Package ‘qspray’

September 7, 2023

Type Package

Title Multivariate Polynomials with Rational Coefficients

Version 2.1.1

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Symbolic calculation and evaluation of multivariate polynomials with rational coefficients. This package is strongly inspired by the 'spray' package. It also provides a function to compute Gröbner bases (reference <doi:10.1007/978-3-319-16721-3>).

License GPL-3

URL https://github.com/stla/qspray

BugReports https://github.com/stla/qspray/issues

Imports DescTools, gmp, methods, purrr, RationalMatrix, Rcpp (>= 1.0.9), Ryacas, utils

LinkingTo BH, Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.2.3

SystemRequirements C++ 17, gmp, mpfr

Collate 'RcppExports.R' 'groebner.R' 'internal.R' 'qspray.R' 'yacas.R'

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Stéphane Laurent [aut, cre],
Robin Hankin [ctb, cph] (author of the 'spray' package, which strongly inspired this package)

Repository CRAN

Date/Publication 2023-09-07 16:20:02 UTC

R topics documented:

`as.function.qspray` *Multivariate polynomial as function*

Description

Coerces a `qspray` polynomial into a function.

Usage

```
## S3 method for class 'qspray'
as.function(x, ...)
```

Arguments

`x`	object of class `qspray`
`...`	ignored

Value

A function having the same variables as the polynomial. It returns a string.

Examples

```
library(qspray)
P <- (qlone(1) + "1/2"*qlone(2))^2 + 5
f <- as.function(P)
f(2, "3/7")
f("x", "y")
# the evaluation is performed by (R)yacas and complex numbers are
# allowed; the imaginary unit is denoted by `I`
f("2 + 2*I", "1/4")
```

`as.qspray` *Coercion to a 'qspray' object*

Description

Coercion to a 'qspray' object

Usage

```
## S4 method for signature 'character'
as.qspray(x)

## S4 method for signature 'qspray'
as.qspray(x)

## S4 method for signature 'numeric'
as.qspray(x)

## S4 method for signature 'bigz'
as.qspray(x)

## S4 method for signature 'bigq'
as.qspray(x)
```

Arguments

`x`	a `qspray` object or an object yielding a quoted integer or a quoted fraction after an application of `as.character`

Value

A `qspray` object.

Examples

```
as.qspray(2)
as.qspray("1/3")
```

`derivQspray` *Partial derivative*

Description

Partial derivative of a qspray polynomial.

Usage

```
derivQspray(qspray, i, derivative = 1)
```

Arguments

`qspray`	object of class `qspray`
`i`	integer, the dimension to differentiate with respect to
`derivative`	integer, how many times to differentiate

Value

A `qspray` object.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
qspray <- 2*x  + 3*x*y
derivQspray(qspray, 1)
```

`dQspray` *Partial differentiation*

Description

Partial differentiation of a qspray polynomial.

Usage

```
dQspray(qspray, orders)
```

Arguments

`qspray`	object of class `qspray`
`orders`	integer vector, the orders of the differentiation

Value

A qspray object.

Examples

```
library(qspray)
x <- qlone(1)
y <- qlone(2)
qspray <- x + 2*y  + 3*x*y
dQspray(qspray, c(1, 1))
derivQspray(derivQspray(qspray, 1), 2)
```

ESFpoly *Elementary symmetric function*

Description

Returns an elementary symmetric function as a polynomial.

Usage

```
ESFpoly(m, lambda)
```

Arguments

m	integer, the number of variables
lambda	an integer partition, given as a vector of decreasing positive integers

Value

A qspray object.

Examples

```
library(qspray)
ESFpoly(3, c(3, 1))
```

evalQspray — *Evaluate a 'qspray' object*

Description

Evaluation of the multivariate polynomial represented by a qspray object.

Usage

```
evalQspray(qspray, values_re, values_im = NULL)
```

Arguments

qspray	a qspray object
values_re	vector of the real parts of the values; each element of as.character(values_re) must be quoted integer or a quoted fraction
values_im	vector of the imaginary parts of the values; each element of as.character(values_im) must be quoted integer or a quoted fraction

Value

A bigq number if values_im=NULL, a pair of bigq numbers otherwise: the real part and the imaginary part of the result.

Examples

```
x <- qlone(1); y <- qlone(2)
P <- 2*x + "1/2"*y
evalQspray(P, c("2", "5/2", "99999")) # "99999" will be ignored
```

groebner — *Gröbner basis*

Description

Returns a Gröbner basis following Buchberger's algorithm using the lexicographical order.

Usage

```
groebner(G, minimal = TRUE, reduced = TRUE)
```

Arguments

G	a list of qspray polynomials, the generators of the ideal
minimal	Boolean, whether to return a minimal basis
reduced	Boolean, whether to return the reduced basis

Value

A Gröbner basis of the ideal generated by `G`, given as a list of qspray polynomials.

References

Cox, Little & O'Shea. *Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Fourth edition, Springer 2015.

Examples

```
library(qspray)
f <- qsprayMaker(string = "x^(3) - 2 x^(1,1)")
g <- qsprayMaker(string = "x^(2,1) - 2 x^(0,2) + x^(1)")
groebner(list(f, g), FALSE, FALSE)
# other example
x <- qlone(1); y <- qlone(2); z <- qlone(3)
f1 <- x^2 + y + z^2 - 1
f2 <- x^2 + y + z - 1
f3 <- x + y^2 + z - 1
gb <- groebner(list(f1, f2, f3))
lapply(gb, prettyQspray, vars = c("x", "y", "z"))
```

`implicitization` *Implicitization with Gröbner bases*

Description

Implicitization of a system of parametric equations (see examples).

Usage

```
implicitization(nvariables, parameters, equations, relations)
```

Arguments

`nvariables`	number of variables
`parameters`	character vector of the names of the parameters, or `NULL` if there's no parameter
`equations`	list of qspray polynomials representing the parametric equations
`relations`	list of qspray polynomials representing the relations between the variables and the parameters, or `NULL` if there is none

Value

A list of qspray polynomials.

Examples

```
library(qspray)
# ellipse example ####
# variables
cost <- qlone(1)
sint <- qlone(2)
# parameters
a <- qlone(3)
b <- qlone(4)
#
nvariables <- 2
parameters <- c("a", "b")
equations <- list(
  "x" = a * cost,
  "y" = b * sint
)
relations <- list(
  cost^2 + sint^2 - 1
)
#
eqs <- implicitization(nvariables, parameters, equations, relations)
```

`integratePolynomialOnSimplex` *Integral of a multivariate polynomial over a simplex*

Description

Returns the exact value of the integral of a multivariate polynomial with rational coefficients over a simplex whose vertices have rational coordinates.

Usage

```
integratePolynomialOnSimplex(P, S)
```

Arguments

P	a `qspray` object
S	the simplex, a `(n+1)xn` matrix such that each entry of the matrix `as.character(S)` is a quoted integer or a quoted fraction

Value

A `bigq` number, the exact value of the integral.

Examples

```
library(qspray)
x <- qlone(1); y <- qlone(2)
P <- x/2 + x*y
S <- rbind(c("0", "0"), c("1", "0"), c("1", "1")) # a triangle
integratePolynomialOnSimplex(P, S)
```

`MSFpoly` *Monomial symmetric function*

Description

Returns a monomial symmetric function as a polynomial.

Usage

```
MSFpoly(m, lambda)
```

Arguments

`m`	integer, the number of variables
`lambda`	an integer partition, given as a vector of decreasing positive integers

Value

A `qspray` object.

Examples

```
library(qspray)
MSFpoly(3, c(3, 1))
```

`prettyQspray` *Pretty polynomial*

Description

Pretty form of a `qspray` polynomial.

Usage

```
prettyQspray(qspray, vars = NULL)
```

Arguments

`qspray`	a `qspray` object
`vars`	variable names; `NULL` for `"x1"`, `"x2"`, ...

Value

A character string.

Examples

```
library(qspray)
P <- (qlone(1) + "1/2"*qlone(2))^2 + 5
prettyP <- prettyQspray(P, vars = c("x", "y"))
prettyP
cat(Ryacas::yac_str(sprintf("PrettyForm(%s)", prettyP)))
Ryacas::yac_str(sprintf("TeXForm(%s)", prettyP))
```

`qdivision` *Division of a qspray polynomial*

Description

Division of a qspray polynomial by a list of qspray polynomials. See the reference for the definition.

Usage

```
qdivision(qspray, divisors, check = TRUE)
```

Arguments

`qspray`	the dividend, a `qspray` object
`divisors`	the divisors, a list of `qspray` objects
`check`	Boolean, whether to check the division; this argument will be removed in a future version

Value

The remainder of the division, a `qspray` object. Moreover, if `qspray` is univariate, the quotient is attached to the remainder as an attribute.

References

Michael Weiss, 2010. Computing Gröbner Bases in Python with Buchberger's Algorithm.

Examples

```
# a univariate example
library(qspray)
x <- qlone(1)
f <- x^4 - 4*x^3 + 4*x^2 - x # 0 and 1 are trivial roots
g <- x * (x - 1)
( r <- qdivision(f, list(g)) ) # should be zero
attr(r, "quotient")
```

`qlone` *Polynomial variable*

Description

Create a polynomial variable.

Usage

```
qlone(n)
```

Arguments

`n`	nonnegative integer, the index of the variable

Value

A `qspray` object.

Examples

```
qlone(2)
```

`qone` *The unit qspray polynomial*

Description

Returns the qspray polynomial identically equal to 1.

Usage

```
qone()
```

Value

A `qspray` object.

qspray-unary *Unary operators for qspray objects*

Description

Unary operators for qspray objects.

Usage

```
## S4 method for signature 'qspray,missing'
e1 + e2

## S4 method for signature 'qspray,missing'
e1 - e2
```

Arguments

e1	object of class qspray
e2	nothing

Value

A qspray object.

qsprayMaker *Make a 'qspray' object*

Description

Make a qspray object from a list of exponents and a vector of coefficients.

Usage

```
qsprayMaker(powers, coeffs, string = NULL)
```

Arguments

powers	list of positive integer vectors
coeffs	a vector such that each element of as.character(coeffs) is a quoted integer or a quoted fraction; it must have the same length as the powers list
string	if not NULL, this argument takes precedence over powers and vertices; it must be a string representing a multivariate polynomial; see the example

Value

A qspray object.

Examples

```
powers <- list(c(1, 1), c(0, 2))
coeffs <- c("1/2", "4")
qsprayMaker(powers, coeffs)
qsprayMaker(string = "1/2 x^(1, 1) + 4 x^(0, 2)")
```

`qzero` *The null qspray polynomial*

Description

Returns the qspray polynomial identically equal to 0.

Usage

```
qzero()
```

Value

A `qspray` object.

Index