Package ‘PolynomF’

October 12, 2022

Type Package
Title Polynomials in R
Description Implements univariate polynomial operations in R, including
polynomial arithmetic, finding zeros, plotting, and some operations on
lists of polynomials.
Version 2.0-5
Date 2022-05-02
NeedsCompilation yes
Imports stats, Rcpp, methods
Depends R (>= 3.0.0), graphics, grDevices
Author Bill Venables, with contribution by Kurt Hornik and Georgi Boshnakov
Maintainer Bill Venables <Bill.Venables@gmail.com>
License GPL-2
Encoding UTF-8
LinkingTo Rcpp
RoxygenNote 7.1.2
Suggests knitr, rmarkdown
VignetteBuilder knitr
Repository CRAN
Date/Publication 2022-05-02 03:30:02 UTC

R topics documented:

as.character.polynom ........................................... 2
as.function.polynom ........................................... 3
c.polynom ....................................................... 4
change_origin ................................................... 4
coeF.polynom .................................................... 5
deriv.polynom ................................................... 6
GCD ............................................................. 7
as.character.polynom

Polynomial coercion to character

Description

Produce a text representation of a polynomial object

Usage

```r
## S3 method for class 'polynom'
as.character(x, variable = "x", decreasing = FALSE, ...)
```

Arguments

- `x`: The polynomial object in question
- `variable`: Character string: what variable name should be used?
- `decreasing`: Logical: in decreasing powers or increasing powers?
- `...`: Additional arguments (ignored as yet)

Value

A character string representation of the polynomial
Examples

```r
p <- poly_from_zeros(-2:3)
as.character(p, "z", FALSE)
as.character(p, "z", TRUE)
p.parse(text = as.character(p, "z", TRUE))[[1]]
```

Description

PolynomF objects ARE functions, but this coercion method creates from a polynomial object a pure function with the coefficients fully exposed in the code and which evaluates the polynomial more efficiently.

Usage

```r
## S3 method for class 'polynom'
as.function(x, variable = "x", ...)

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

Arguments

- `x` A polynomial object
- `variable` Character string: what variable name should be used?
- `...` Additional arguments

Value

An explicit R function evaluating the polynomial

Examples

```r
p <- poly_from_zeros(-2:3)
p
as.function(p)
```
c.polynom

Concatenation of polynomial objects into lists

Description
Concatenation of polynomial objects into lists

Usage

## S3 method for class 'polynom'
c(..., recursive = FALSE)

## S3 method for class 'polylist'
c(..., recursive = FALSE)

Arguments

... Polynomial or polylist objects
recursive Logical, should the concatenation flatten all component lists?

Value
A polylist object with all arguments included

change_origin

Change origin of a polynomial

Description
Given a polynomial P(x) and a new origin o, find the polynomial Q(x) = P(x + o). I.e. Q(0) = P(o)

Usage

change_origin(p, o, ...)

## Default S3 method:
change_origin(p, o, ...)

## S3 method for class 'polynom'
change_origin(p, o, ...)

## S3 method for class 'polylist'
change_origin(p, o, ...)
Arguments

- \( p \) A polynom or polylist object
- \( o \) A single numeric quantity specifying the new \( x \)-origin
- \( \ldots \) currently not used

Value

A polynom or polylist object with \( x \) measured from the new origin

Description

Extract polynomial coefficients

Usage

```r
## S3 method for class 'polynom'
coef(object, \ldots)

## S3 method for class 'polylist'
coef(object, \ldots)
```

Arguments

- \( \text{object} \) A polynomial object or list thereof
- \( \ldots \) Ignored

Value

A numeric vector of coefficients

Examples

```r
p <- polynomial(1:3) * polynomial(5:1)
coef(p)
```
Description

Find the derivative or indefinite integral of a polynomial object, or list thereof.

Usage

```r
## S3 method for class 'polynom'
deriv(expr, ...)
integral(expr, ...)

## Default S3 method:
integral(expr, ...)

## S3 method for class 'polynom'
integral(expr, limits = NULL, ...)

## S3 method for class 'polylist'
deriv(expr, ...)

## S3 method for class 'polylist'
integral(expr, ...)
```

Arguments

- `expr` A polynomial object, or list thereof
- `...` Unused as yet
- `limits` Real limits of a definite integral

Value

A coefficient vector, or list thereof

Examples

```r
p <- poly_from_roots(-2:3)
p
deriv(p)
integral(p)
```
GCD

Greatest common divisor

Description

Find a monic polynomial of maximal degree that divides each of a set of polynomials exactly

Usage

GCD(...)
greatest_common_divisor(...)  
## S3 method for class 'polynom'
GCD(...)  
## S3 method for class 'polylist'
GCD(...)

Arguments

... A list of polynomials or polylist objects

Value

A polynomial giving the greatest common divisor, as defined above

Examples

p <- poly_calc(0:5)
r <- poly_calc(1:6)
greatest_common_divisor(p, r)
solve(greatest_common_divisor(p, r))
lowest_common_multiple(p, r)
solve(lowest_common_multiple(p, r))

GroupGenerics Summary and Math methods for polynomials

Description

These provide methods for the generic function Summary and Math for polynomial and polylist objects. For Summary only sum and prod members are implemented
Usage

```r
## S3 method for class 'polynom'
Summary(..., na.rm = FALSE)

## S3 method for class 'polylist'
Summary(..., na.rm = FALSE)

## S3 method for class 'polynom'
Math(x, ...)

## S3 method for class 'polylist'
Math(x, ...)
```

Arguments

- `...` Additional arguments
- `na.rm` Logical: should missing values be removed?
- `x` a "polynom" or "polylist" objects.

Value

The result of the group generic operation

Examples

```r
lis <- as_polylist(lapply(-2:3, function(x) polynomial() - x))
prod(lis)
sum(lis)
solve(prod(lis))
solve(sum(lis))
```

---

**LCM**

*Lowest Common Multiple*

Description

For a list of polynomials, find the lowest degree monic polynomial into which each divides exactly

Usage

```r
LCM(...)

lowest_common_multiple(...)

## S3 method for class 'polynom'
LCM(...)

## S3 method for class 'polylist'
LCM(...)
```
Arguments

... A list of polynomials or polylist objects

Value

A polynomial giving the lowest common multiple

Examples

```r
p <- poly_calc(0:5)
r <- poly_calc(1:6)
greatest_common_divisor(p, r)
solve(greatest_common_divisor(p, r))
lowest_common_multiple(p, r)
solve(lowest_common_multiple(p, r))
```

---

neville

Lagrange Interpolation Polynomials

Description

Compute the Lagrange Interpolation Polynomial from a given set of x- and y-values, or, alternatively, compute the interpolated values at a set of given x-values. Two algorithms are provided, namely Neville’s algorithm, or a more direct version based on the usual Lagrange formula. The latter is generally faster but the former can be more accurate numerically.

Usage

```r
neville(x, y, x0 = polynomial())
lagrange(x, y, x0 = polynomial())
```

Arguments

- `x` A numeric vector of x-values
- `y` A numeric values of y-values corresponding to the x-values
- `x0` Either a polynomial object or a vector of x-values for which interpolated y-values are required.

Value

Either an interpolation polynomial object or a vector of interpolated y-values
Examples

```r
set.seed(123)
x <- 1:5
y <- rnorm(x)
xout <- 0.5 + 1:4

p1 <- neville(x, y)
plot(p1, xlim = range(x), ylim = extendrange(y, f = 1), panel.first = grid())
points(x, y, col = 4)
points(xout, lagrange(x, y, xout), col = 2)
```

---

### Ops.polynom

#### Polynomial arithmetic

**Description**

Group generic function to implement arithmetic operations on polynomial objects.

**Usage**

```r
## S3 method for class 'polynom'
Ops(e1, e2)

## S3 method for class 'polylist'
Ops(e1, e2)
```

**Arguments**

- `e1, e2` A numeric vector of a polynomial object. At least one of `e1` or `e2` must be an object of class "polynom" or "polylist".

**Value**

A polynomial or polylist object representing the result of the operation.

**Examples**

```r
x <- polynomial()
(p <- (x-1)^5 - 1)
(p1 <- (p + 1)/(x - 1)^2 - 1)
for(i in 0:10) cat(coef((x+1)^i), "\n")
```
plot.polylist

Plot method for polynomials

Description

Plot methods for polynom or polylist objects

Usage

## S3 method for class 'polylist'
plot(
x, 
xlim = 0:1, 
ylim = range(Px), 
type = "l", 
xlab = "x", 
ylab = "P(x)", 
..., 
col = seq_along(x), 
lty = if (length(col) == 1) seq_along(x) else "solid", 
len = 1000, 
legend = FALSE
)

## S3 method for class 'polynom'
plot(
x, 
xlim = 0:1, 
ylim = range(Px), 
type = "l", 
xlab = "x", 
ylab = "p(x)", 
..., 
len = 1000, 
limits = pu[1:2]
)

## S3 method for class 'polynom'
lines(x, ..., len = 1000, limits = pu[1:2])

## S3 method for class 'polynom'
points(x, ..., len = 100, limits = pu[1:2])

## S3 method for class 'polylist'
lines(
x, 
...,
```r
len = 1000,
limits = pu[1:2],
col = seq_along(x),
ltk = if (length(col) == 1) seq_along(x) else "solid"
)

## S3 method for class 'polylist'
points(x, ..., len = 100)
```

### Arguments

- **x**: A polynom or polylist object to be plotted
- **xlim, ylim**: as for graphics::plot
- **type**: as for graphics::plot
- **xlab, ylab**: as for graphics::plot
- **...**: additional arguments passed on to methods
- **col, lty**: Colour(s) and line type(s) as for graphics::plot
- **len**: positive integer defining the point or curve resolution
- **legend**: logical: for "polylist" objects, should a legend be drawn alongside the main plot?
- **limits**: x-limits for the polynomial, default: the entire plot. For polylist objects this may be a two column matrix.

### Value

Nothing of interest, invisibly

### Examples

```r
p <- poly_from_zeros((-3):4)
plot(p)
lines(deriv(p), col = "red")
```
Usage

polynom(a = c(0, 1), ..., eps = 0)
polynomial(a = c(0, 1), ..., eps = 0)
as_polynom(a)
is_polynom(a)
polylist(...)
is_polylist(x)
as_polylist(x)

Arguments

a         A polynom object, or a numeric vector of coefficients (in "power series" order)
          or a vector object which can be coerced to one.
...
eps      A small non-negative tolerance to check for zero components.
x         An object of class "polylist", at least potentially.

Value

A polynomial object.

Examples

(s <- polynomial())
p <- polynomial(c(1, 5, 4, 1)/11)
oldPar <- par(mar = c(5,5,2,2)+0.1)
plot(p, xlim = 0:1, ylim = 0:1, type = "n", bty="n",
     xlab = "s", ylab = expression({P}^{(n)}(s)))
lines(s, limits = 0:1)
P <- p
for(j in 1:7) {
  lines(P, col = j+1, limits = 0:1)
  P <- p(P)
}
lines(P, limits = 0:1, col = 9)
(r <- Re(solve((p-s)/(1-s))))
arrows(r, p(r), r, par("usr")[3], lwd = 0.5,
       length = 0.125, angle = 15)
text(r, 0.025, paste("r =", format(r, digits = 3)))
leg <- sapply(0:8, function(x) bquote({P}^{(.(x))}(s)))
legend("topleft", legend = as.expression(leg),
        lty = "solid", col = 1:9, bty = "n", ncol=3)
par(oldPar)
rm(leg, oldPar, p, P, r, s, j)
**poly_calc**  

Lagrange interpolation polynomial

**Description**

Calculate the Lagrange interpolation polynomial, or list of polynomials, given a set of (x, y) points to fit.

**Usage**

```r
poly_calc(x, y, tol = sqrt(.Machine$double.eps), lab = dimnames(y)[[2]])
```

```r
poly_from_zeros(…)
```

```r
poly_from_roots(…)
```

```r
poly_from_values(x, y, tol = sqrt(.Machine$double.eps), lab = dimnames(y)[[2]])
```

**Arguments**

- `x`  
  A numeric vector of x-points at which the y-values are specified.

- `y`  
  Either a numeric vector of the same length as `x` or a numeric matrix with rows matching the length of `x`. If `y` is missing (not specified) then a polynomial with zero at `x` is returned.

- `tol`  
  A numeric tolerance for duplicated `x` values.

- `lab`  
  A character string vector of names for the list result when `y` is a matrix.

- `…`  
  A list of specified zeros (for subsidiary functions)

**Value**

An interpolation polynomial, or list of interpolating polynomials.

**Examples**

```r
(p <- poly_calc(0:5))  ## same as poly_from_zeros(0:5)
(p <- poly_calc(0:5, exp(0:5)))
plot(p)
curve(exp, add = TRUE, col = "red")
```
**poly_orth**

### Description

Generate a list of polynomials up to a specified degree, orthogonal with respect to the natural inner product on a discrete, finite set of x-values with equal weights.

### Usage

```
poly_orth(x, degree = length(unique(x)) - 1, norm = TRUE)
```

### Arguments

- **x**  
  A numeric vector

- **degree**  
  The desired maximum degree

- **norm**  
  Logical: should polynomials be normalised to length one?

### Value

A list of orthogonal polynomials as a polylist object

### Examples

```
x <- c(0:3, 5)
P <- poly_orth(x)
plot(P, lty = "solid")
Pf <- as.function(P)
zap(crossprod(Pf(x)))
```

---

**poly_orth_general**

### General Orthogonal Polynomials

### Description

Generate sets of polynomials orthogonal with respect to a general inner product. The inner product is specified by an R function of (at least) two polynomial arguments.
Usage

poly_orth_general(inner_product, degree, norm = FALSE, ...)

Hermite(p, q = p)
Legendre(p, q = p)
ChebyshevT(p, q = p)
ChebyshevU(p, q = p)
Jacobi(p, q = p, alpha = -0.5, beta = alpha)
Discrete(p, q = p, x, w = function(x, ...) 1, ...)

Arguments

inner_product An R function of two "polynom" arguments with the second polynomial having a default value equal to the first. Additional arguments may be specified. See examples
degree A non-negative integer specifying the maximum degree
norm Logical: should the polynomials be normalized?
... additional arguments passed on to the inner product function
p, q Polynomials
alpha, beta Family parameters for the Jacobi polynomials
x numeric vector defining discrete orthogonal polynomials
w a weight function for discrete orthogonal polynomials

Details

Discrete orthogonal polynomials, equally or unequally weighted, are included as special cases. See the Discrete inner product function.
Computations are done using the recurrence relation with computed coefficients. If the algebraic expressions for these recurrence relation coefficients are known the computation can be made much more efficient.

Value

A "polylist" object containing the orthogonal set

Examples

(P0 <- poly_orth(0:5, norm = FALSE))
(P1 <- poly_orth_general(Discrete, degree = 5, x = 0:5, norm = FALSE))
sapply(P0-P1, function(x) max(abs(coef(x)))) ## visual check for equality
(P0 <- poly_orth_general(Legendre, 5))
##### should be same as P0, up to roundoff
(P1 <- poly_orth_general(Jacobi, 5, alpha = 0, beta = 0))
### check
sapply(P0-P1, function(x) max(abs(coef(x))))

---

**predict.polynom**  
Evaluate a polynomial

### Description
Evaluate a polynomial, or polylist object components.

### Usage

```r
## S3 method for class 'polynom'
predict(object, newdata, ...)

## S3 method for class 'polylist'
predict(object, newdata, ...)
```

### Arguments
- **object**  
  A polynomial or polylist object
- **newdata**  
  A target object at which to evaluate.
- **...**  
  Not used

### Value
If `newdata` is a numeric vector, a numeric vector of results. If `newdata` is a polynomial, then the composition is returned as a polynomial, or polylist object.

---

**print.polylist**  
Print method for polynomial objects

### Description
Print method for polynomial objects

### Usage

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

### Arguments
- **x**  
  A polynomial object or list thereof
- **...**  
  Additional arguments passed on to methods
Value

The original object, invisibly.

Description

Standard method for printing polynomial objects

Usage

## S3 method for class 'polynom'
print(x, variable = "x", digits = getOption("digits"), decreasing = FALSE, ...)

Arguments

x A polynomial object
variable Character string: what variable name should be given?
digits Integer: how many decimal digits to use?
decreasing Logical: in descending powers, or ascending?
... Additional arguments

Value

The original object x, invisibly

Description

Repeat components of a polylist object

Usage

## S3 method for class 'polylist'
rep(x, times, ...)

## S3 method for class 'polynom'
rep(x, times, ...)
solve.polynom

Arguments

x          A single polynom or polylist object
times, ...  As for the base package function rep.

Value

The resulting polylist object.

solve.polynom  Find Polynomial Zeros

Description

Solve polynomial equations, a(x) = b(x), or alternatively find the zeros of the polynomial a(x) - b(x)

Usage

## S3 method for class 'polynom'
solve(a, b, ...)

## S3 method for class 'polylist'
solve(a, b, ...)

Arguments

a, b          Polynomials for the LHS and RHS respectively
...           Currently unused

Value

A vector of roots, usually complex

Examples

p <- poly_calc(0:5)
solve(p)
solve(p, 1)
**summary.polynom**  
*Polynomial summary*

**Description**

Provide a succinct summary of the critical points of a polynomial, or list thereof.

**Usage**

```r
## S3 method for class 'polynom'
summary(object, ...)

## S3 method for class 'polylist'
summary(object, ...)

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

**Arguments**

- `object, x`  
  A polynomial or polylist object
- `...`  
  Currently unused

**Value**

A list giving the zeros, stationary points and points of inflexion of the polynomial(s).

**Examples**

```r
p <- poly_calc(0:5)
summary(p)
```

---

**tangent**  
*Tangent lines*

**Description**

Find the tangent line to a polynomial at one or more x-points.

**Usage**

```r
tangent(p, x0)
```

**Arguments**

- `p`  
  A polynomial object
- `x0`  
  A numeric vector of values at which the tangent line(s) are required
### Value

A linear polynomial giving the tangent line, or a list of such polynomials

### Examples

```r
p <- poly_from_zeros(c(0, 0:5, 4))
plot(p, xlab = expression(italic(x)), ylab = expression(italic(P(x))),
     main = parse(text = paste("italic(P(x) ==",
                                as.character(p, decreasing = TRUE),")")))
x0 <- solve(deriv(p))  ## stationary points
lines(tangent(p, x0), col = "dark green", lty = "solid",
      limits = cbind(x0-1/4, x0+1/4))
points(x0, p(x0), col = "dark green")

x0 <- solve(deriv(deriv(p)))  ## points of inflexion
lines(tangent(p, x0), col = "red", lty = "solid", lwd = 2,
      limits = cbind(x0-1/4, x0+1/4))
points(x0, p(x0), col = "red")
legend("bottomleft", c("Stationary points", "Points of inflexion"),
       pch = 19, col = c("dark green", "red"), lty = "solid",
       cex = 0.7, bg = "beige", box.lwd = 0.25)
```

### Description

Remove duplicated polynomials in a polylist object

### Usage

```r
## S3 method for class 'polylist'
unique(x, incomparables = FALSE, ...)
```

### Arguments

- `x`: A polylist object
- `incomparables`: Logical: as for the base function `unique`
- `...`: As for the base function `unique`

### Value

A polylist object with no duplicated components
A convenience function for setting polynomial coefficients likely to be entirely round-off error to zero. The decision is relegated to the function `base::zapsmall`, to which this is a front-end.

Usage

zap(x, digits = getOption("digits"))

## Default S3 method:
zap(x, digits = getOption("digits"))

## S3 method for class 'polynom'
zap(x, digits = getOption("digits"))

## S3 method for class 'polylist'
zap(x, digits = getOption("digits"))

## S3 method for class 'list'
zap(x, digits = getOption("digits"))

Arguments

x  A polynomial or polylist object

digits As for base::zapsmall

Value

A polynomial or polylist object with minuscule coefficients set to zero.

Examples

(P <- poly_orth(-2:2, norm = FALSE))
zap(35*P)
Description

Extract components of a list of polynomials

Usage

```r
## S3 method for class 'polylist'
x[i]
```

Arguments

- `x` A polylist object
- `i` An index vector of any congruent form

Value

A polylist object of the components
Index

[.polylist, 23
as.character.polynom, 2
as.function.polylist
  (as.function.polynom), 3
as.function.polynom, 3
as_polylist (polynom), 12
as_polynom (polynom), 12
c.polylist(c.polynom), 4
c.polynom, 4
change_origin, 4
ChebyshevT (poly_orth_general), 15
ChebyshevU (poly_orth_general), 15
goef.polylist (coef.polynom), 5
coef.polynom, 5
deriv.polylist (deriv.polynom), 6
deriv.polynom, 6
Discrete (poly_orth_general), 15
GCD, 7
greatest_common_divisor (GCD), 7
GroupGenerics, 7
Hermite (poly_orth_general), 15
integral (deriv.polynom), 6
is_polylist (polynom), 12
is_polynom (polynom), 12
Jacobi (poly_orth_general), 15
lagrange (neville), 9
LCM, 8
Legendre (poly_orth_general), 15
lines.polylist (plot.polylist), 11
lines.polynom (plot.polylist), 11
lowest_common_multiple (LCM), 8
Math.polylist (GroupGenerics), 7
Math.polynom (GroupGenerics), 7
neville, 9
Ops.polylist (Ops.polynom), 10
Ops.polynom, 10
plot.polylist, 11
plot.polynom (plot.polylist), 11
points.polylist (plot.polylist), 11
points.polynom (plot.polylist), 11
poly_calc, 14
poly_from_roots (poly_calc), 14
poly_from_values (poly_calc), 14
poly_from_zeros (poly_calc), 14
poly_orth, 15
poly_orth_general, 15
polylist (polynom), 12
polynom, 12
polynomial (polynom), 12
predict.polylist (predict.polynom), 17
predict.polynom, 17
print.polylist, 17
print.polynom, 18
print.summary.polynom
  (summary.polynom), 20
rep.polylist, 18
rep.polynom (rep.polylist), 18
solve.polylist (solve.polynom), 19
solve.polynom, 19
Summary.polylist (GroupGenerics), 7
summary.polylist (summary.polynom), 20
Summary.polynom (GroupGenerics), 7
summary.polynom, 20
tangent, 20
unique.polylist, 21
zap, 22