arg[1]
  y <- arg[2]
  z <- arg[3]
  ff <- sin(x)*cos(y)*exp(z);
return(ff)
}
divonne(NDIM, NCOMP, integrand, rel.tol=1e-3,  abs.tol=1e-12,
        flags=list(verbose=2),  key1= 47)

# Example with a peak-finder function
```

```
NMAX <- 4

peakf <- function(bounds) {
#  print(bounds) # matrix (ndim,2)
  x <- matrix(0, ncol=NMAX, nrow=NDIM)
   pas <- 1/(NMAX-1)
   # 1ier point
   x[,1] <- rep(0, NDIM)
   # Les autres points
   for (i in 2:NMAX) {
      x[,i] <- x[,(i-1)] + pas
    }
  return(x)
} #end peakf

divonne(NDIM, NCOMP, integrand,
               flags=list(verbose=0) ,
                peakfinder=peakf, nextra=NMAX)
```

`print.cuba` *Method print for package R2Cuba*

Description

Print an object of class 'cuba', i.e an output of the functions of this package.

Usage

```
## S3 method for class 'cuba'
print(x, ...)
```

Arguments

`x`	an object of class 'cuba'
`...`	arguments passed to the default function 'format'

Value

No return

See Also

`format`, `R2Cuba-package`

suave *Multidimensional numerical integration with SUbregion-Adaptive Vegas algorithm*

Description

Suave uses vegas-like importance sampling combined with a globally adaptive subdivision strategy: Until the requested accuracy is reached, the region with the largest error at the time is bisected in the dimension in which the fluctuations of the integrand are reduced most. The number of new samples in each half is prorated for the fluctuation in that half.

Usage

```
suave(ndim, ncomp, integrand, ...,
      lower=rep(0,ndim), upper=rep(1,ndim),
      rel.tol= 0.001, abs.tol = 0,
     flags=list(verbose=1, final=1, pseudo.random=0, smooth=0, mersenne.seed=NULL),
      min.eval=0,  max.eval=50000,
      nnew=1000, flatness= 50)
```

Arguments

ndim	same as cuhre
ncomp	same as cuhre
integrand	same as cuhre; But, here, the input argument phw contains the weight of the point being sampled. This extra value can safely be ignored.
...	same as cuhre
lower	same as cuhre
upper	same as cuhre
rel.tol	same as cuhre
abs.tol	same as cuhre
flags	same as cuhre. flags may have an additional component: smooth. When = 0, apply additional smoothing to the importance function, this moderately improves convergence for many integrands. When = 1 , use the importance function without smoothing, this should be chosen if the integrand has sharp edges.
min.eval	same as cuhre
max.eval	same as cuhre
nnew	the number of new integrand evaluations in each subdivision.
flatness	This parameter determines how prominently "outliers", i.e. individual samples with a large fluctuation, figure in the total fluctuation, which in turn determines how a region is split up. As suggested by its name, flatness should be chosen large for "flat" integrands and small for "volatile" integrands with high peaks.

Note that since `flatness` appears in the exponent, one should not use too large values (say, no more than a few hundred) lest terms be truncated internally to prevent overflow. More details about this parameter can be found Hahn's paper from 2005 and in Cuba documentation.

Details

See details in the documentation.dériv

Value

Idem as `cuhre`

References

T. Hahn (2005) CUBA-a library for multidimensional numerical integration. *Computer Physics Communications*, **168**, 78-95.

See Also

`cuhre`, `divonne`, `vegas`

Examples

```
integrand <- function(arg, weight) {
  x <- arg[1]
  y <- arg[2]
  z <- arg[3]
  ff <- sin(x)*cos(y)*exp(z);
return(ff)
} # end integrand
suave(3, 1, integrand, rel.tol=1e-3,  abs.tol=1e-12,
             flags=list(verbose=2, final=0))
```

---

vegas *Multidimensional numerical integration with a Monte Carlo algorithm*

Description

Implement a Monte Carlo algorithm for multidimensional numerical integration. This algorithm uses importance sampling as a variance-reduction technique. Vegas iteratively builds up a piecewise constant weight function, represented on a rectangular grid. Each iteration consists of a sampling step followed by a refinement of the grid.

Usage

```
vegas(ndim, ncomp, integrand, ...,
      lower=rep(0,ndim), upper=rep(1,ndim),
      rel.tol= 0.001,  abs.tol = 0,
     flags=list(verbose=1, final=1, pseudo.random=0, smooth=0, mersenne.seed=NULL),
      min.eval=0,  max.eval=50000,
      nstart=1000,  nincrease=500, nbatch=1000, gridno=0, state.file=NULL)
```

Arguments

ndim	same as cuhre
ncomp	same as cuhre
integrand	same as cuhre; But, here, the input argument phw contains the weight of the point being sampled. This extra value can safely be ignored.
...	same as cuhre
lower	same as cuhre
upper	same as cuhre
rel.tol	same as cuhre
abs.tol	same as cuhre
flags	same as cuhre. But flags may have an additional component: smooth. When smooth = 0, apply additional smoothing to the importance function, this moderately improves convergence for many integrands. When smooth = 1 , use the importance function without smoothing, this should be chosen if the integrand has sharp edges. *Note*: Value 3 of flags$verbose has the same effect as value 2 (Vegas does not partition the integration region).
min.eval	same as cuhre
max.eval	same as cuhre
nstart	the number of integrand evaluations per iteration to start with.
nincrease	the increase in the number of integrand evaluations per iteration. The j-th iteration evaluates the integrand at nstart+(j-1)*nincrease points.
nbatch	Vegas samples points not all at once, but in batches of a predetermined size, to avoid excessive memory consumption. nbatch is the number of points sampled in each batch. Tuning this number should usually not be necessary as performance is affected significantly only as far as the batch of samples fits into the CPU cache.
gridno	an integer. Vegas may accelerate convergence to keep the grid accumulated during one integration for the next one, if the integrands are reasonably similar to each other. Vegas maintains an internal table with space for ten grids for this purpose. If gridno is a number between 1 and 10, the grid is not discarded at the end of the integration, but stored in the respective slot of the table for a future invocation. The grid is only re-used if the dimension of the subsequent integration is the same as the one it originates from. In repeated invocations it may become necessary to flush a slot in memory. In this case the negative of the

grid number should be set. Vegas will then start with a new grid and also restore the grid number to its positive value, such that at the end of the integration the grid is again stored in the indicated slot.

`state.file` the name of an external file. Vegas can store its entire internal state (i.e. all the information to resume an interrupted integration) in an external file.

The state file is updated after every iteration. If, on a subsequent invocation, Vegas finds a file of the specified name, it loads the internal state and continues from the point it left off. Needless to say, using an existing state file with a different integrand generally leads to wrong results. Once the integration finishes successfully, i.e. the prescribed accuracy is attained, the state file is removed. This feature is useful mainly to define 'check-points' in long-running integrations from which the calculation can be restarted.

Details

See details in the documentation.

Value

Idem as cuhre, except from `nregions` (not present)

References

G. P. Lepage (1978) A new algorithm for adaptive multidimensional integration. *J. Comput. Phys.*, **27**, 192-210.

G. P. Lepage (1980) VEGAS - An adaptive multi-dimensional integration program. Research Report CLNS-80/447. Cornell University, Ithaca, N.-Y.

T. Hahn (2005) CUBA-a library for multidimensional numerical integration. *Computer Physics Communications*, **168**, 78-95.

See Also

cuhre, suave, divonne

Examples

```
integrand <- function(arg, weight) {
  x <- arg[1]
  y <- arg[2]
  z <- arg[3]
  ff <- sin(x)*cos(y)*exp(z);
return(ff)
} # end integrand
vegas(3, 1, integrand, rel.tol=1e-3,  abs.tol=1e-12, flags=list(verbose=2))
```

Index