Package ‘fastGHQuad’

February 19, 2015

Type Package

Title Fast Rcpp implementation of Gauss-Hermite quadrature

Version 0.2

Date 2014-08-13

Author Alexander W Blocker

Maintainer Alexander W Blocker <ablocker@gmail.com>

Description Fast, numerically-stable Gauss-Hermite quadrature

License MIT + file LICENSE

LazyLoad yes

URL https://github.com/awblocker/fastGHQuad

Depends Rcpp (>= 0.11.0)

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-08-14 07:11:15

R topics documented:

  fastGHQuad-package ........................................ 2
  aghQuad .................................................. 3
  evalHermitePoly .......................................... 4
  findPolyRoots ............................................. 5
  gaussHermiteData ......................................... 6
  ghQuad .................................................... 7
  hermitePolyCoeff ......................................... 8

Index 10
A package for fast, numerically-stable computation of Gauss-Hermite quadrature rules

Description

This package provides functions to compute Gauss-Hermite quadrature rules very quickly with a higher degree of numerical stability (tested up to 2000 nodes).

Details

It also provides function for adaptive Gauss-Hermite quadrature, extending Laplace approximations (as in Liu & Pierce 1994).

Author(s)

Alexander W Blocker
Maintainer: Alexander W Blocker <ablocker@gmail.com>

References


See Also

gaussHermiteData, aghQuad, ghQuad

Examples

# Get quadrature rule
rule <- gaussHermiteData(1000)

# Find a normalizing constant
g <- function(x) 1/(1+x^2/10)^((11/2)) # t distribution with 10 df
aghQuad(g, 0, 1.1, rule)
# actual is
1/dt(0,10)

# Find an expectation
aghQuad

Adaptive Gauss-Hermite quadrature using Laplace approximation

Description

Convenience function for integration of a scalar function g based upon its Laplace approximation.

Usage

aghQuad(g, muHat, sigmaHat, rule, ...)

Arguments

g Function to integrate with respect to first (scalar) argument
muHat Mode for Laplace approximation
sigmaHat Scale for Laplace approximation (\sqrt{-1/H}), where H is the second derivative of g at muHat
rule Gauss-Hermite quadrature rule to use, as produced by gausshermitedata
... Additional arguments for g

Details

This function approximates

\[ \int_{-\infty}^{\infty} g(x) \, dx \]

using the method of Liu & Pierce (1994). This technique uses a Gaussian approximation of g (or the distribution component of g, if an expectation is desired) to "focus" quadrature around the high-density region of the distribution. Formally, it evaluates:

\[ \sqrt{2\pi} \sum_i w_i \exp(x_i^2) g(\mu + \sqrt{2} \sigma x_i) \]

where x and w come from the given rule.

This method can, in many cases (where the Gaussian approximation is reasonably good), achieve better results with 10-100 quadrature points than with 1e6 or more draws for Monte Carlo integration. It is particularly useful for obtaining marginal likelihoods (or posteriors) in hierarchical and multilevel models — where conditional independence allows for unidimensional integration, adaptive Gauss-Hermite quadrature is often extremely effective.

Value

Numeric (scalar) with approximation integral of g from -Inf to Inf.
Author(s)

Alexander W Blocker <ablocker@gmail.com>

References


See Also

gaussHermiteData, ghQuad

Examples

# Get quadrature rules
rule10 <- gaussHermiteData(10)
rule100 <- gaussHermiteData(100)

# Estimating normalizing constants
g <- function(x) 1/(1+x^2/10)^((11/2)) # t distribution with 10 df
aghQuad(g, 0, 1.1, rule10)
aghQuad(g, 0, 1.1, rule100)
# actual is 1/dt(0,10)

# Can work well even when the approximation is not exact
aghQuad(g, 0, 2, rule10)
aghQuad(g, 0, 2, rule100)
# actual is 2

# Estimating expectations
# Variances for the previous two distributions
aghQuad(g, 0, 1.1, rule10)
aghQuad(g, 0, 1.1, rule100)
# actual is 1.25

# Can work well even when the approximation is not exact
aghQuad(g, 0, 2, rule10)
aghQuad(g, 0, 2, rule100)
# actual is 2

---

evalHermitePoly

Evaluate Hermite polynomial at given location
findPolyRoots

Description
Evaluate Hermite polynomial of given degree at given location. This function is provided for demonstration/teaching purposes; this method is not used by gaussHermiteData. It is numerically unstable for high-degree polynomials.

Usage
```
evalHermitePoly(x, n)
```

Arguments
- `x`: Vector of location(s) at which polynomial will be evaluated
- `n`: Degree of Hermite polynomial to compute

Value
Vector of length(x) values of Hermite polynomial

Author(s)
Alexander W Blocker <ablocker@gmail.com>

See Also
gaussHermiteData, aghQuad, ghQuad

findPolyRoots

Description
Finds real parts of polynomial’s roots via eigendecomposition of companion matrix. This method is not used by gaussHermiteData. Only the real parts of each root are retained; this can be useful if the polynomial is known a priori to have all roots real.

Usage
```
findPolyRoots(c)
```

Arguments
- `c`: Coefficients of polynomial

Value
Numeric vector containing the real parts of the roots of the polynomial defined by c
gaussHermiteData

Author(s)
Alexander W Blocker <ablocker@gmail.com>

See Also
 gaussHermiteData, aghQuad, ghQuad

gaussHermiteData

Compute Gauss-Hermite quadrature rule

Description
Computes Gauss-Hermite quadrature rule of requested order using Golub-Welsch algorithm. Returns result in list consisting of two entries: x, for nodes, and w, for quadrature weights. This is very fast and numerically stable, using the Golub-Welsch algorithm with specialized eigendecomposition (symmetric tridiagonal) LAPACK routines. It can handle quadrature of order 1000+.

Usage
 gaussHermiteData(n)

Arguments
 n Order of Gauss-Hermite rule to compute (number of nodes)

Details
This function computes the Gauss-Hermite rule of order n using the Golub-Welsch algorithm. All of the actual computation is performed in C/C++ and FORTRAN (via LAPACK). It is numerically-stable and extremely memory-efficient for rules of order 1000+.

Value
A list containing:

x the n node positions for the requested rule
w the w quadrature weights for the requested rule

Author(s)
Alexander W Blocker <ablocker@gmail.com>

References
ghQuad

See Also
aghQuad, ghQuad

ghQuad

Convenience function for Gauss-Hermite quadrature

Description
Convenience function for evaluation of Gauss-Hermite quadrature

Usage
ghQuad(f, rule, ...)

Arguments
f Function to integrate with respect to first (scalar) argument; this does not include the weight function exp(-x^2)
rule Gauss-Hermite quadrature rule to use, as produced by gausshermitedata
... Additional arguments for f

Details
This function performs classical unidimensional Gauss-Hermite quadrature with the function f using the rule provided; that is, it approximates
\[ \int_{-\infty}^{\infty} f(x) \exp(-x^2) \, dx \]
by evaluating
\[ \sum_i w_i f(x_i) \]

Value
Numeric (scalar) with approximation integral of f(x)*exp(-x^2) from -Inf to Inf.

Author(s)
Alexander W Blocker <ablocker@gmail.com>

References
See Also

`gaussHermiteData`, `ghQuad`

Examples

```r
# Get quadrature rules
rule10 <- gaussHermiteData(10)
rule100 <- gaussHermiteData(100)

# Check that rule is implemented correctly
f <- function(x) rep(1,length(x))
if (isTRUE(all.equal(sqrt(pi), ghQuad(f, rule10), ghQuad(f, rule100)))) {
  print(ghQuad(f, rule10))
  print(ghQuad(f, rule100))
}
# These should be 1.772454

f <- function(x) x
if (isTRUE(all.equal(0.0, ghQuad(f, rule10), ghQuad(f, rule100)))) {
  print(ghQuad(f, rule10))
  print(ghQuad(f, rule100))
}
# These should be zero
```

---

**hermitePolyCoef**

*Get coefficient of Hermite polynomial*

**Description**

Calculate coefficients of Hermite polynomial using recursion relation. This function is provided for demonstration/teaching purposes; this method is not used by `gaussHermiteData`. It is numerically unstable for high-degree polynomials.

**Usage**

`hermitePolyCoef(n)`

**Arguments**

- `n`  
  Degree of Hermite polynomial to compute

**Value**

Vector of (n+1) coefficients from requested polynomial

**Author(s)**

Alexander W Blocker <ablocker@gmail.com>
See Also

gaussHermiteData, aghQuad, ghQuad
Index

*Topic **math**
  aghQuad, 3
  evalHermitePoly, 4
  findPolyRoots, 5
  gaussHermiteData, 6
  ghQuad, 7
  hermitePolyCoeff, 8

*Topic **package**
  fastGHQuad-package, 2

aghQuad, 2, 3, 5–7, 9

evalHermitePoly, 4

fastGHQuad (fastGHQuad-package), 2
fastGHQuad-package, 2
findPolyRoots, 5

gaussHermiteData, 2–6, 6, 7–9
ghQuad, 2, 4–7, 7, 8, 9

hermitePolyCoeff, 8