

Package ‘quadriceps’

October 6, 2026

Title Positive-Weight Cubature Rules for the Gaussian Weight and the Cube

Version 0.1.0

Description Positive-weight cubature rules in several dimensions for the Gaussian weight (ghpos) and the uniform weight on the cube (lepos): the smallest rules known to the author for dimensions 2 to 5, with a fallback to the cheapest tensor product of lower-dimensional rules elsewhere. The rules are described in Pinkse (2026) <doi:10.48550/arXiv.2609.26840>, <doi:10.5281/zenodo.22904159>. The R twin of the 'Julia' package 'Quadriceps.jl'.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

URL <https://github.com/NittanyLion/quadriceps-r>

BugReports <https://github.com/NittanyLion/quadriceps-r/issues>

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Joris Pinkse [aut, cre, cph],
Moustapha Diallo [cph] (some of the stored rules for the cube descend from his published rules (MIT license, see LICENSE.note))

Maintainer Joris Pinkse <pinkse@gmail.com>

Repository CRAN

Date/Publication 2026-10-06 07:20:02 UTC

Contents

available_rules	2
exactness_error	3
ghpos	3
lepos	5
nnodes	6
ruleinfo	7

Index**8**

available_rules	<i>List the stored rules</i>
-----------------	------------------------------

Description

All stored rules of a family, one row per rule, sorted by dimension and degree. Tensor products, which `pragmatic = TRUE` builds on demand, are not listed.

Usage

```
available_rules(family)
```

Arguments

`family` "gh" (Gaussian weight) or "le" (uniform weight on the cube).

Value

A data frame with columns

- `family`, `d` (dimension), `q`, `p` (degree of exactness, $p = 2q - 1$), `n` (number of nodes);
- `moller`: Möller's lower bound on `n`, or `-1` where it is not tabulated; a rule with `n == moller` is proven minimal;
- `relerr`: largest relative monomial error of the stored double-precision rule, measured when the data were built (see [exactness_error\(\)](#));
- `minweight`: smallest weight (always > 0);
- `interior`: `TRUE` when every node lies inside the integration domain;
- `origin`: who the rule belongs to: "own", or "derived: ...", "same-rule: ...", "transcribed: ..." followed by the published source;
- `source_id`: the origin as the small integer stored in `rules.bin` (0: own; 3: derived from Diallo and Worku; 10 and up: a published rule; see `FORMAT.md` in the repository).

Examples

```
head(available_rules("gh")[, c("d", "q", "p", "n")])
```

exactness_error	<i>Largest relative monomial error of a rule</i>
-----------------	--

Description

The maximum, over all monomials x^a of total degree $\leq p$, of $|\sum_i w_i x_i^a - E x^a| / \max(\sum_i |w_i| |x_i^a|)$, for the normalized weight of family ("gh": $N(0, I_d)$; "le": uniform on $[0, 1]^d$). The denominator is the scale of the sum being computed, so a value near machine epsilon means the rule is exact to rounding. Rules returned with `normalize = FALSE` must be checked in the normalized frame.

Usage

```
exactness_error(nodes, weights = NULL, p, family)
```

Arguments

nodes	An $n \times d$ matrix of nodes, one per row (or a rule: a list with components nodes and weights, in which case weights is taken from it).
weights	The n weights.
p	Degree up to which exactness is checked.
family	"gh" or "le".

Value

The error, a single number.

Examples

```
r <- ghpos(4, 5) # degree 2 * 5 - 1 = 9
exactness_error(r, p = 9, family = "gh")
```

ghpos	<i>Positive-weight cubature rule for the Gaussian weight</i>
-------	--

Description

The smallest positive-weight rule the package has for the Gaussian weight in d dimensions.

Usage

```
ghpos(d, q = NULL, p = NULL, normalize = TRUE, pragmatic = FALSE)
```

Arguments

d	Dimension, $d \geq 1$.
q	Number of nodes of the one-dimensional Gauss rule whose exactness is wanted, $q \geq 1$. Give q or p, not both.
p	Degree of exactness, $p \geq 0$, as an alternative to q.
normalize	With TRUE (the default), the weight is the standard normal density $(2\pi)^{-d/2} \exp(- x ^2/2)$: the weights sum to 1 and the rule computes $E f(Z)$ for $Z \sim N(0, I_d)$. With FALSE, the weight is $\exp(- x ^2)$, the convention of <code>statmod::gauss.quad(kind = "hermite")</code> , and the weights sum to $\pi^{d/2}$.
pragmatic	What to do when no stored rule covers the request. With FALSE (the default), signal an error of class <code>quadriceps_no_rule</code> . With TRUE, return the cheapest (fewest nodes) tensor product of lower-dimensional rules instead: stored rules and one-dimensional Gauss-Hermite rules, combined over the split of d that minimizes the number of nodes. Such a product is a valid positive-weight rule of the requested degree; it is just not small. With TRUE the product is also returned in the rare case that it has strictly fewer nodes than the stored rule, so the result is always the cheapest the package can build.

Details

`ghpos(d, q)` follows one-dimensional Gauss-Hermite functions such as `statmod::gauss.quad(q, kind = "hermite")`: there, q is the number of nodes of the one-dimensional Gauss rule, which is exact to degree $2q - 1$. Here, `ghpos(d, q)` returns a rule with the same exactness in d dimensions, degree $p = 2q - 1$, that replaces the q^d -node product grid; for $d = 1$ it is the q-node Gauss-Hermite rule itself. Alternatively give the degree: `ghpos(d, q)` is `ghpos(d, p = 2 * q - 1)`.

With `rule <- ghpos(d, q)`, `sum(rule$weights * f(rule$nodes))` (for a vectorized f of the rows) equals the integral of f against the weight for every polynomial f of total degree $\leq p$, up to rounding; `ruleinfo()` gives the measured error of each stored rule.

Rules are stored for $2 \leq d \leq 5$ at odd degrees, up to a ceiling that depends on d (see `available_rules()`).

A request is served by the smallest stored rule of degree $\geq p$, so an even p gets the rule for $p + 1$.

For $d = 1$ the Gauss rule is computed on the fly (Golub-Welsch) and is never an error.

Value

A list with components `nodes`, an $n \times d$ matrix with one node per row (also for $d = 1$), and `weights`, a vector of n strictly positive weights.

See Also

`lepos()`, `nnodes()`, `ruleinfo()`, `available_rules()`

Examples

```
r <- ghpos(3, 4) # degree 7: 27 nodes instead of 4^3 = 64
sum(r$weights * r$nodes[, 1]^2 * r$nodes[, 2]^4) # E[Z1^2 Z2^4] = 3
identical(ghpos(3, p = 7), r)
```

```

try(ghpos(3, 21))                # nothing stored at that degree
r <- ghpos(3, 21, pragmatic = TRUE) # tensor product of lower-dimensional rules
r <- ghpos(7, 5, pragmatic = TRUE)  # d = 7 as (d = 2) x (d = 5)

```

lepos

*Positive-weight cubature rule for the uniform weight on the cube***Description**

The smallest positive-weight rule the package has for the uniform weight on a d -dimensional cube.

Usage

```
lepos(d, q = NULL, p = NULL, normalize = TRUE, pragmatic = FALSE)
```

Arguments

<code>d</code>	Dimension, $d \geq 1$.
<code>q</code>	Number of nodes of the one-dimensional Gauss rule whose exactness is wanted, $q \geq 1$. Give <code>q</code> or <code>p</code> , not both.
<code>p</code>	Degree of exactness, $p \geq 0$, as an alternative to <code>q</code> .
<code>normalize</code>	With <code>TRUE</code> (the default), the weight is the uniform density on $[0, 1]^d$: the weights sum to 1 and the rule computes $E f(U)$ for U uniform on the unit cube. With <code>FALSE</code> , the rule is for the plain integral over $[-1, 1]^d$, the convention of <code>statmod::gauss.quad(kind = "legendre")</code> , and the weights sum to 2^d .
<code>pragmatic</code>	As for <code>ghpos()</code> : <code>FALSE</code> signals an error of class <code>quadriceps_no_rule</code> when no stored rule covers the request; <code>TRUE</code> returns the cheapest tensor product of lower-dimensional rules (stored rules and one-dimensional Gauss-Legendre rules).

Details

`lepos(d, q)` follows one-dimensional Gauss-Legendre functions such as `statmod::gauss.quad(q, kind = "legendre")`: `q` is the number of nodes of the one-dimensional Gauss rule, and `lepos(d, q)` returns a rule with the same exactness in d dimensions, degree $p = 2q - 1$, that replaces the q^d -node product grid; for $d = 1$ it is the q -node Gauss-Legendre rule itself. Alternatively give the degree `p`.

The details given for `ghpos()` apply here too. All nodes of every stored Le rule lie inside the cube.

Value

A list with components `nodes` ($n \times d$ matrix, one node per row) and `weights` (n strictly positive weights).

See Also

`ghpos()`, `nnodes()`, `ruleinfo()`, `available_rules()`

Examples

```

r <- lepos(2, 5) # degree 9: 17 nodes on [0,1]^2 instead of 25
sum(r$weights * r$nodes[, 1]^3 * r$nodes[, 2]^2) # 1/4 * 1/3

r <- lepos(2, 5, normalize = FALSE) # the same rule on [-1,1]^2, weights sum to 4
r <- lepos(6, 6, pragmatic = TRUE) # d = 6 from a product of stored rules

```

nnodes	<i>Number of nodes of a rule, without building it</i>
--------	---

Description

The number of nodes of the rule that `ghpos()` (family = "gh") or `lepos()` (family = "le") returns for the same arguments. Signals the same error when there is no rule.

Usage

```
nnodes(family, d, q = NULL, p = NULL, pragmatic = FALSE)
```

Arguments

family	"gh" or "le".
d	Dimension, $d \geq 1$.
q	Number of nodes of the one-dimensional Gauss rule whose exactness is wanted, $q \geq 1$. Give q or p, not both.
p	Degree of exactness, $p \geq 0$, as an alternative to q.
pragmatic	What to do when no stored rule covers the request. With FALSE (the default), signal an error of class <code>quadriceps_no_rule</code> . With TRUE, return the cheapest (fewest nodes) tensor product of lower-dimensional rules instead: stored rules and one-dimensional Gauss-Hermite rules, combined over the split of d that minimizes the number of nodes. Such a product is a valid positive-weight rule of the requested degree; it is just not small. With TRUE the product is also returned in the rare case that it has strictly fewer nodes than the stored rule, so the result is always the cheapest the package can build.

Value

The node count, as a double (tensor products in high dimensions exceed the integer range).

Examples

```

nnodes("gh", 5, 7)
nnodes("gh", 7, 5, pragmatic = TRUE)
5^7 # the product grid

```

ruleinfo

*Describe a rule: its factors and their origins***Description**

Describes the rule that `ghpos()` or `lepos()` returns for the same arguments.

Usage

```
ruleinfo(family, d, q = NULL, p = NULL, pragmatic = FALSE)
```

Arguments

family	"gh" or "le".
d	Dimension, $d \geq 1$.
q	Number of nodes of the one-dimensional Gauss rule whose exactness is wanted, $q \geq 1$. Give q or p, not both.
p	Degree of exactness, $p \geq 0$, as an alternative to q.
pragmatic	What to do when no stored rule covers the request. With FALSE (the default), signal an error of class <code>quadriceps_no_rule</code> . With TRUE, return the cheapest (fewest nodes) tensor product of lower-dimensional rules instead: stored rules and one-dimensional Gauss-Hermite rules, combined over the split of d that minimizes the number of nodes. Such a product is a valid positive-weight rule of the requested degree; it is just not small. With TRUE the product is also returned in the rare case that it has strictly fewer nodes than the stored rule, so the result is always the cheapest the package can build.

Details

Use it to find out whom to cite: `origin` names the published source of every rule that is not the package author's own.

Value

A data frame with one row per tensor factor, in the order of the columns of nodes, and the columns of `available_rules()`. A one-dimensional Gauss factor has `origin = "Gauss"` and NA in the columns that do not apply. A request answered by a single stored rule gives one row.

Examples

```
ruleinfo("le", 3, p = 41)$origin
ruleinfo("gh", 7, 5, pragmatic = TRUE)[, c("d", "p", "n", "origin")]
```

Index

available_rules, [2](#)
available_rules(), [4](#), [5](#), [7](#)

exactness_error, [3](#)
exactness_error(), [2](#)

ghpos, [3](#)
ghpos(), [5–7](#)

lepos, [5](#)
lepos(), [4](#), [6](#), [7](#)

nnodes, [6](#)
nnodes(), [4](#), [5](#)

ruleinfo, [7](#)
ruleinfo(), [4](#), [5](#)