Package 'SphericalCubature'

July 2, 2014

Type Package

Title Numerical integration over spheres and balls in n-dimensions; multivariate polar coordinates

Version 1.0.1

Date 2013-05-24

Author John P. Nolan, American University

Maintainer John P. Nolan <jpnolan@american.edu>

Depends R (>= 2.1.15), cubature

Description This package defines several methods to integrate functions over the unit sphere and ball in n-dimensional Euclidean space. Routines for converting to/from multivariate polar/spherical coordinates are also provided.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2013-05-25 15:35:14

R topics documented:

SphericalCubature-package

Numerical integration over spheres and balls in n-dimensions; multivariate polar/spherical coordinates

Description

This package defines functions to integrate a (single) function f(x)=f(x[1],...,x[n]) over the unit sphere and balls in n-dimensional Euclidean space:

$$\int_S f(s)ds \qquad \text{and} \qquad \int_B f(x)dx,$$

where the first integral is over the unit sphere S, an (n-1) dimensional surface, and the second integral is over the unit ball B, an n dimensional solid.

There are three classes of methods:

1. exact methods for polynomials in any dimension (fast)
2. a method due to Stroud for smooth integrands on the sphere (in dimensions n=3,4,...,16) (slower)
3. adaptive methods for integrands with different behavior in different reasons (slowest)

Methods 2 and 3 are approximations: like any numerical quadrature algorithm, they may give bad results if the integrand changes abruptly on a small region. This happens even in one dimension, and is more difficult to find and deal with in higher dimensions. (One attempt to handle is the 'split' versions of the adaptive methods, functions adaptIntegrateSphereSplit and adaptIntegrateBallSplit, where one can split the region of integration based on knowledge of the integrand.)

An explicit goal was to get beyond the cases where n=2 or n=3, so some efficiency has been sacrificed. In all the methods, the higher the dimension n, the longer the compute time. For methods 2 and 3, compute times get noticeable when n > 5. It was also necessary for the problem that motivated this package to deal with integrands that had sharp spikes, and that requires some sort of adaptive technique.

The package includes functions to convert to/from polar coordinates in higher dimensions.

This is a first attempt to provide methods for integrating over spheres and balls in multiple dimensions. One possible improvement is speed: coding routines in C would give a significant increase in speed. Another possible extension is to include other multivariate integration methods, e.g. the package R2cuba. This may provide a way to approximate higher dimensional integrals in some cases, if the integrand is well behaved. Vector valued integrands are not supported; this could be incorporated in the future.

Constructive comments for improvements are welcome; actually implementing any suggestions will be dependent on time constraints.

Version history:

- 1.0.0 (2013-05-16) original package
- 1.0.1 (2013-05-24) fix mistake in adaptIntegrateBallSplit, fix example in integratePolynomialSphere, add more documentation

Details

Package:	SphericalCubature
Type:	Package
Version:	1.0.1
Date:	2013-05-24
License:	GPL

Author(s)

John P. Nolan

Maintainer: John P. Nolan <jpnolan@american.edu>

See Also

integrateSpherePolynomial, integrateBallPolynomial, integrateSphereStroud11, sphereArea, ballVolume, polar2rect, rect2polar, adaptIntegrateSphere, adaptIntegrateSphereSplit, adaptIntegrateBall, adaptIntegrateBallSplit

Examples

```
#  integral should just be the area of sphere in n dimensions
f1 <- function( x ) { return(1.0) }
n <- 3
sphereArea( n )
integrateSphereStroud11( f1, n )
p <- list(coef=1.0,k=matrix( rep(0L,n), nrow=1,ncol=n))
integrateSpherePolynomial( p )
adaptIntegrateSphere( f1, n )$value

# test of polynomial integration
f2 <- function( x ) { return(x[1]^2) }
sphereArea(n)/n # exact answer
integrateSphereStroud11( f2, n )
p <- list(coef=1.0,k=matrix( c(2L,rep(0L,n-1)), nrow=1) )
integrateSpherePolynomial( p )
adaptIntegrateSphere( f2, n )$value
```

`adaptIntegrateSphere` *Adaptive integration over sphere or ball in n-dimensions*

Description

Approximate the integral over the sphere or ball in n-dimensions. Can also integrate over sectors of the sphere/ball, see details. These functions will be slow, but may be necessary to get accurate answers if the integrand function f(x) is not well-behaved. If the integrand changes rapidly in certain regions, the basic routines adaptIntegrateSphere and adaptIntegrateBall will likely miss these abrupt changes and give inaccurate results. For cases where the location of the rapid changes are known, the functions adaptIntegrateSphereSplit and adaptIntegrateBallSplit allow you to split the region of integration and capture those changes.

Usage

```
adaptIntegrateSphere(f, n, lowerLimit = rep(0, n - 1),
    upperLimit = c(rep(pi, n - 2), 2 * pi), tol = 1e-05, ...)
adaptIntegrateSphereSplit(f, n, xstar, width = 0, lowerLimit = rep(0, n - 1),
    upperLimit = c(rep(pi, n - 2), 2 * pi), tol = 1e-05, ...)

adaptIntegrateBall(f, n, lowerLimit = rep(0, n - 1),
    upperLimit = c(rep(pi, n - 2), 2 * pi), R = c(0, 1), tol = 1e-05, ...)
adaptIntegrateBallSplit(f, n, xstar, width = 0, lowerLimit = rep(0, n - 1),
    upperLimit = c(rep(pi, n - 2), 2 * pi), R = c(0, 1), tol = 1e-05, ...)
```

Arguments

f	Integrand function f(x)=f(x[1],...,x[n]).
n	dimension of the space. The sphere is an (n-1) dimensional manifold inside n-space, the ball is an n-dimensional solid.
lowerLimit	Polar angular coordinates of lower limit
upperLimit	Polar angular coordinates of upper limit
tol	tolerance, the desired accuracy of the result. The functions try to get abs(exact-value) < tol
...	optional arguments passed to f. If used, these should be specified with a tag, e. g. param1=7
R	a numeric vector of length 2, integration is performed over the region with R[1] < radius < R[2].
xstar	(n x m) matrix whose columns give the directions where the integrand changes quickly, where the region will be subdivided to focus on that region. (The length of a column vector is not used, just it's direction.)
width	width of 'splitting regions', a vector of length m. If it is of length 1, then that valued is repeated for each j in 1:m. If width[j]=0, the angular region is split just at the points given by the columns xstar[,j]. If width[j] > 0, then angular region is split at an angle plus and minus width[j].

Details

Approximate the integral of f(x) over (part of) the sphere or ball in n-space. The approach is simplistic: reparameterize the region in polar coordinates. For the sphere, this makes the region of

integration a hyper-rectangle in dimension (n-1) in the angle space (here the radius is fixed: R=1). For the ball, the polar representation in terms of angles and radius gives an region of integration that is an n dimensional hyper-rectangle.

The region of integration can be a subset of the sphere/ball by specifying a patch/sector in polar coordinates. To integrate over a subregion, bounds for the polar integration have be specified. For example, in two dimensions, you can integrate over the top half of the circle by specifying lowerLimit=0.0 and upperLimit=pi to adaptIntegrateSphere. Likewise for the ball, to integrate over the part of the annulus with inner radius .2 and outer radius .7 that is in the first quadrant, specify lowerLimit=0.0, upperLimit=pi/2, R=c(.2,.7).

Value

For adaptIntegrateSphere and adaptIntegrateBall, the function returns a list containing several fields. There is always a field

`value` Giving the approximate value of the integral.

The other fields depend on the dimension: when n=2, the other fields are what is returned by the function integrate( ) in base R; when n > 2, the other fields are the fields returned by function adaptIntegrate( ) in package cubature.

For adaptIntegrateSphereSplit and adaptIntegrateBallSplit, a single value is returned. (This is because these functions make multiple calls to the adaptive integration routine and the results of each call are not saved.

See Also

polar2rect, rect2polar

Examples

```
f1 <- function( x ) { return(x[1]^2+3*x[2]+exp(x[3])) }
n <- 3
adaptIntegrateSphere( f1, n )
adaptIntegrateSphereSplit( f1, n, xstar=matrix(c(1,1,1),nrow=3) )
adaptIntegrateBall( f1, n )
adaptIntegrateBallSplit( f1, n, xstar=matrix(c(1,1,1),nrow=3) )

# test of adaptive integration with deliberate splitting
# function f3 has a sharp spike in the direction (1,2),
# elsewhere it has value 1
f3 <- function( x ) {
  x0 <- c(1.0,2.0)/sqrt(5.0)
  dist <- sqrt(sum( (x-x0)^2) )
  y <- 10-5000*dist
  y <- 1 + max(y,0)
  return(y)  }

# no splitting: this straightforward attempt at integration misses
# the spike and sees the integrand as =1 everywhere, so returns the arclengh 2*pi
n <- 2
adaptIntegrateSphere( f3, n )
```

```
# deliberate splitting at specified points, but still misses spike
# default width=0 splits the region of integration from [0,2*pi] to [0,a] U [a,2*pi],
# where tan(a)=2/1.
xstar <- matrix( c(1.0,2.0,-1.0,1.0), nrow=2 )
adaptIntegrateSphereSplit( f3, n, xstar=xstar )

# deliberate splitting around specified points, 'smart' choice of width gets the spike
# Here the region of integration is split into [0,a-.01] U [a-.01,a+.01] U [a+.01,2*pi]
adaptIntegrateSphereSplit( f3, n, xstar=xstar, width=c(0.01,0.01) )
```

`integrateSpherePolynomial`

Integration of polynomials over sphere or ball.

Description

Exact integration of polynomial over sphere or ball in n-dimensions.

Usage

```
integrateSpherePolynomial(p, valueOnly = TRUE)
integrateBallPolynomial(p, R = c(0, 1))
```

Arguments

`p`	a list specifying the coefficients and powers of the polynomial. See details below
`valueOnly`	boolean saying whether to return only the value of the integral, or return both the value and a intermediate terms. These intermediate terms are used by integrateBallPolynomial( ).
`R`	inner and outer radius of the annular region: R[1] <= radius <= R[2].

Details

Compute the exact integral over the sphere in n dimensions of a polynomial
p(x[1],...,x[n])=sum (coef[i] * x[1]^k[i,1] * ... * x[n]^k[i,n]),
where the sum is over i=1,...,m. The polynomial is specified as a list p with fields

- coef, an m-vector of doubles
- k, an (m x n) matrix of integers

m and n are given implicitly in the sizes of these arrays output is normally just a number, the value of the integral. If integrateSpherePolynomial is called with valueOnly=FALSE, a list with two fields:

- value, a double containing the value of the integral
- term, a vector of length m of values used in function IntegratePolynomialBall( )

Value

integrateSpherePolynomial( ) normally just returns a value of the integral, but if valueOnly=FALSE, it will return a list containing the value and intermediate terms. These intermediate terms correspond to the integral of each monomial term in the polynomial; they are used by integrateBallPolynomial( ).

integrateBallPolynomial( ) returns just the value of the integral.

References

Method is from How to Integrate a Polynomial over a Sphere, by G. Folland (2001), MAA Monthly 108, pg. 446-448.

Examples

```
n <- 3
# specify the polynomial p(x) = 1.0 * x[1]^2 * x[2]^0 * x[3]^0 + 7.0 * x[1]^0 * x[2]^3 * x[3]
p <- list(coef=c(1.0,7.0),k=matrix( c(2L,0L,0L,0L,3L,0L), byrow=TRUE, nrow=2) )
integrateSpherePolynomial( p )
integrateBallPolynomial( p )

# compare to adaptive integration
f4 <- function( x ) { return( x[1]^2 + 7*x[2]^2*x[3] ) }
adaptIntegrateSphere( f4, n )$value
adaptIntegrateBall( f4, n )$value
```

---

integrateSphereStroud11

Integrate a function over the sphere in n-dimensions.

Description

Approximate the integral of a function f(x)=f(x[1],...,x[n]) over the unit sphere in n-space using Stroud's method of degree 11.

Usage

```
integrateSphereStroud11(f, n, ...)
```

Arguments

f	function f(x)=f(x[1],...,x[n]) to integrate
n	dimension of the space, implemented for n in the range 3:16.
...	optional arguments passed to f( ). If these are specified, they should be labeled with a tag, e.g. param1=3.4

Details

This method works if the integrand is smooth. If the function changes rapidly, adaptive integration can be tried as described in 'See Also' below.

Value

A single number, the approximation to the integral.

References

Stroud integration and related functions, adapted from fortran code by John Burkhart found at http://people.sc.fsu.edu/~jburkardt/f77_src/stroud/stroud.html
Based on the book by A. H. Stroud, Approximate Calculation of multiple integrals, 1971, page 296-297.

See Also

adaptIntegrateSphere, adaptIntegrateBall

Examples

```
f2 <- function( x ) { return(x[1]^2) }
integrateSphereStroud11( f2, n=3 )
```

---

rect2polar *n-dimensional polar coordinate transformations*

Description

Convert between polar and rectangular coordinates in n-dimensions. The point (x[1],...,x[n]) in rectangular coordinates corresponds to the point (r,phi[1],...,phi[n-1]) in polar coordinates.

Usage

```
polar2rect(r, phi)
rect2polar(x)
```

Arguments

r	a vector of radii of length m.
phi	a (n-1) x m matrix of angles.
x	(n x m) matrix, with column j being the point in n-dimensional space.

Details

n dimensional polar coordinates are given by the following:
rectangular x=(x[1],...,x[n]) corresponds to polar (r,phi[1],...,phi[n-1]) by
x[1] = r*cos(phi[1])
x[2] = r*sin(phi[1])*cos(phi[2])
x[3] = r*sin(phi[1])*sin(phi[2])*cos
...
x[n-1]= r*sin(phi[1])*sin(phi[2])*...*sin(phi[n-2])*cos(phi[n-1])
x[n] = r*sin(phi[1])*sin(phi[2])*...*sin(phi[n-2])*sin(phi[n-1])

Here phi[1],...,phi[n-2] in [0,pi), and phi[n-1] in [0,2*pi). For multivariate integration, the Jacobian of the above tranformation is J(phi) = r^(n-1) * prod(sin(phi[1:(n-2)])^((n-2):1)); note that phi[n-1] does not appear in the Jacobian.

Value

For polar2rect( ), an (n x m) matrix of rectangular coordinates.

For rect2polar( ), a list with fields:

`r`	a vector of length m containing the radii
`phi`	an (n x m) matrix of angles

Examples

```
x <- matrix( 1:9, nrow=3 )
x
a <- rect2polar( x )
a
polar2rect( a$r, a$phi )
```

---

sphereArea *Surface area of spheres, volumes of balls in n-dimensions.*

Description

Calculates the (n-1) dimensional surface area of a sphere and the n dimensional volume of a ball in n-space.

Usage

```
sphereArea(n, R = 1)
ballVolume(n, R = 1)
```

Arguments

`n`	Dimension of the space.
`R`	Radius of the sphere/ball.

Value

Single number that is the area of the sphere/volume of the ball.

Examples

```
sphereArea(n=5)
ballVolume(n=5)
```

`SphericalMisc` *Miscellaneous functions used by SphericalCubature package.*

Description

These functions are not intended for general use, they are only documented here to acknowledge their existence and to get R build system from complaining about undocumented functions.

adaptIntegrateCheck is used by the adaptive integration functions to check input parameters, partitionRegion is used by the 'split' versions of the the adaptive integration functions, nextGraySubset is used by IntegrateSphereStroud11, nextMultiIndex is used by adaptive integration functions.

Usage

```
adaptIntegrateCheck( n, lowerLimit, upperLimit, R, xstar, width )
partitionRegion( xstar, width, lowerLimit, upperLimit )
nextGraySubset( gray.list )
nextMultiIndex( j, size )
```

Arguments

`n`	dimension of the space
`lowerLimit`	lower angular limit for integration region
`upperLimit`	upper angular limit for integration region
`R`	inner and outer radii for integration region
`xstar`	directions where function changes rapidly
`width`	width of subdivisions
`gray.list`	list used by Stroud integration
`j`	current multi-index
`size`	length of multi-index

Index