Package ‘minimaxApprox’

June 20, 2024

Type Package

Title Implementation of Remez Algorithm for Polynomial and Rational Function Approximation

Version 0.4.3

Date 2024-06-20

Description Implements the algorithm of Remez (1962) for polynomial minimax approximation and of Cody et al. (1968) <doi:10.1007/BF02162506> for rational minimax approximation.

License MPL-2.0

URL https://github.com/aadler/MiniMaxApprox

BugReports https://github.com/aadler/MiniMaxApprox/issues

Imports stats, graphics

Suggests tinytest, covr

ByteCompile yes

NeedsCompilation yes

Encoding UTF-8

UseLTO yes

Author Avraham Adler [aut, cre, cph] (<https://orcid.org/0000-0002-3039-0703>)

Maintainer Avraham Adler <Avraham.Adler@gmail.com>

Repository CRAN

Date/Publication 2024-06-20 07:10:02 UTC

Contents

minimaxApprox-package 2
coef.minimaxApprox 3
minimaxApprox 4
minimaxErr 8
minimaxEval 9
plot.minimaxApprox 10
print.minimaxApprox 11

Index **12**

`minimaxApprox-package` *Implementation of Remez Algorithm for Polynomial and Rational Function Approximation*

Description

Implements the algorithm of Remez (1962) for polynomial minimax approximation and of Cody et al. (1968) <doi:10.1007/BF02162506> for rational minimax approximation.

Details

The DESCRIPTION file:

Package:	minimaxApprox
Type:	Package
Title:	Implementation of Remez Algorithm for Polynomial and Rational Function Approximation
Version:	0.4.3
Date:	2024-06-20
Authors@R:	person(given="Avraham", family="Adler",role=c("aut", "cre", "cph"), email="Avraham.Adler@gmail.c
Description:	Implements the algorithm of Remez (1962) for polynomial minimax approximation and of Cody et al. (
License:	MPL-2.0
URL:	https://github.com/aadler/MiniMaxApprox
BugReports:	https://github.com/aadler/MiniMaxApprox/issues
Imports:	stats, graphics
Suggests:	tinytest, covr
ByteCompile:	yes
NeedsCompilation:	yes
Encoding:	UTF-8
UseLTO:	yes
Author:	Avraham Adler [aut, cre, cph] (<https://orcid.org/0000-0002-3039-0703>)
Maintainer:	Avraham Adler <Avraham.Adler@gmail.com>
Archs:	x64

Index of help topics:

```
coef.minimaxApprox      Extract coefficients from a '"minimaxApprox"'
                        object
minimaxApprox           Minimax Approximation of Functions
minimaxApprox-package   Implementation of Remez Algorithm for
                        Polynomial and Rational Function Approximation
minimaxErr              Evaluate the Minimax Approximation Error
minimaxEval             Evaluate Minimax Approximation
plot.minimaxApprox      Plot errors from a '"minimaxApprox"' object
print.minimaxApprox     Print method for a '"minimaxApprox object"'
```

Author(s)

Avraham Adler [aut, cre, cph] (<https://orcid.org/0000-0002-3039-0703>)

Maintainer: Avraham Adler <Avraham.Adler@gmail.com>

`coef.minimaxApprox` *Extract coefficients from a* `"minimaxApprox"` *object*

Description

Extracts the numerator and denominator vectors from a `"minimaxApprox"` object. For objects with both Chebyshev and monomial coefficients, it will extract both.

Usage

```
## S3 method for class 'minimaxApprox'
coef(object, ...)
```

Arguments

`object`	An object inheriting from class `"minimaxApprox"`.
`...`	Other arguments.

Value

Coefficients extracted from the `"minimaxApprox"` object. A list containing:

`a`	The polynomial coefficients or the rational numerator coefficients.
`b`	The rational denominator coefficients. Missing for polynomial approximation.
`aMono`	The polynomial coefficients or the rational numerator coefficients for the monomial basis when the approximation was done using Chebyshev polynomials. Missing if only the monomial basis was used.
`bMono`	The rational denominator coefficients for the monomial basis when the approximation was done using Chebyshev polynomials. Missing if either only the monomial basis was used or for polynomial approximation.

Author(s)

Avraham Adler `<Avraham.Adler@gmail.com>`

See Also

`minimaxApprox`

Examples

```
PP <- minimaxApprox(exp, 0, 1, 5)
coef(PP)
identical(unlist(coef(PP), use.names = FALSE), c(PP$a, PP$aMono))

RR <- minimaxApprox(exp, 0, 1, c(2, 3), basis = "m")
coef(RR)
identical(coef(RR), list(a = RR$a, b = RR$b))
```

`minimaxApprox` *Minimax Approximation of Functions*

Description

Calculates minimax approximations to functions. Polynomial approximation uses the Remez (1962) algorithm. Rational approximation uses the Cody-Fraser-Hart (Cody et al., 1968) version of the algorithm. When using monomials as the polynomial basis, the Compensated Horner Scheme of Langlois et al. (2006) is used.

Usage

```
minimaxApprox(fn, lower, upper, degree, relErr = FALSE, basis ="Chebyshev",
              xi = NULL, opts = list())
```

Arguments

`fn`	function; A vectorized univariate function having `x` as its first argument. This could be a built-in R function, a predefined function, or an anonymous function defined in the call; see **Examples**.
`lower`	numeric; The lower bound of the approximation interval.
`upper`	numeric; The upper bound of the approximation interval.
`degree`	integer; Either a single value representing the requested degree for polynomial approximation or a vector of length 2 representing the requested degrees of the numerator and denominator for rational approximation.
`relErr`	logical; If `TRUE`, calculate the minimax approximation using *relative* error. The default is `FALSE` which uses *absolute* error.
`basis`	character; Which polynomial basis to use in the analysis. `"Monomial"` uses the standard x^k basis. `"Chebyshev"` uses the Chebyshev polynomials of the first kind, T_k. The default is `"Chebyshev"`, and the parameter is case-insensitive and may be abbreviated.
`xi`	numeric; For rational approximation, a vector of initial points of the correct length—$\sum(\texttt{degree}) + 2$. If missing, the approximation will use the appropriate Chebyshev nodes. Polynomial approximation **always** uses Chebyshev nodes and will ignore `xi` with a message.
`opts`	list; Configuration options including:

- `maxiter`: integer; The maximum number of iterations to attempt convergence. Defaults to 100.
- `miniter`: integer; The minimum number of iterations before allowing convergence. Defaults to 10.
- `conviter`: integer; The number of successive iterations with the same results allowed before assuming no further convergence is possible. Defaults to 30. Will overwrite `maxiter` and `miniter` if `conviter` is explicitly passed and is larger than either one.
- `showProgress`: logical; If `TRUE` will print error values at each iteration.
- `convrat`: numeric; The convergence ratio tolerance. Defaults to $1 + 1 \times 10^{-9}$. See **Details**.
- `tol`: numeric; The absolute difference tolerance. Defaults to 1×10^{-14}. See **Details**.
- `tailtol`: numeric; The tolerance of the coefficient of the largest power of `x` to be ignored when performing the polynomial approximation a second time. Defaults to the smaller of 1×10^{-10} or $\frac{\texttt{upper}-\texttt{lower}}{10^6}$. Set to `NULL` to skip the `degree + 1` check completely. See **Details**.
- `ztol`: numeric; The tolerance for each polynomial or rational numerator or denominator coefficient's contribution to **not** to be set to 0. Similar to polynomial `tailtol` but applied at each step of the algorithm. Defaults to `NULL` which leaves all coefficients as they are regardless of magnitude. See **Details**.

Details

Convergence: The function implements the Remez algorithm using linear approximation, chiefly as described by Cody et al. (1968). Convergence is considered achieved when all three of the following criteria are met:

1. The observed error magnitudes are within tolerance of the expected error—the **Distance Test**.
2. The observed error magnitudes are within tolerance of each other—the **Magnitude Test**.
3. The observed error signs oscillate—the **Oscillation Test**.

"Within tolerance" can be met in one of two ways:

1. **Difference**: The difference between the absolute magnitudes is less than or equal to `tol`.
2. **Ratio**: The ratio between the absolute magnitudes of the larger and smaller is less than or equal to `convrat`.

For efficiency, the **Distance Test** is taken between the absolute value of the largest observed error and the absolute value of the expected error. Similarly, the **Magnitude Test** is taken between the absolute value of the largest observed error and the absolute value of the smallest observed error. Both tests can be passed by **either** being within `tol` or `convrat` as described above. However, when the **Difference** test returns values less than machine precision, it is ignored in favor of the **Ratio** test.

When the error values remain within tolerance of each other over `conviter` iterations, the algorithm will stop, as it is expected that no further precision will be gained by continued iterations.

Polynomial Evaluation: Monomial polynomials are evaluated using the Compensated Horner Scheme of Langlois et al. (2006) to enhance both stability and precision. Chebyshev polynomials are evaluated normally. There may be cases where the algorithm will fail using the monomial basis but succeed using Chebyshev polynomials and vice versa. The default is to use the Chebyshev polynomials.

Polynomial Algorithm "Singular Error" Response: When too high of a degree is requested for the tolerance of the algorithm, it often fails with a singular matrix error. In this case, for the *polynomial* version, the algorithm will try looking for an approximation of degree `n + 1`. If it finds one, **and** the contribution of that coefficient to the approximation is $\leq$ `tailtol`, it will ignore that coefficient and return the resulting degree `n` polynomial, as the largest coefficient is effectively 0. The contribution is measured by multiplying that coefficient by the endpoint with the larger absolute magnitude raised to the `n + 1` power. This is done to prevent errors in cases where a very small coefficient is found on a range with very large absolute values and the resulting contribution to the approximation is **not** *de minimis*. Setting `tailtol` to `NULL` will skip the `n + 1` test completely.

Close-to-Zero Tolerance: For each step of the algorithms' iterations, the contribution of the found coefficient to the total sum (as measured in the above section) is compared to the `ztol` option. When less than or equal to `ztol`, that coefficient is set to 0. Setting `ztol` to `NULL` skips the test completely. For intervals near or containing zero, setting this option to anything other than `NULL` may result in either non-convergence or poor results. It is recommended to keep it as `NULL`, although there are edge cases where it may allow convergence where a standard call may fail.

Value

`minimaxApprox` returns an object of class `"minimaxApprox"` which inherits from the class list.

The generic accessor function `coef` will extract the numerator and denominator vectors. There are also default `print` and `plot` methods.

An object of class `"minimaxApprox"` is a list containing the following components:

- `a` — The polynomial or rational numerator coefficients. When using Chebyshev polynomials, these are the coefficients for T_k. When using monomials, these are the coefficients for x^k.
- `b` — The rational denominator coefficients. When using Chebyshev polynomials, these are the coefficients for T_k. When using monomials, these are the coefficients for x^k. Missing for polynomial approximation.
- `aMono` — When using Chebyshev polynomials, these are the polynomial or rational numerator coefficients for monomial expansion in x^k. Missing for monomial-based approximation.
- `bMono` — When using Chebyshev polynomials, these are the rational denominator coefficients for monomial expansion in x^k. Missing for both polynomial and monomial-based rational approximation.
- `ExpErr` — The absolute value of the expected error as calculated by the Remez algorithms.
- `ObsErr` — The absolute value of largest observed error between the function and the approximation at the extremal points.
- `iterations` — The number of iterations of the algorithm. This does not include any iterations required to converge the error value in rational approximation.
- `Extrema` — The extrema at which the minimax error was achieved.

`Warning`	A logical flag indicating if any warnings were thrown.

The object also contains the following attributes:

`type`	"Rational" or "Polynomial".
`basis`	"Monomial" or "Chebyshev".
`func`	The function being approximated.
`range`	The range on which the function is being approximated.
`relErr`	A logical indicating that relative error was used. If `FALSE`, then absolute error was used.
`tol`	The tolerance used for the **Distance Test**.
`convrat`	The tolerance used for the **Magnitude Test**.

Note

At present, the algorithms are implemented using machine double precision, which means that the approximations are at best slightly worse. Research proceeds on more precise, stable, and efficient implementations. So long as the package remains in an experimental state—noted by a 0 major version—the API may change at any time.

Author(s)

Avraham Adler <`Avraham.Adler@gmail.com`>

References

Remez, E. I. (1962) *General computational methods of Chebyshev approximation: The problems with linear real parameters*. US Atomic Energy Commission, Division of Technical Information. AEC-tr-4491

Fraser W. and Hart J. F. (1962) "On the computation of rational approximations to continuous functions", *Communications of the ACM*, **5**(7), 401–403, doi:10.1145/368273.368578

Cody, W. J. and Fraser W. and Hart J. F. (1968) "Rational Chebyshev approximation using linear equations", *Numerische Mathematik*, **12**, 242–251, doi:10.1007/BF02162506

Langlois, P. and Graillat, S. and Louvet, N. (2006) "Compensated Horner Scheme", in *Algebraic and Numerical Algorithms and Computer-assisted Proofs.* Dagstuhl Seminar Proceedings, **5391**, doi:10.4230/DagSemProc.05391.3

See Also

`minimaxEval`, `minimaxErr`

Examples

```
minimaxApprox(exp, 0, 1, 5)                    # Built-in & polynomial

fn <- function(x) sin(x) ^ 2 + cosh(x)         # Pre-defined
minimaxApprox(fn, 0, 1, c(2, 3), basis = "m")  # Rational
```

```
minimaxApprox(function(x) x ^ 3 / sin(x), 0.7, 1.6, 6L)  # Anonymous

fn <- function(x) besselJ(x, nu = 0)                     # More than one input
b0 <- 0.893576966279167522                               # Zero of besselY
minimaxApprox(fn, 0, b0, c(3L, 3L))                      # Cf. DLMF 3.11.19
```

`minimaxErr` *Evaluate the Minimax Approximation Error*

Description

Evaluates the difference between the function and the minimax approximation at `x`.

Usage

```
minimaxErr(x, mmA)
```

Arguments

`x`	a numeric vector
`mmA`	a "`minimaxApprox`" return object

Details

This is a convenience function to evaluate the approximation error at `x`. It will use the same polynomial basis as was used in the approximation; see `minimaxApprox` for more details.

Value

A vector of the same length as `x` containing the approximation error values.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

See Also

`minimaxApprox`, `minimaxEval`

Examples

```
# Show results
x <- seq(0, 0.5, length.out = 11L)
mmA <- minimaxApprox(exp, 0, 0.5, 5L)
err <- minimaxEval(x, mmA) - exp(x)
all.equal(err,  minimaxErr(x, mmA))

# Plot results
x <- seq(0, 0.5, length.out = 1001L)
plot(x, minimaxErr(x, mmA), type = "l")
```

minimaxEval *Evaluate Minimax Approximation*

Description

Evaluates the rational or polynomial approximation stored in `mmA` at `x`.

Usage

```
minimaxEval(x, mmA, basis = "Chebyshev")
```

Arguments

`x`	a numeric vector
`mmA`	a `"minimaxApprox"` return object
`basis`	character; Which polynomial basis to use in to evaluate the function; see `minimaxApprox` for more details. If Chebyshev is requested but the analysis used only monomials, the calculation will proceed using the monomials with a message. The default is `"Chebyshev"`, and the parameter is case-insensitive and may be abbreviated.

Details

This is a convenience function to evaluate the approximation at `x`.

Value

A vector of the same length as `x` containing the approximated values.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

See Also

`minimaxApprox`, `minimaxErr`

Examples

```
# Show results
x <- seq(0, 0.5, length.out = 11L)
mmA <- minimaxApprox(exp, 0, 0.5, 5L)
apErr <- abs(exp(x) - minimaxEval(x, mmA))
all.equal(max(apErr), mmA$ExpErr)

# Plot results
curve(exp, 0.0, 0.5, lwd = 2)
curve(minimaxEval(x, mmA), 0.0, 0.5, add = TRUE, col = "red", lty = 2L, lwd = 2)
```

plot.minimaxApprox *Plot errors from a "minimaxApprox" object*

Description

Produces a plot of the error of the "minimaxApprox" object, highlighting the error extrema and bounds.

Usage

```
## S3 method for class 'minimaxApprox'
plot(x, y, ...)
```

Arguments

x	An object inheriting from class "minimaxApprox".
y	Ignored. In call as required by R in Writing R Extensions:chapter 7.
...	Further arguments to plot. Specifically to pass `ylim` to allow for zooming in or out.

Value

No return value; called for side effects.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

See Also

`minimaxApprox`

Examples

```
PP <- minimaxApprox(exp, 0, 1, 5)
plot(PP)
```

print.minimaxApprox — *Print method for a* "minimaxApprox object"

Description

Provides a more human-readable output of a "minimaxApprox" object.

Usage

```
## S3 method for class 'minimaxApprox'
print(x, digits = 14L, ...)
```

Arguments

`x`	An object inheriting from class "minimaxApprox".
`digits`	integer; Number of digits to which to round the ratio.
`...`	Further arguments to `print`.

Details

To print the raw "minimaxApprox" object use `print.default`.

Value

No return value; called for side effects.

Author(s)

Avraham Adler <Avraham.Adler@gmail.com>

See Also

minimaxApprox

Examples

```
PP <- minimaxApprox(sin, 0, 1, 8)
PP
print(PP, digits = 2L)
print.default(PP)
```

Index

∗ **NumericalMathematics**
coef.minimaxApprox, 3
minimaxApprox, 4
minimaxErr, 8
minimaxEval, 9
plot.minimaxApprox, 10
print.minimaxApprox, 11
∗ **hplot**
plot.minimaxApprox, 10
∗ **methods**
coef.minimaxApprox, 3
plot.minimaxApprox, 10
print.minimaxApprox, 11
∗ **optimize**
minimaxApprox, 4
∗ **package**
minimaxApprox-package, 2
∗ **print**
print.minimaxApprox, 11

class, *3*, *6*, *10, 11*
coef.minimaxApprox, 3

list, *3, 4*, *6*

minimaxApprox, *3*, 4, *8–11*
minimaxApprox-package, 2
minimaxErr, *7*, 8, *9*
minimaxEval, *7, 8*, 9

plot.minimaxApprox, 10
print.minimaxApprox, 11