Package 'ecd'

December 22, 2015

Type Package

Title Elliptic Distribution Based on Elliptic Curves

Version 0.6.3

Date 2015-12-22

Author Stephen H-T. Lihn [aut, cre]

Maintainer Stephen H-T. Lihn <stevelihn@gmail.com>

Description An implementation of the univariate elliptic distribution and elliptic option pricing model. It provides detailed functionality and data sets for the distribution and modelling. Especially, it contains functions for the computation of density, probability, quantile, fitting procedures, option prices, volatility smile. It also comes with sample financial data, and plotting routines.

Depends R (>= 3.1.2)

Imports stats, Rmpfr, polynom, xts, zoo, optimx, moments, parallel, graphics, methods, yaml, RSQLite

Suggests knitr, testthat, roxygen2, ghyp, fOptions

License Artistic-2.0

Encoding latin1

LazyData true

Collate 'ecd-adj-gamma-method.R' 'ecd-asymp-stats-method.R' 'ecd-ccdf-method.R' 'ecd-cdf-method.R' 'ecd-numericMpfr-class.R' 'ecdq-class.R' 'ecd-package.R' 'ecd-class.R' 'ecd-constructor.R' 'ecd-cubic-method.R' 'ecd-cusp-a2r-method.R' 'ecd-cusp-constructor.R' 'ecd-cusp-std-moment-method.R' 'ecd-data-config-internal.R' 'ecd-data-method.R' 'ecd-data-stats-method.R' 'ecd-df2ts-method.R' 'ecd-discr-generic.R' 'ecd-distribution-method.R' 'ecld-class.R' 'ecd-ecldOrEcd-class.R' 'ecd-ellipticity-generic.R' 'ecd-estimate-const-method.R' 'ecd-fit-data-method.R' 'ecd-fit-ts-conf-method.R' 'ecd-has-quantile-method.R' 'ecd-imgf-method.R' 'ecd-integrate-method.R' 'ecd-integrate-pdf-generic.R' 'ecd-is-numericMpfr-internal.R' 'ecd-jinv-generic.R' 'ecd-lag-method.R' 'ecd-manage-hist-tails-method.R' 'ecd-max-kurtosis-method.R' 'ecd-model-internal.R' 'ecd-moment-generic.R'

'ecd-mp2f-method.R' 'ecd-mpfr-method.R'
'ecd-mpfr-qagi-method.R' 'ecd-mpnum-method.R'
'ecd-ogf-method.R' 'ecd-pdf-method.R' 'ecd-plot-2x2-generic.R'
'ecd-polar-constructor.R' 'ecd-quantilize-generic.R'
'ecd-rational-method.R' 'ecd-read-csv-by-symbol-method.R'
'ecd-read-symbol-conf-method.R' 'ecd-sd-method.R'
'ecd-setup-const-method.R' 'ecd-solve-cusp-asym-method.R'
'ecd-solve-generic.R' 'ecd-solve-sym-generic.R'
'ecd-solve-trig-generic.R' 'ecd-standardfit-method.R'
'ecd-stats-method.R' 'ecd-toString-method.R'
'ecd-ts-lag-stats-method.R' 'ecd-uniroot-method.R'
'ecd-y-slope-generic.R' 'ecd-y0-isomorphic-method.R'
'ecdattr-class.R' 'ecdattr-constructor.R'
'ecdattr-enrich-method.R' 'ecdattr-pairs-method.R'
'ecdattr-pairs-polar-method.R' 'ecdb-class.R'
'ecdb-bootstrap-generic.R' 'ecdb-constructor.R'
'ecdb-dbSendQuery-method.R' 'ecdb-ecdattr-generic.R'
'ecdb-helpers-internal.R' 'ecdb-history-generic.R'
'ecdb-protectiveCommit-method.R' 'ecdb-read-generic.R'
'ecdb-summary-generic.R' 'ecdb-write-generic.R'
'ecdq-constructor.R' 'ecld-cdf-method.R' 'ecld-const-method.R'
'ecld-constructor.R' 'ecld-gamma-method.R' 'ecld-imgf-method.R'
'ecld-imnt-method.R' 'ecld-ivol-ogf-star-method.R'
'ecld-mgf-term-method.R' 'ecld-moment-method.R'
'ecld-mpnum-method.R' 'ecld-mu-D-method.R' 'ecld-ogf-method.R'
'ecld-ogf-star-method.R' 'ecld-op-V-method.R'
'ecld-pdf-method.R' 'ecld-sd-method.R' 'ecld-sged-method.R'
'ecld-solve-method.R' 'ecld-y-slope-method.R'
'ecop-bs-implied-volatility-method.R'
'ecop-bs-option-price-method.R' 'ecop-opt-class.R'
'ecop-class.R' 'ecop-constructor.R' 'ecop-plot-option-method.R'
'ecop-polyfit-option-method.R' 'ecop-read-csv-by-symbol.R'

RoxygenNote 5.0.1

R topics documented:

`ecd-package` *ecd: A package for the elliptic distribution.*

Description

The ecd package provides the core class and functions to calculate the elliptic distribution. They are either generic or use `ecd.` namespace. The lambda distribution is using `ecld.` namespace. SGED is considered part of ecld. The option pricing API is using `ecop.` namespace.

Author(s)

Stephen H-T. Lihn

bootstrap.ecdb *Bootstrap data for the Elliptic DB (ECDB)*

Description

Main interface to generate data for ECDB based on the configuration.

Usage

```
## S3 method for class 'ecdb'
bootstrap(object, action = "all", skip.existing = TRUE)

bootstrap(object, action = "all", skip.existing = TRUE)

## S4 method for signature 'ecdb'
bootstrap(object, action = "all", skip.existing = TRUE)
```

Arguments

object	an object of ecdb class.
action	the action operating on the ecdb.
skip.existing	logical, if TRUE (default), skip if action already done in history.

Value

Row count.

dec *The Elliptic Distribution*

Description

Density, distribution function, quantile function, and random generation for the univariate elliptic distribution.

Usage

```
dec(x, object = ecd())

pec(q, object = ecd())

qec(p, object = ecd(with.quantile = TRUE), debug = FALSE)

rec(n, object = ecd(with.quantile = TRUE))
```

Arguments

x	numeric vector of quantiles.
object	an object of ecd class. To achieve high performance for qec and rec, it should be created with with.quantile=TRUE.
q	numeric vector of quantiles.
p	numeric vector of probabilities.
debug	logical, whether to print debug message, default is FALSE.
n	number of observations.

Value

dec gives the density, pec gives the distribution function, qec gives the quantile function, rec generates random deviates.

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(with.quantile=TRUE)
x <- seq(-20, 20, by=5)
dec(x,d)
pec(x,d)
p <- c(0.0001, 0.001, 0.01, 0.99, 0.999, 0.9999)
qec(p,d)
rec(100,d)
```

---

discr.ecd *Discriminant of the elliptic curve $y(x)$*

Description

Discriminant of the elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
discr(object, no.validate = FALSE)

discr(object, no.validate = FALSE)

## S4 method for signature 'ecd'
discr(object, no.validate = FALSE)
```

Arguments

object	an object of ecd class
no.validate	logical, if TRUE, don't validate presence of beta. Default is FALSE.

Value

the discriminant

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd(-1,1)
discr(d)
```

`ecd` *Constructor of ecd class*

Description

Construct an ecd class by providing the required parameters. The default is the standard cusp distribution. Cusp is validated by `eps = max(.Machine$double.eps*1000, 1e-28)`.

Usage

```
ecd(alpha = 0, gamma = 0, sigma = 1, beta = 0, mu = 0, cusp = 0,
  lambda = 3, with.stats = TRUE, with.quantile = FALSE,
  bare.bone = FALSE, verbose = FALSE)
```

Arguments

`alpha`	numeric, the flatness parameter. Default: 0.
`gamma`	numeric, the sharpness parameter. Default: 0.
`sigma`	numeric, the scale parameter. Must be positive. Default: 1.
`beta`	numeric, the skewness parameter. Default: 0.
`mu`	numeric, the location parameter. Default: 0.
`cusp`	logical, indicate type of cusp. The singular points in cusp requires special handling. Default: 0, not a cusp. 1: cusp with alpha specified. 2: cusp with gamma specified.
`lambda`	numeric, the leading exponent for the special model. Default: 3.
`with.stats`	logical, also calculate statistics, default is `TRUE`.
`with.quantile`	logical, also calculate quantile data, default is `FALSE`.
`bare.bone`	logical, skip both const and stats calculation, default is `FALSE`. This for debug purpose for issues on integrating $e^y(x)$.
`verbose`	logical, display timing information, for debugging purpose, default is `FALSE`.

Value

An object of ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
d <- ecd(1,1)
d <- ecd(alpha=1, gamma=1)
```

`ecd-class` *The ecd class*

Description

This S4 class is the major object class for elliptic distribution. It stores the ecd parameters, numerical constants that facilitates quadpack integration, statistical attributes, and optionally, an internal structure for the quantile function.

Slots

`call` The match.call slot

`alpha,gamma,sigma,beta,mu` a length-one numeric. These are core ecd parameters.

`cusp` a length-one numeric as cusp indicator. 0: not a cusp; 1: cusp specified by `alpha`; 2: cusp specified by `gamma`.

`lambda` a length-one numeric, the leading exponent for the special model, default is 3.

`R,theta` a length-one numeric. These are derived ecd parameters in polar coordinate.

`use.mpfr` logical, internal flag indicating whether to use mpfr.

`const` A length-one numeric as the integral of $exp(y(x))$ that normalizes the PDF.

`const_left_x` A length-one numeric marking the left point of PDF integration.

`const_right_x` A length-one numeric marking the right point of PDF integration.

`stats` A list of statistics, see `ecd.stats` for more information.

`quantile` An object of ecdq class, for quantile calculation.

`model` A vector of four strings representing internal classification: `long_name.skew`, codelong_name, `short_name.skew`, `short_name`. This slot doesn't have formal use yet.

ecd.adj_gamma *Discriminant-adjusted gamma*

Description

Adjust gamma by diescriminant conversion formula so that the critical line is a straight 45-degree line. The inverse adjustment is also provided.

Usage

```
ecd.adj_gamma(gamma)

ecd.adj2gamma(adj_gamma)
```

Arguments

gamma	numeric, the gamma paramter
adj_gamma	numeric, the disciminant-adjusted gamma

Value

adjusted gamma (or the reverse of adjustment)

Examples

```
gamma2 <- ecd.adj_gamma(c(1,2))
gamma <- ecd.adj2gamma(c(1,2))
```

ecd.asymp_stats *Compute asymptotic statistics of an ecd object*

Description

The main API for asymptotic statistics. It follows the same definition of moments, except the integral of PDF is limited to a range of quantile. That is to truncate the tails. The asymptotic kurtosis is also called truncated kurtosis.

Usage

```
ecd.asymp_stats(object, q)

ecd.asymp_kurtosis(object, q)
```

Arguments

object	an object of ecd class with quantile
q	numeric vector of quantiles

Value

a list of stats list, or a vector of kurtosis

Examples

```
## Not run:
    d <- ecd(1,1, with.quantile=TRUE)
    q <- 0.01
    ecd.asymp_stats(d,q)
    ecd.asymp_kurtosis(d,q)

## End(Not run)
```

---

ecd.ccdf *Complementary CDF of ecd*

Description

Complementary CDF of ecd, integration of PDF from `x` to `Inf`

Usage

```
ecd.ccdf(object, x, to.x = Inf, piece.wise = FALSE, f = NULL,
  verbose = FALSE)
```

Arguments

`object`	An object of ecd class
`x`	A numeric vector of `x`
`to.x`	A value or a vector of starting `x`, default `Inf` This is for internal use only.
`piece.wise`	Logical. If `TRUE`, use cumulative method for large array. Default to FALSE. Use it with a scalar `to.x`.
`f`	an optional extension to perform integral on function other than 1. This is for internal use only. You should use the respective wrappers.
`verbose`	logical, display timing information, for debugging purpose.

Value

The CCDF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
x <- seq(0, 10, by=1)
ecd.ccdf(d,x)
```

ecd.cdf *CDF of ecd*

Description

CDF of ecd, integration of PDF from `-Inf` (or a point of choice) to `x`

Usage

```
ecd.cdf(object, x, from.x = -Inf, piece.wise = FALSE, f = NULL,
  verbose = FALSE)
```

Arguments

`object`	An object of ecd class
`x`	A numeric vector of `x`
`from.x`	A value or a vector of starting `x`, default `-Inf`
`piece.wise`	Logical. If `TRUE`, use cumulative method for large array. Default to FALSE. Use it with a scalar `from.x`.
`f`	an optional extension to perform integral on function other than 1. This is for internal use only. You should use the respective wrappers.
`verbose`	logical, display timing information, for debugging purpose.

Value

The CDF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
x <- seq(-10, 10, by=1)
ecd.cdf(d,x)
ecd.cdf(d,1, from.x = -1)
```

ecd.cubic *Generate or solve the cubic polynomial for ecd*

Description

Generate or solve the polynomial from ecd. This is usually transient for solve. Or it can be used for studying singular points.

Usage

```
ecd.cubic(object, x = 0, solve = TRUE)
```

Arguments

object	An object of ecd class
x	A vector of x dimension
solve	Logical, solve the polynomial, default = TRUE.

Value

list of the polynomial object, or result of solve.

Examples

```
d <- ecd()
ecd.cubic(d)
ecd.cubic(d, 0)
```

---

ecd.cusp *Cusp constructor of ecd class*

Description

Construct an ecd class for cusp distribution by specifying either alpha or gamma, but not both. At the moment, it can't handle beta.

Usage

```
ecd.cusp(alpha = NaN, gamma = NaN, sigma = 1, mu = 0,
  with.stats = TRUE, with.quantile = FALSE, bare.bone = FALSE,
  verbose = FALSE)
```

Arguments

alpha	numeric, the flatness parameter. Default: NaN.
gamma	numeric, the sharpness parameter. Default: NaN.
sigma	numeric, the scale parameter. Must be positive. Default 1.
mu	numeric, the location parameter. Default: 0.
with.stats	logical, also calculate statistics, default is TRUE.
with.quantile	logical, also calculate quantile data, default is FALSE.
bare.bone	logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.
verbose	logical, display timing information, for debugging purpose, default is FALSE.

Value

The ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd.cusp(alpha=1)
d <- ecd.cusp(gamma=-1)
```

`ecd.cusp_a2r` *Conversion between alpha and gamma for cusp distribution*

Description

`ecd.cusp_a2r` converts from alpha to gamma. `ecd.cusp_r2a` converts from gamma to alpha.

Usage

```
ecd.cusp_a2r(alpha)

ecd.cusp_r2a(gamma)
```

Arguments

`alpha`	numeric
`gamma`	numeric

Value

gamma for `a2r`; alpha for `r2a`.

Author(s)

Stephen H-T. Lihn

Examples

```
gamma <- ecd.cusp_a2r(alpha=1)
alpha <- ecd.cusp_r2a(gamma=1)
```

`ecd.cusp_std_moment` *The moments, characteristic function (CF), and moment generating function (MGF) of standard cusp distribution.*

Description

The moments of standard cusp distribution are calculated via Gamma function. The CF and MGF are calculated as sum of moment terms. The CF is a complex number. Since the terms in MGF is ultimately diverging, the sum is truncated before the terms are increasing.

Usage

```
ecd.cusp_std_moment(n)

ecd.cusp_std_cf(t, mu = 0, sigma = 1, rel.tol = 1e-08,
  show.warning = FALSE)

ecd.cusp_std_mgf(t, mu = 0, sigma = 1, rel.tol = 1e-07,
  show.warning = FALSE)
```

Arguments

n	integer vector specifying the n-th moments
t	numeric vector for CF and MGF
mu	length-one numeric, specifying mean for CF and MGF
sigma	length-one numeric, specifying volatility for CF and MGF
rel.tol	relative tolerance
show.warning	logical, to show warning or not.

Value

the values of the moments, CF, MGF

Examples

```
ecd.cusp_std_moment(c(2,4))
```

---

ecd.data *Read sample data*

Description

Read sample data into xts by specifying the symbol. The xts object has two rows: the prices indexed by the dates.

Usage

```
ecd.data(symbol = "dji")
```

Arguments

symbol	Character for the symbol of the time series. Default: dji

Value

The xts object for the time series

Examples

```
dji <- ecd.data()
wti <- ecd.data("wti")
```

ecd.data_stats *Statistics and histogram on log returns*

Description

Statistics and histogram on log returns are added to the xts attributes

Usage

```
ecd.data_stats(ts = "dji", breaks = 20, merge_tails = c(0, 0),
  with.tail = FALSE, tail.N1 = 7, tail.N2 = 5)
```

Arguments

ts	can be either a symbol of sample data, or the xts object from sample data
breaks	A length-one numeric, breaks for generating the histogram.
merge_tails	A length-two numeric vector. The first element is how many points in the left tail of histogram to be dropped during fitting. The second element is how many points in the right tail of histogram to be dropped during fitting.
with.tail	logical, include tail statistics, mainly on asypmtotic kurtosis. Default: FALSE.
tail.N1	a numeric, defining the wider range of tail statistics
tail.N2	a numeric, defining the smaller range of tail statistics

Value

The xts object containing ecd added attributes

Examples

```
dji <- ecd.data_stats(ecd.data("dji"))
dji <- ecd.data_stats("dji")
```

ecd.df2ts *Utility to standardize timeseries from data.frame to xts*

Description

This utility converts the df input to an xts object with columns and statistics required for the fitting/plot utility in the ecd package. The require columns are Date, Close, logr. This utility can also be used to convert the input from Quandl.

Usage

```
ecd.df2ts(df, date_format = "%m/%d/%Y", dt = "Date", col_in = "Close",
  col_out = "Close", do.logr = TRUE, rnd.zero = 0.01)
```

Arguments

df	Data.frame of the time serie
date_format	Character, date format of the input date column. It can be NULL to indicate no date conversion is needed. Default: "%m/%d/%Y".
dt	Character, the name of the input date column. Default: "Date"
col_in	Character, the name of the input closing price column. Default: "Close"
col_out	Character, the name of the output closing price column. Default: "Close"
do.logr	logical, if TRUE (default), produce xts object of logr; otherwise, just the col_out column.
rnd.zero	numeric, a small random factor to avoid an unreal peak of zero log-returns.

Value

The xts object for the time series

Examples

```
## Not run:
ecd.df2ts(df)

## End(Not run)
```

---

ecd.estimate_const *Estimate the normalization constant for an ecd object*

Description

This is an internal helper function for ecd constructor Its main function is to estimate const using analytical formula, without any dependency on statistics and numerical integration.

Usage

```
ecd.estimate_const(object)
```

Arguments

object	An object of ecd class

Value

numeric, estimated const

Examples

```
ecd.estimate_const(ecd(100,100, sigma=0.1, bare.bone=TRUE))
```

ecd.fit_data *Sample data fit*

Description

Fitting sample data to ecd with a starting set of parameters. This is the highest level wrapper of the fitting routine.

Usage

```
ecd.fit_data(symbol = "dji", iter = 1000, FIT = FALSE, EPS = FALSE,
  conf_file = "conf/ecd-fit-conf.yml", eps_file = NULL, qa.fit = FALSE)
```

Arguments

symbol	Character. The symbol of sample data. Default: dji.
iter	A length-one numeric. Number of maximum iterations. Default: 1000.
FIT	Logical, indicating whether to call linear regression, default = FALSE
EPS	Logical, indicating whether to save the plot to EPS, default = FALSE
conf_file	File name fof symbol config, default to conf/ecd-fit-conf.yml
eps_file	File name for eps output
qa.fit	Logical, qa the standardfit_fn once.

Value

Final ecd object

Examples

```
## Not run:
dji <- ecd.fit_data("dji", FIT=T)

## End(Not run)
```

ecd.fit_ts_conf *Timeseries fitting utility*

Description

Fitting timeseries with provided conf as starting set of parameters.

Usage

```
ecd.fit_ts_conf(ts, conf, iter = 1000, FIT = FALSE, EPS = FALSE,
  eps_file = NULL, qa.fit = FALSE)
```

Arguments

ts	An xts object from either ecd.data or ecd.df2ts.
conf	A nested list object, the configuration.
iter	A length-one numeric. Number of maximum iterations. Default: 1000.
FIT	Logical, indicating whether to call linear regression, default = FALSE
EPS	Logical, indicating whether to save the plot to EPS, default = FALSE
eps_file	File name for eps output
qa.fit	Logical, qa the standardfit_fn once.

Value

Final ecd object

Examples

```
## Not run:
d <- ecd.fit_ts_conf(ts, conf)

## End(Not run)
```

---

ecd.has_quantile *Whether the ecd object has quantile data or not*

Description

Whether the ecd object has quantile data or not. This is mostly for internal use.

Usage

```
ecd.has_quantile(object)
```

Arguments

object	an object of ecd class

Value

logical, whether the object has quantile data or not.

Author(s)

Stephen H-T. Lihn

ecd.imgf *Incomplete MGF of ecd*

Description

Incomplete moment generating function (IMGF) of ecd, integration of $e^z P(z)$ for z from x to Inf. ecd.mu_D is simply a wrapper around MGF.

Usage

```
ecd.imgf(object, x = -Inf, t = 1, minus1 = FALSE, unit.sigma = FALSE,
  n.sigma = .ecd.mpfr.N.sigma, verbose = FALSE)

ecd.mu_D(object)
```

Arguments

object	an object of ecd class
x	a numeric vector of x, default to -Inf
t	a numeric value for MGF, default to 1
minus1	logical, subtracting one from e^{tx}
unit.sigma	logical, transforming to unit sigma to achieve greater stability. Due to the instability of quadpack for ecd.integrate_pdf, default to TRUE. But constructing a new ecd object has significant overhead, be aware of it in performance sensitive program.
n.sigma	length-one numeric, specifying the max number of sigma to check for truncation.
verbose	logical, display timing information, for debugging purpose.

Value

The IMGF

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0, 0, sigma=0.01)
x <- seq(0, 1, by=0.1)
ecd.imgf(d, x)
```

ecd.integrate *Wrapper to integrate numeric and mpfr*

Description

The wrapper handles chooses to to use `integrate` for numeric; or to use `integrateR` for mpfr. Since the later doesn't allow infinity, there is a special handling to replace infinity with a large multiple of `sigma`.

Usage

```
ecd.integrate(object, f, lower, upper, ...,
  abs.tol = .Machine$double.eps^0.25, mpfr.qagi = TRUE,
  show.warning = TRUE)
```

Arguments

`object`	An object of ecd class. This object can be bare-boned.
`f`	An R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
`lower`	Numeric, the lower limit of integration. Can be infinite.
`upper`	Numeric, the upper limit of integration. Can be infinite.
`...`	Addtional arguments for `f`.
`abs.tol`	numeric, the suggested absolute tolerance.
`mpfr.qagi`	logical, to use quadpack qagi transformation for infinity.
`show.warning`	logical, to suppress warnings or not.

Value

The `integrate` object

Author(s)

Stephen H. Lihn

ecd.lag *Utility to shift a vector of numeric or mpfr*

Description

This utility is basically the same as `Hmisc::Lag`, but it handles mpfr vector properly.

Usage

```
ecd.lag(x, shift = 1, na.omit = FALSE)
```

Arguments

x	a vector of numeric or mpfr
shift	integer, cells to shift
na.omit	logical, whether to remove the NAs

Value

the shifted vector

Examples

```
x <- ecd.lag(c(1,2,3))
y <- ecd.lag(ecd.mpfr(c(1,2,3)))
```

---

ecd.manage_hist_tails *Manage histogram tails*

Description

Manage histogram tails to remove very far outliers. `histuple` is `list(hx = hist$mids, hy = hist$counts)`, which is an internal representation of histogram

Usage

```
ecd.manage_hist_tails(htu, merge_tails = c(0, 0))
```

Arguments

htu	list, input histuple
merge_tails	length-two numeric vector, points to be merged for left and right tails

Value

list, histuple

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
htu2 <- ecd.manage_hist_tails(htu, c(1,2))

## End(Not run)
```

ecd.max_kurtosis *Utility to calculate where the maximum kurtosis is on the positive j=0 line*

Description

This utility calculates the kurtosis for alpha from 2.85 to 3.00. Then the location and value of maximum kurtosis is presented.

Usage

```
ecd.max_kurtosis(jinv = 0)
```

Arguments

jinv	specify 0 (default) or 1728.

Value

numeric vector, in which the first element is alpha, and the second element is the maximum kurtosis.

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
    k <- ecd.max_kurtosis()
    alpha <- k[1]
    kurtosis <- k[2]

## End(Not run)
```

ecd.mp2f *Wrapper to convert mpfr to numeric*

Description

Convert mpfr to numeric primarily for display messages.

Usage

```
ecd.mp2f(x)
```

Arguments

x	an object of mpfr class. If x is numeric class, it will be passed through.

Value

a numeric vector

Examples

```
x <- ecd.mp2f(ecd.mpfr(c(1,2,3)))
```

`ecd.mpfr` *Wrapper to convert numeric to mpfr*

Description

Convert numeric to mpfr for ecd calculations. `ecd.mp1` is the constant 1 wrapped in mpfr class. `ecd.erf` is a wrapper on `Rmpfr::erf`. `ecd.devel` is a developer tool to size down intensive mpfr tests for CRAN. Set `ecd_devel` in R options or OS env to change its value.

Usage

```
ecd.mpfr(x, precBits = getOption("ecd.precBits"))

ecd.mp1

ecd.erf(x)

ecd.devel()
```

Arguments

`x`	a numeric vector or list. If `x` is mpfr class, it will be passed through.
`precBits`	an integer for mpfr precBits. Default is from `getOption("ecd.precBits")`.

Format

An object of class `mpfr` of length 1.

Value

The `mpfr` object

Examples

```
x <- ecd.mpfr(1)
y <- ecd.mpfr(c(1,2,3))
z <- ecd.mp1
```

ecd.mpfr_qagi *Utility to integrate mpfr with infinity via qagi*

Description

This utility supplements `Rmpfr::integrateR` with the quadpack qagi method to handle integration involving infinity. Qagi is a transformation of $x/sigma = (1-t)/t$ for positive x, and $x/sigma = (t-1)/t$ for negative x. $t = 0$ is represented by `.Machine$double.eps`. This utility requires (a) `lower` or `upper` is `+/-Inf`; (b) `lower` and `upper` are of the same sign.

Usage

```
ecd.mpfr_qagi(object, f, lower, upper, ...,
  abs.tol = .Machine$double.eps^0.25, show.warning = TRUE)
```

Arguments

`object`	an object of ecd class
`f`	an R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
`lower`	numeric, the lower limit of integration. Can be infinite.
`upper`	numeric, the upper limit of integration. Can be infinite.
`...`	addtional arguments for `f`.
`abs.tol`	numeric, the suggested absolute tolerance.
`show.warning`	logical, to suppress warnings or not.

Value

The `integrate` object

Author(s)

Stephen H. Lihn

ecd.mpnum *Wrappers for ecd to maintain consistent type between mpfr and numeric*

Description

Primarily to make sure `x` is converted to mpfr vector if it is not, when `use.mpfr` is set.

Usage

```
ecd.mpnum(object, x)

ecd.ifelse(object, test, yes, no)

ecd.sapply(object, x, FUN, ...)
```

Arguments

object	an object of ecd class
x	a vector of numeric or mpfr.
test	logical, test of ifelse.
yes	return values for true elements of test
no	return values for false elements of test
FUN	the function to be applied to each element of x
...	optional arguments to FUN

Value

a numeric or mpfr vector

Author(s)

Stephen H. Lihn

---

ecd.ogf *Option generating function of ecd*

Description

Option generating function (OGF) of ecd. For call, it is integration of $(e^z - e^k)P(z)$ for z from k to Inf. For put, it is integration of $(e^k - e^z)P(z)$ for z from -Inf to k.

Usage

```
ecd.ogf(object, k, otype = "c", unit.sigma = FALSE, verbose = FALSE)
```

Arguments

object	an object of ecd class
k	a numeric vector of log-strike
otype	character, specifying option type: c or p.
unit.sigma	logical, transforming to unit sigma to achieve greater stability.
verbose	logical, display timing information, for debugging purpose.

Value

The option price normalized by underlying

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0, 0, sigma=0.01)
k <- seq(-0.1, 0.1, by=0.01)
ecd.ogf(d, k, "c")
```

ecd.pdf *Calculate the PDF of an ecd object*

Description

Calculate the PDF of an ecd object

Usage

```
ecd.pdf(object, x)
```

Arguments

object	an object of ecd class
x	numeric vector of x dimension

Value

numeric vector of the PDF

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-10, 10, by=1)
ecd.pdf(d,x)
```

ecd.polar *Polar constructor of ecd class*

Description

Construct an ecd class by specifying R and theta. They are converted to alpha and gamma, then passed onto the ecd constructor.

Usage

```
ecd.polar(R = NaN, theta = NaN, sigma = 1, beta = 0, mu = 0,
  cusp = 0, with.stats = TRUE, with.quantile = FALSE, bare.bone = FALSE,
  verbose = FALSE)
```

Arguments

R	numeric, the radius parameter. Default is NaN.
theta	numeric, the angle parameter. Default: NaN.
sigma	numeric, the scale parameter. Must be positive. Default: 1.
beta	numeric, the skewness parameter. Default: 0.
mu	numeric, the location parameter. Default: 0.
cusp	logical, indicate type of cusp (0,1,2).
with.stats	logical, also calculate statistics, default is TRUE.
with.quantile	logical, also calculate quantile data, default is FALSE.
bare.bone	logical, skip both const and stats calculation, default is FALSE. This for debug purpose for issues on integrating $e^y(x)$.
verbose	logical, display timing information, for debugging purpose, default is FALSE.

Value

The ecd class

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd.polar(R=1, theta=0.5*pi)
```

---

ecd.rational *Utility to convert a numeric to a rational*

Description

Convert a numeric x to rational p/q, which is then used for polynomial construction

Usage

```
ecd.rational(x)
```

Arguments

x	numeric

Value

a vector of two integers, representing numerator and denominator

Examples

```
pq <- ecd.rational(2.5)
```

ecd.read_csv_by_symbol *Read csv file of sample data*

Description

This is a helper utility to read sample csv file into data frame. The main use for external users is to read the option data since it has a different format than other price timeseries data.

Usage

```
ecd.read_csv_by_symbol(symbol = "dji")
```

Arguments

symbol	Character for the symbol of the time series. Default: dji

Value

The data.frame object

Examples

```
dji <- ecd.read_csv_by_symbol("dji")
spx <- ecd.read_csv_by_symbol("spxoption2")
```

ecd.read_symbol_conf *Read conf for sample data*

Description

Read conf for sample data

Usage

```
ecd.read_symbol_conf(symbol, conf_file = "conf/ecd-fit-conf.yml")
```

Arguments

symbol	Character. The symbol of sample data. Default: dji.
conf_file	File name fof symbol config, default to conf/ecd-fit-conf.yml

Value

the conf object

Examples

```
## Not run:
conf <- ecd.read_symbol_conf("dji")

## End(Not run)
```

ecd.sd *Standard deviation, variance, mean, skewness, and kurtosis of ecd*

Description

Convenience wrappers around ecd's stats data

Usage

```
ecd.sd(object)

ecd.var(object)

ecd.mean(object)

ecd.skewness(object)

ecd.kurt(object)

ecd.kurtosis(object)
```

Arguments

object	an object of ecd class

Value

numeric or mpfr

Examples

```
d <- ecd(-1,1)
ecd.sd(d)
ecd.var(d)
ecd.mean(d)
ecd.skewness(d)
ecd.kurt(d)
```

ecd.setup_const *Integration preprocessor for an ecd object*

Description

This is an internal helper function for ecd constructor Its main function is to determine const, const_left_x, and const_right_x during object construction.

Usage

```
ecd.setup_const(object, verbose = FALSE)
```

Arguments

object	An object of ecd class
verbose	logical, display timing information, for debugging purpose.

Value

list(const, const_left_x, const_right_x)

Author(s)

Stephen H. Lihn

Examples

```
ecd.toString(ecd(-1,1, sigma=0.1))
```

---

ecd.solve_cusp_asym *Trigonometric solution for asymmetric cusp distribution*

Description

The simplified trigonometric solution for $x^2 = -y^3 - beta * x * y$

Usage

```
ecd.solve_cusp_asym(x, beta)
```

Arguments

x	Array of x dimension
beta	the skew parameter

Value

Array of y

Examples

```
x <- seq(-100,100,by=0.1)
y <- ecd.solve_cusp_asym(x, beta=0.5)
```

ecd.stats *Compute statistics of an ecd object*

Description

Compute statistics for m1, m2, m3, m4, mean, var, skewness, kurtosis. This is used as part of ecd constructor.

Usage

```
ecd.stats(object, asymp.q = NULL, verbose = FALSE)
```

Arguments

object	an object of ecd class
asymp.q	If specified, a length-one numeric as asymptotic quantile for the asymptotic statistics. There is a wrapper in ecd.asymp_stats
verbose	logical, display timing information, for debugging purpose.

Value

a list of m1, m2, m3, m4, mean, var, skewness, kurtosis

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(1,1)
ecd.stats(d)
```

ecd.toString *String representation of ecd*

Description

A string representation of an ecd object. Can be used for warning or error.

Usage

```
ecd.toString(object, full = FALSE)
```

Arguments

object	An object of ecd class
full	logical, indicating if long form (multiple lines) should be rendered.

Value

character

Examples

```
ecd.toString(ecd(-1,1, sigma=0.1))
```

`ecd.ts_lag_stats` *Lag statistics on timeseries of log returns*

Description

Lag statistics on log returns are added to the xts attributes. It takes a vector of lags and calculates the mean, stdev, var, skewness, and kurtosis for cumulative log returns of each lag. The data is stored as a list of vectors under `lagstats` attribute. Be aware this function uses multicore lapply.

Usage

```
ecd.ts_lag_stats(ts = "dji", lags, absolute = FALSE)
```

Arguments

`ts`	the xts object from sample data. The ts must have the logr column. If a string is given, it will be replaced with sample data of the symbol.
`lags`	a numeric vector of integers greater than 0.
`absolute`	logical, if `TRUE`, statistics calculated on absolute log returns. Default: `FALSE`.

Value

The xts object containing `lagstats` attribute

Examples

```
## Not run:
dji <- ecd.ts_lag_stats(ecd.data("dji"), 2)

## End(Not run)
```

ecd.uniroot *Uniroot wrapper*

Description

This function wraps ordinary uniroot and unirootR (from Rmpfr) to the same interface.

Usage

```
ecd.uniroot(f, lower, upper, use.mpfr = FALSE,
  tol = .Machine$double.eps^0.25, maxiter = 1000)
```

Arguments

f	the function for which the root is sought.
lower, upper	the lower and upper end points of the interval to be searched.
use.mpfr	logical, to use MPFR (default), or else uniroot in stats.
tol	the desired accuracy (convergence tolerance).
maxiter	the maximum number of iterations.

Value

uniroot result

Author(s)

Stephen H. Lihn

ecd.y0_isomorphic *The analytic solution of $y(0)$ via isomorphic mapping.*

Description

This utility can be called two ways: (a) specify R and theta; (b) provide the ecd object. But not at the same time.

Usage

```
ecd.y0_isomorphic(theta = NaN, R = 1, object = NULL)
```

Arguments

theta	numeric vector, the polar coordinate
R	numeric vector, the polar coordinate
object	optionally, a single ecd object

Value

the value of y(0)

Examples

```
t <- 45/180*pi
ecd.y0_isomorphic(t)
```

`ecdattr` *Constructor of* `ecdattr` *class for the Elliptic Database (ECDB)*

Description

Construct an `ecdattr` class by providing the required parameters. This object has one-to-one correspondence to the rows in `ECDATTR` table. This is used primarily as object wrapper for safe update to ECDB.

Usage

```
ecdattr(alpha, gamma = NaN, cusp = 0, use.mpfr = FALSE)
```

Arguments

`alpha`	numeric, must be an integer after multiplied by 1000000.
`gamma`	numeric, must be an integer after multiplied by 1000000. `NaN` if cusp is 1.
`cusp`	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
`use.mpfr`	logical, whether to use mpfr for ecd object, default is `FALSE`.

Value

an object of `ecdattr` class

Examples

```
a <- ecdattr(1,1)
b <- ecdattr(alpha=1, cusp=1)
```

`ecdattr-class` *An S4 class to represent the* `ecdattr` *row in the Elliptic Database (ECDB)*

Description

The `ecdattr` class serves as an object-oriented interface between R and ECDB. This class is used extensively during the `bootstrap` process. A list of light-weight `ecdattr` objects is created first by `ecdattr.pairs` function, then the `ecdattr.enrich` function is invoked in parallel to calculate additional ecd attributes.

Slots

`call` the match.call slot

`alpha` numeric

`gamma` numeric. When cusp is 1, `gamma` is derived.

`cusp` numeric, representing type of cusp. Only 0 (default) and 1 are allowed.

`use.mpfr` logical, whether to use mpfr for ecd object.

`enriched` logical. If `TRUE`, it indicates the object has been enriched with ecd attributes.

`alpha_m` numeric, `alpha`*1000000.

`gamma_m` numeric, `gamma`*1000000.

`ecd` an object of ecd class.

`attr` list of attributes. They are `NULL` protected for SQLite.

`ecdattr.enrich` *Enrich a basic* `ecdattr` *object*

Description

It takes a basic `ecdattr` object, enrich it with ecd attributes. This function is computationally heavy. So the objects are often wrapped in a list and computed via `parallel::mclapply`.

Usage

```
ecdattr.enrich(p)
```

Arguments

`p`	a basic `ecdattr` object

Value

an enriched `ecdattr` object

`ecdattr.pairs` *Create a list of basic* `ecdattr` *objects*

Description

The list is created by the Cartesian product between `alpha` and `gamma`. This contains the data points of a rectangular area defined by `alpha,gamma`. If `cusp` is 1, data points are on the critical line specified by `alpha`.

Usage

```
ecdattr.pairs(alpha, gamma, cusp = 0, use.mpfr = FALSE)
```

Arguments

alpha, gamma	numeric vectors
cusp	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
use.mpfr	logical, whether to use mpfr for ecd object, default is FALSE.

Value

a list of basic ecdattr objects.

---

ecdattr.pairs_polar *Create a list of basic* ecdattr *objects in polar coordinate*

Description

The list is created by the Cartesian product between R and theta. This contains the data points of a circular area defined by R, theta. If cusp is 1, data points are on the critical line specified by R.

Usage

```
ecdattr.pairs_polar(R, theta, cusp = 0, use.mpfr = FALSE)
```

Arguments

R, theta	numeric vectors
cusp	numeric, representing type of cusp. Only 0 (default) and 1 are allowed.
use.mpfr	logical, whether to use mpfr for ecd object, default is FALSE.

Value

a list of basic ecdattr objects.

---

ecdb *Constructor of ecdb class for the elliptic database*

Description

Construct an ecdb class by providing the required parameters. The default is to use the internal database location. But the internal db is limited in size. The the elliptic database stores the stdev, kurtosis, discriminant, j-invariant, and ellipticity. for alpha and gamma between -100 and 100. Step size is 1 for -100 to 100; 0.25 for -50 to 50; 0.1 for -10 to 10; 0.025 between -6 and 1. Speical lines with step size of 0.001 for j0 and j1728 between -10 and 10; 0.01 for kmax and critical between 0 and 100. For asym1X, step size is 10 from 100 to 1000. For asym2X, step size is 100 from 1000 to 10000. For asym3X, step size is 1000 from 10000 to 60000. For polar-q1, step size is 0.025 from 0 to 20 for log2(R), and integer angles, 0-89.

Usage

```
ecdb(file = NULL, newdb = FALSE)
```

Arguments

file	Character, the full path to an elliptic database. Use "internal" to force the usage of the internal db.
newdb	Logical. If TRUE, remove existing db and create a new one. Default: FALSE.

Value

An object of ecdb class

Examples

```
db <- ecdb("internal")
```

---

ecdb-class *setClass for ecdb class*

Description

setClass for ecdb class

Slots

`call` the match.call slot

`file` character, the full path to an elliptic database.

`conn` an object of SQLiteConnection class.

`is.internal` logical, whether the connected db is internal.

`conf` list of configuration for data generation assigned by the constructor. Typical user should not have to modify this list unless you need to generate more data for advanced research.

Author(s)

Stephen H-T. Lihn

---

ecdb.dbSendQuery *Send query to the elliptic database*

Description

This API is used for write operations such as CREATE and INSERT.

Usage

```
ecdb.dbSendQuery(db, statement, ...)
```

Arguments

db	an object of ecdb class
statement	character, the SQL statement
...	database-specific parameters may be specified here

Value

a result set object

Author(s)

Stephen H-T. Lihn

---

ecdb.protectiveCommit *Protective commit*

Description

Protective commit after sending query to the elliptic database.

Usage

```
ecdb.protectiveCommit(db)
```

Arguments

db	an object of ecdb class

Value

The db object

Author(s)

Stephen H-T. Lihn

---

ecdq *Constructor of ecdq class*

Description

Construct an ecdq class by providing the required parameters.

Usage

```
ecdq(ecd, verbose = FALSE)
```

Arguments

ecd	An object of ecd class
verbose	logical, display timing information, for debugging purpose.

Value

An object of ecdq class

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
    d <- ecd()
    dq <- ecdq(d)

## End(Not run)
```

---

ecdq-class *setClass for ecdq class*

Description

setClass for ecdq class, the quantile generator

Slots

`call` the match.call slot

`xseg.from,xseg.to` numeric vectors. The from and to for each x segment.

`cseg.from,cseg.to` numeric vectors. The from and to for each cdf segment.

`cseg.min,cseg.max` numeric. The min and max of cdf segments.

`N_seg` numeric. Number of segments.

`cdf.fit` A vector of `lm` object, one for each segment.

`x_left_tail,x_right_tail` numeric. The starting x of left and right tails.

`fit.left,fit.right` objects of `lm` class for fitting the tails.

`conf` list of miscelaneous configurations. For debugging purpose.

---

ecld *Constructor of ecld class*

Description

Construct an ecld class by providing the required parameters. The default is the standard symmetric cusp distribution. The default also doesn't calculate any ecd extension. `ecld.from` allows you to pass the parameters from an existing ecd object. `ecld.validate` checks if an object is ecld class.

Usage

```
ecld(lambda = 3, sigma = 1, beta = 0, mu = 0, with.ecd = FALSE,
  with.mu_D = FALSE, with.RN = FALSE, is.sged = FALSE, verbose = FALSE)

ecld.from(object, with.ecd = FALSE, with.mu_D = FALSE, with.RN = FALSE,
  verbose = FALSE)

ecld.validate(object, sged.allowed = FALSE, sged.only = FALSE)
```

Arguments

lambda	numeric, the lambda parameter. Must be positive. Default: 3.
sigma	numeric, the scale parameter. Must be positive. Default: 1.
beta	numeric, the skewness parameter. Default: 0.
mu	numeric, the location parameter. Default: 0.
with.ecd	logical, also calculate the ecd object, default is FALSE.
with.mu_D	logical, also calculate the risk-neutral drift, default is FALSE. If TRUE, this flag supercedes with.ecd. Also mu must set to zero.
with.RN	logical, also calculate the risk-neutral ecd object, default is FALSE. If TRUE, this flag supercedes with.mu_D.
is.sged	logical, if TRUE, interpret parameters as SGED.
verbose	logical, display timing information, for debugging purpose, default is FALSE.
object	an object of ecld class
sged.allowed	logical, used in ecld.validate to indicate if the function allows SGED.
sged.only	logical, used in ecld.validate to indicate if the function is only for SGED.

Value

an object of ecld class

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld()
ld <- ecld(2, 0.01)
```

---

ecld-class *An S4 class to represent the lambda distribution*

Description

The ecld class serves as an object-oriented interface for the lambda distribution. The ecld prefix will also be used as the namespace for many analytic formulai derived in lambda distribution, especially when lambda = 1,2,3. Because of the extensive use of analytic formulai and enhanced precision through the unit distribution, MPFR is not needed in most cases. This makes option pricing calculation in ecld much faster than its counterpart built on the more general-purpose ecd library.

Slots

`call` the match.call slot

`lambda` numeric

`sigma` numeric

`beta` numeric

`mu` numeric

`use.mpfr` logical, whether to use mpfr for ecld object. If any of the above parameters is mpfr, then this flag is set to `TRUE`.

`is.sged` logical, if `TRUE`, interpret parameters as SGED.

`ecd` the companion object of ecd class (optional)

`mu_D` the risk-neutral drift (optional)

`ecd_RN` the risk-neutral companion object of ecd class (optional)

`status` numeric, bitmap recording the state of the calculation layers. 1: bare bone; 2: ecd; 4: mu_D; 8: ecd_RN

Author(s)

Stephen H. Lihn

`ecld.cdf` *CDF and CCDF of ecld*

Description

The analytic solutions for CDF and CCDF of ecld, if available. `ecld.cdf_gamma` is a sub-module with the CDF expressed as incomplete gamma function. SGED is supported only in `ecld.cdf` and `ecld.ccdf`.

Usage

```
ecld.cdf(object, x)

ecld.ccdf(object, x)

ecld.cdf_integrate(object, x)

ecld.cdf_gamma(object, x)
```

Arguments

`object`	an object of ecld class
`x`	a numeric vector of `x`

Value

The CDF or CCDF vector

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
ecld.cdf(ld,x)
```

---

ecld.const *Analytic solution of the normalization constant for lambda distribution*

Description

The normalization constant C. SGED is supported.

Usage

```
ecld.const(object)
```

Arguments

object	an object of ecld class

Value

numeric

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(3)
ecld.const(ld)
```

---

ecld.gamma *Incomplete gamma function and asymptotic expansion*

Description

ecld.gamma is the wrapper for incomplete gamma function $\Gamma(s,x)$. It is mainly to wrap around pgamma. And ecld.gamma_hgeo is the asymptotic expansion of $\Gamma(s,x)$ using hypergeometric series, $e^{-x}x^{s-1}{}_2F_0(1,1-s;;-1/x)$. It is mainly used in for star OGF $L^*(k;\lambda)$. ecld.gamma_2F0 is simply ${}_2F_0(1,1-s;;-1/x)$, which is used in the star OGF expansion.

Usage

```
ecld.gamma(s, x = 0)

ecld.gamma_hgeo(s, x, order)

ecld.gamma_2F0(s, x, order)
```

Arguments

s	numeric
x	numeric
order	numeric, the order of the power series

Value

numeric

Author(s)

Stephen H-T. Lihn

---

ecld.imgf *Incomplete moment generating function (IMGF) of ecld*

Description

The analytic solutions for IMGF of ecld, if available. Note that, by default, risk neutrality is honored. However, you must note that when fitting market data, this is usually not true. SGED is supported.

Usage

```
ecld.imgf(object, k, otype = "c", RN = TRUE)

ecld.imgf_gamma(object, k, otype = "c", RN = TRUE)

ecld.imgf_integrate(object, k, otype = "c", RN = TRUE)
```

Arguments

object	an object of ecld class
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.
RN	logical, use risk-neutral assumption for mu_D

Value

numeric, incomplete MGF

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01)
ecld.imgf(ld,0)
```

`ecld.imnt` *Incomplete moment (imnt) of ecld*

Description

The analytic solutions for imnt of ecld, if available. Note that, by default, risk neutrality is honored. `ecld.imnt_sum` provides an alternative method to calculate IMGF.

Usage

```
ecld.imnt(object, ki, order, otype = "c")

ecld.imnt_integrate(object, ki, order, otype = "c")

ecld.imnt_sum(object, ki, order, otype = "c")
```

Arguments

`object`	an object of ecld class
`ki`	numeric vector of normalized log-strike, `(k-mu)/sigma`
`order`	numeric. Order of the moment to be computed. For `ecld.imnt_sum`, this is the maximum order to be truncated. For small sigma at lambda=3, this can be simply 2. If `Inf`, the slope truncation procedure will be used to determine the maximum order. However, due to the numeric limit of `pgamma`, it is capped at 100.
`otype`	character, specifying option type: `c` (default) or `p`.

Value

numeric vector

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
ki <- seq(-0.1, 0.1, by=0.01)
ecld.imnt(ld,ki, 1)
```

`ecld.ivol_ogf_star` *Calculate implied volatility using star OGF and small sigma formula*

Description

Calculate implied volatility using star OGF and small sigma formula. Only the postive ki and Lc is supported. SGED is not supported yet.

Usage

```
ecld.ivol_ogf_star(object, ki, epsilon = 0, otype = "c",
  order.local = Inf, order.global = Inf, ignore.mu = FALSE)
```

Arguments

`object`	an object of ecld class
`ki`	a numeric vector of log-strike
`epsilon`	numeric, small asymptotic premium added to local regime
`otype`	option type
`order.local`	numeric, order of the hypergeometric series to be computed for local regime. Default is `Inf`, use the incomplete gamma. When it is `NaN`, `L*` value is suppressed.
`order.global`	numeric, order of the hypergeometric series to be computed for global regime. Default is `Inf`, use the incomplete gamma.
`ignore.mu`	logical, ignore `exp(mu)` on both sides, default is `FALSE`.

Value

The state price of option in star OGF terms. For `ecld.ivol_ogf_star`, it is σ_1.

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.001)
ecld.ivol_ogf_star(ld, 0)
```

ecld.mgf_term *The term structure of ecld symmetric MGF*

Description

ecld.mgf_term and ecld.mgf_diterm are the term and derivative of the term by order (n) in the summation of MGF. ecld.mgf_trunc uses ecld.mgf_diterm to locate the truncation of MGF terms. ecld.mgf_trunc_max_sigma locates the maximum sigma that keeps MGF finite for each lambda. SGED is supported.

Usage

```
ecld.mgf_term(object, order, t = 1)

ecld.mgf_diterm(object, order, t = 1)

ecld.mgf_trunc(object, t = 1)

ecld.mgf_trunc_max_sigma(object, order = 1)
```

Arguments

object	an object of ecd class
order	numeric. Order of the term (moment)
t	numeric, for MGF

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3, sigma=0.01*ecd.mp1)
ecld.mgf_trunc(ld)
```

ecld.moment *The moments and MGF of ecld*

Description

Compute the moments and MGF of ecld for mu=0 (centered), via analytical result whenever is available. SGED is supported.

Usage

```
ecld.moment(object, order, ignore.mu = TRUE)

ecld.mgf(object, t = 1)
```

Arguments

object	an object of ecd class
order	numeric, order of the moment to be computed
ignore.mu	logical, disgard mu; otherwise, stop if mu is not zero.
t	numeric, for MGF

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(lambda=3, sigma=0.01*ecd.mp1)
ecld.moment(ld, 2)
ecld.mgf(ld)
```

---

ecld.mpnum *Wrappers for ecld to maintain consistent type between mpfr and numeric*

Description

Primarily to make sure x is converted to mpfr vector if it is not, when use.mpfr is set.

Usage

```
ecld.mpnum(object, x)

ecld.ifelse(object, test, yes, no)

ecld.sapply(object, x, FUN, ...)

ecld.mclapply(object, x, FUN, ...)
```

Arguments

object	an object of ecd class
x	a vector of numeric or mpfr.
test	logical, test of ifelse.
yes	return values for true elements of test
no	return values for false elements of test
FUN	the function to be applied to each element of x
...	optional arguments to FUN

Value

a numeric or mpfr vector

Author(s)

Stephen H-T. Lihn

---

ecld.mu_D *mu_D of ecld*

Description

The analytic solutions for risk-neutral drift. If analytic form doesn't exist, it uses integral of unit distribution. This is different from ecld.mgf where series summation is used.

Usage

```
ecld.mu_D(object, validate = TRUE)
```

Arguments

object	an object of ecld class
validate	logical, if true (default), stop when the result is NaN or infinite.

Value

numeric

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
ecld.mu_D(ld)
```

ecld.ogf *Option generating function (OGF) of ecld*

Description

The analytic solutions for OGF of ecld, if available. Note that, by default, risk neutrality is honored. However, you must note that when fitting market data, this is usually not true.

Usage

```
ecld.ogf(object, k, otype = "c", RN = TRUE)

ecld.ogf_integrate(object, k, otype = "c", RN = TRUE)

ecld.ogf_gamma(object, k, otype = "c", RN = TRUE)

ecld.ogf_imnt_sum(object, k, order, otype = "c", RN = TRUE)

ecld.ogf_log_slope(object, k, otype = "c", RN = TRUE)
```

Arguments

object	an object of ecld class
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.
RN	logical, use risk-neutral assumption for mu_D
order	numeric, order of the moment to be computed

Value

The state price of option

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
k <- seq(-0.1, 0.1, by=0.05)
ecld.ogf(ld,k)
```

ecld.ogf_star *Star OGF of ecld*

Description

The star OGF of ecld is the limiting OGF for small sigma. It only depends on the normalized k and lambda. Its dependency on sigma and mu is removed. SGED is not supported yet.

Usage

```
ecld.ogf_star(object, ki)

ecld.ogf_star_hgeo(object, ki, order = 4)

ecld.ogf_star_exp(object, ki, order = 3)

ecld.ogf_star_gamma_star(object, ki, order = 6)
```

Arguments

object	an object of ecld class
ki	a numeric vector of log-strike
order	numeric, order of the hypergeometric series to be computed

Value

The state price of option in star OGF terms.

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.001*ecd.mp1)
ki <- seq(1, 5, by=1)
ecld.ogf_star(ld, ki)
```

ecld.op_V *The O, V, U operators in option pricing model*

Description

The O operator takes a vector of implied volatility $\sigma_1(k)$ and transforms them to a vector of option state prices. The V operator takes a vector of option state prices and transforms them to a vector of implied volatility $\sigma_1(k)$. The U operator calculates the log-slope of the option prices.

Usage

```
ecld.op_V(L, k, otype = "c", stop.on.na = FALSE, use.mc = TRUE)

ecld.op_O(sigma1, k, otype = "c")

ecld.op_U_lag(L, k, sd, n = 2)
```

Arguments

L	a vector of option state prices
k	a numeric vector of log-strike
otype	character, specifying option type: c (default) or p.
stop.on.na	logical, to stop if fails to find solution. Default is to use NaN and not stop.
use.mc	logical, to use mclapply (default), or else just use for loop. For loop option is typically for debugging.
sigma1	a vector of implied volatility (without T)
sd	numeric, the stdev of the distribution. Instead, if an ecld or ecd object is provided, the stdev will be calculated from it.
n	numeric, number of lags in ecld.op_U_lag.

Value

a numeric vector

Author(s)

Stephen H. Lihn

---

ecld.pdf *Calculate the PDF of an ecld object*

Description

Calculate the PDF of an ecld object

Usage

```
ecld.pdf(object, x)
```

Arguments

object	an object of ecd class
x	numeric vector of x dimension

Value

numeric vector of the PDF

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(lambda=3)
x <- seq(-10, 10, by=1)
ecld.pdf(ld,x)
```

---

ecld.sd *Compute statistics analytically for an ecld object*

Description

Compute statistics for mean, var, skewness, kurtosis, from the known analytical result. SGED is supported.

Usage

```
ecld.sd(object)

ecld.var(object)

ecld.mean(object)

ecld.skewness(object)

ecld.kurtosis(object)

ecld.kurt(object)
```

Arguments

object	an object of ecld class

Value

numeric or mpfr

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3)
ecld.sd(ld)
ecld.var(ld)
ecld.mean(ld)
ecld.skewness(ld)
ecld.kurt(ld)
```

`ecld.sged_const` *The integral solutions of SGED*

Description

These integrals are mainly used as validation to analytic solutions. If you must use them, be mindful of their slower speeds.

Usage

```
ecld.sged_const(object)

ecld.sged_cdf(object, x)

ecld.sged_moment(object, order)

ecld.sged_mgf(object, t = 1)

ecld.sged_imgf(object, k, t = 1, otype = "c")

ecld.sged_ogf(object, k, otype = "c")
```

Arguments

`object`	an sged object of ecld class
`x`	a numeric vector of `x`
`order`	numeric, order of the moment to be computed
`t`	numeric, for MGF and IMGF
`k`	a numeric vector of log-strike
`otype`	character, specifying option type: `c` (default) or `p`.

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(3)
ecld.const(ld)
```

ecld.solve *Analytic solution for $y(x)$ in lambda distribution*

Description

Analytic solution for $y(x)$ if available. $ecld.laplace_B$ is a utility function for the slopes of a skew Laplace distribution at lambda=2: B^+ and B^- with $B^0/2 = B^+ + B^-$. If `sigma` is provided, B notation is expanded for IMGF where $B^+_\sigma B^-_\sigma = \exp(\mu_D)$. SGED is supported.

Usage

```
ecld.solve(a, b, ...)

ecld.laplace_B(beta, sgn = 0, sigma = 0)
```

Arguments

`a`	an object of ecld class
`b`	a vector of x values
`...`	Not used. Only here to match the generic signature.
`beta`	the skew parameter
`sgn`	sign of $-1, 0, +1$
`sigma`	the scale parameter, optional

Value

A vector for $y(x)$

Author(s)

Stephen H. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
ecld.solve(ld,x)
```

ecld.y_slope *Analytic solution for the slope of $y(x)$ in lambda distribution*

Description

Analytic solution for the slope of $y(x)$ if available. $ecld.y_slope_trunc$ calculates the MGF truncation point where $dy/dx + t = 1$. SGED is supported.

Usage

```
ecld.y_slope(object, x)

ecld.y_slope_trunc(object, t = 1)
```

Arguments

object	an object of ecld class
x	a vector of x values
t	numeric, for MGF truncation

Value

numeric

Author(s)

Stephen H-T. Lihn

Examples

```
ld <- ecld(sigma=0.01*ecd.mp1)
x <- seq(-0.1, 0.1, by=0.01)
ecld.y_slope(ld,x)
ecld.y_slope_trunc(ld)
```

---

ecldOrEcd-class *The ecldOrEcd class*

Description

The S4 class union of ecld and ecd, primarily used to define slot in ecop.opt class. Its usage is rather cumbersome, so the end user should avoid it as much as possible.

---

ecop-class *An S4 class to represent the top-level option model*

Description

The ecop class serves as an object-oriented container for the option pricing model. It does have a specific purpose at the moment - that is, to produce all the data for the charts of the paper, based on CBOE data structure. Therefore, user may not find it general enough. That probably will be the case for the time being until more popularity calls for a more generic container.

Slots

`call` the match.call slot

`conf` list, configuration

`key` character

`symbol` character

`datadate` Date

`days` numeric, days between datadate and expiry date

`ttm` numeric, time to maturity in days/365

`int_rate` numeric

`div_yield` numeric

`put_data` the put data of ecop.opt class

`call_data` the call data of ecop.opt class

`put_conf` list, the put configuration

`call_conf` list, the call configuration

Author(s)

Stephen H-T. Lihn

---

ecop.bs_implied_volatility

Implied volatility of Black-Sholes model

Description

This is the standard library to calculate implied volatility σ_{BS} in Black-Sholes model. There is no external dependency on elliptic distribution.

Usage

```
ecop.bs_implied_volatility(V, K, S, ttm, int_rate = 0, div_yield = 0,
  otype = "c", stop.on.na = FALSE, use.mc = TRUE)
```

Arguments

`V`	numeric vector of option prices
`K`	numeric vector of strike prices
`S`	length-one numeric for underlying price
`ttm`	length-one numeric for time to maturity, in the unit of days/365.
`int_rate`	length-one numeric for risk-free rate, default to 0.
`div_yield`	length-one numeric for dividend yield, default to 0.
`otype`	character, specifying option type: `c` or `p`.
`stop.on.na`	logical, to stop if fails to find solution. Default is to use NaN and not stop.
`use.mc`	logical, to use mclapply (default), or else just use for loop. For loop option is typically for debugging.

Value

The implied volatity σ_{BS}.

Examples

```
V <- c(1.8, 50)
K <- c(2100, 2040)
S <- 2089.27
T <- 1/365
y <- 0.019
ecop.bs_implied_volatility(V, K, S, ttm=T, div_yield=y, otype="c")
# expect output of 12.8886% and 29.4296%
```

---

ecop.bs_option_price *Calculate option price from implied volatility in Black-Sholes model*

Description

This is the standard library to calculate option price from implied volatility σ_{BS} in Black-Sholes model. There is no external dependency on elliptic distribution.

Usage

```
ecop.bs_option_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0,
  otype = "c")

ecop.bs_call_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0)

ecop.bs_put_price(ivol, K, S, ttm, int_rate = 0, div_yield = 0)
```

Arguments

ivol	numeric vector of implied volatility
K	numeric vector of strike prices
S	length-one numeric for underlying price
ttm	length-one numeric for time to maturity, in the unit of days/365.
int_rate	length-one numeric for risk-free rate, default to 0.
div_yield	length-one numeric for dividend yield, default to 0.
otype	character, c or p. Default is c.

Value

The call/put prices

Examples

```
ivol <- c(0.128886, 0.294296)
K <- c(2100, 2040)
S <- 2089.27
T <- 1/365
y <- 0.019
ecop.bs_option_price(ivol, K, S, ttm=T, div_yield=y, otype="c")
# expect output of c(1.8, 50)
```

`ecop.from_symbol_conf` *Constructor of ecop class by read conf for option sample data*

Description

Read conf for option sample data and fitting parameters

Usage

```
ecop.from_symbol_conf(key, conf_file = "conf/ecop-fit-conf.yml",
  conf_data = NULL)

ecop.read_symbol_conf(key, conf_file = "conf/ecop-fit-conf.yml")

ecop.build_opt(ecop, df, otype)
```

Arguments

`key`	character. The top-level key in conf
`conf_file`	file name fof symbol config, default to conf/ecld-fit-conf.yml
`conf_data`	optionally feed config through a list. If this is not null, this takes priority and `conf_file` will be ignored.
`ecop`	an ecop object with conf
`df`	dataframe of a single closing date and time to maturity
`otype`	option type

Value

the ecop object

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
    conf <- ecop.read_symbol_conf("spx2_1d")
    op <- ecop.from_symbol_conf("spx2_1d")

## End(Not run)
```

ecop.opt-class *An S4 class to represent the option data and model calculation*

Description

The ecop.opt class serves as an object-oriented container for the type-specific (p or c) option data.

Slots

call the match.call slot

otype character, option type

range.from numeric, starting price range

range.to numeric, ending price range

momentum numeric, momentum for tranlation (T) operator

epsilon numeric, asymptotic premium

k_cusp numeric, the suggested cusp location for poly fit of prices

ecldOrEcd the ecld/ecd class to calculate theoretical values in local regime

S underyling price, this can be overriden by conf

S_raw underyling price (before override)

strike strike price

k log-strike price

V_last last option price

V_bid bid option price

V_ask ask option price

V finalized option price (likely mid-point)

Author(s)

Stephen H. Lihn

ecop.plot_option *Plot option chain charts using conf from option sample data*

Description

This utility produces standardized plots of 3. The first plot is the option state price and fits. The second plot is the log-slope of option state prices and fits. The thrid plot is the implied volatility and fits.

Usage

```
ecop.plot_option(object, otype, simulate = TRUE)
```

Arguments

object	an ecop object with conf
otype	option type
simulate	logic, if TRUE, simulate according to lambda transformation and lambda distribution.

Value

The ecop.opt object

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
    op <- ecop.from_symbol_conf("spx2_1d")
    par(mfcol=c(3,2))
    ecop.plot_option(op, otype="c")
    ecop.plot_option(op, otype="p")

## End(Not run)
```

---

ecop.polyfit_option *Poly fit on option prices*

Description

The poly fits on logarithm of option prices are performed for each side of the suggested cusp (specified by k.cusp). This utiility is used mainly to remove the market data noise for the calculation of log-slope of option prices.

Usage

```
ecop.polyfit_option(k, V, k.cusp, k.new, degree.left = 6, degree.right = 6)
```

Arguments

k	numeric, vector of log-strike
V	numeric, vectors of option prices
k.cusp	length-one numeric, the suggested cusp location
k.new	numeric, vector of log-strike to evaluate the poly fit
degree.left	length-one numeric, specifying the degree of poly fit for the left tail
degree.right	length-one numeric, specifying the degree of poly fit for the right tail

Value

The state prices from the poly fit

Author(s)

Stephen H-T. Lihn

`ecop.read_csv_by_symbol` *Read option data csv*

Description

Read option data csv into dataframe, enriched with `Date, expiration_date, days`.

Usage

```
ecop.read_csv_by_symbol(symbol)
```

Arguments

`symbol`	character, option data symbol

Value

dataframe

Author(s)

Stephen H-T. Lihn

Examples

```
df <- ecop.read_csv_by_symbol("spxoption2")
```

`ellipticity.ecd` *Ellipticity of ecd object*

Description

Ellipticity of ecd object, defined as half of the distance between the two elliptic points.

Usage

```
## S3 method for class 'ecd'
ellipticity(object, tol = 1e-04)

ellipticity(object, tol = 1e-05)

## S4 method for signature 'ecd'
ellipticity(object, tol = 1e-04)
```

Arguments

object	An object of ecd class
tol	Numeric, the tolerance of precision during subdivision. Default: 1e-4 of stdev.

Value

a list with 3 major numbers: xe1= negative x_e, xe2= positive x_e, avg= ellipticity

Examples

```
d <- ecd(0,1)
ellipticity(d)
```

---

history.ecdb *List of history in the Elliptic DB*

Description

List of unique history reflecting the bootstrap activities.

Usage

```
## S3 method for class 'ecdb'
history(object)

history(object)

## S4 method for signature 'ecdb'
history(object)
```

Arguments

object	an object of ecdb class.

Value

list of history

Author(s)

Stephen H-T. Lihn

`integrate_pdf.ecd` *Integrate a function with PDF of the distribution*

Description

Integrate a function with PDF of the distribution. The integration is seperated into three segments to ensure convergence.

Usage

```
## S3 method for class 'ecd'
integrate_pdf(object, f, lower, upper, ..., show.warning = TRUE,
  verbose = FALSE)

integrate_pdf(object, f, lower, upper, ...)

## S4 method for signature 'ecd'
integrate_pdf(object, f, lower, upper, ...,
  show.warning = TRUE, verbose = FALSE)
```

Arguments

`object`	An object of ecd class
`f`	An R function taking a numeric first argument and returning a numeric vector of the same length. Returning a non-finite element will generate an error.
`lower`	Numeric, the lower limit of integration. Can be infinite.
`upper`	Numeric, the upper limit of integration. Can be infinite.
`...`	Addtional arguments for `f`.
`show.warning`	logical, display warning messages.
`verbose`	logical, display timing information, for debugging purpose.

Value

A list of class "`integrate`".

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
integrate_pdf(d, function(x){x^2}, -Inf, Inf)
```

jinv.ecd *J-invariant of the elliptic curve $y(x)$*

Description

J-invariant of the elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
jinv(object, no.validate = FALSE)

jinv(object, no.validate = FALSE)

## S4 method for signature 'ecd'
jinv(object, no.validate = FALSE)
```

Arguments

object	an object of ecd class
no.validate	logical, if TRUE, don't validate presence of beta. Default is FALSE.

Value

the j-invariant

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd(1,1)
j <- jinv(d)
```

moment.ecd *Compute the moment of ecd via integration*

Description

Compute the moment of ecd via integration between -Inf and Inf. The asymp.lower and asymp.upper parameters are used for asymptotic statistics, to study the effect of finite observations.

Usage

```
## S3 method for class 'ecd'
moment(object, order, center = FALSE, asymp.lower = -Inf,
  asymp.upper = Inf, verbose = FALSE)

moment(object, order, center = FALSE, asymp.lower = -Inf,
  asymp.upper = Inf, verbose = FALSE)

## S4 method for signature 'ecd'
moment(object, order, center = FALSE, asymp.lower = -Inf,
  asymp.upper = Inf, verbose = FALSE)
```

Arguments

object	an object of ecd class
order	numeric. Order of the moment to be computed
center	logical. If set to TRUE, calculate central moments. Default: FALSE.
asymp.lower	numeric, lower bound for asymptotic statistics, default: -Inf.
asymp.upper	numeric, upper bound for asymptotic statistics, default: Inf.
verbose	logical, display timing information, for debugging purpose.

Value

Numeric. The moment.

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd()
moment(d, 2)
```

---

numericMpfr-class *The numericMpfr class*

Description

The S4 class union of numeric and mpfr, primarily used to define slots in ecd class. The use of MPFR does not necessarily increase precision. Its major strength in ecd is ability to handle very large numbers when studying asymptotic behavior, and very small numbers caused by small sigma when studying high frequency option data. Since there are many convergence issues with integrating PDF using native integrateR library, the ecd package adds many algorithms to improve its performance. These additions may decrease precision (knowningly or unknowningly) for the sake of increasing performance. More research is certainly needed in order to cover a vast range of parameter space!

plot_2x2.ecd *Standard 2x2 plot for sample data*

Description

Standard 2x2 plot for sample data

Usage

```
plot_2x2.ecd(object, ts, EPS = FALSE, eps_file = NA)

plot_2x2(object, ts, EPS = FALSE, eps_file = NA)

## S4 method for signature 'ecd'
plot_2x2(object, ts, EPS = FALSE, eps_file = NA)
```

Arguments

object	An object of ecd class.
ts	The xts object for the timeseries.
EPS	Logical, indicating whether to save the plot to EPS, default = FALSE
eps_file	File name for eps output

Examples

```
## Not run:
plot_2x2(d, ts)

## End(Not run)
```

quantilize.ecd *Add the quantile data to the ecd object*

Description

Add the quantile data to the ecd object if it is not created yet.

Usage

```
## S3 method for class 'ecd'
quantilize(object, show.warning = FALSE)

quantilize(object, show.warning = FALSE)

## S4 method for signature 'ecd'
quantilize(object, show.warning = FALSE)
```

Arguments

object	an object of ecd class
show.warning	logical, if TRUE, display a warning message. Default is FALSE.

Value

an object of ecd class with a newly generated ecdq object.

Author(s)

Stephen H-T. Lihn

Examples

```
## Not run:
   d <- ecd(-1,1)
   quantilize(d)

## End(Not run)
```

---

read.ecdb *Read API for the ecdb*

Description

Read ecdb into data.frame. This can be accomplished by either specifying the range of alpha,gamma or the cartesian product of alpha,gamma point by point, or both. If both are specified, it follows a similar logic as plot how x,y is scoped by xlim,ylim.

Usage

```
## S3 method for class 'ecdb'
read(object, alpha = NULL, gamma = NULL, alim = NULL,
  glim = NULL, cusp = 0, polar_ext = FALSE)

read(object, alpha = NULL, gamma = NULL, alim = NULL, glim = NULL,
  cusp = 0, polar_ext = FALSE)

## S4 method for signature 'ecdb'
read(object, alpha = NULL, gamma = NULL, alim = NULL,
  glim = NULL, cusp = 0, polar_ext = FALSE)
```

Arguments

object	an object of ecdb class
alpha, gamma	numeric vectors of points for cartesian product
alim, glim	length-two numeric vectors of min and max range
cusp	numeric. Type of cusp. Only 0 and 1 are allowed. If cusp=1, read cusp data on the critical line. Reading cusp data must be done from the alpha side. Default: 0.
polar_ext	logical, for polar coordinate extension: R, theta, angle. Default: FALSE.

Value

The data.frame from ECDATTR table.

`solve.ecd` *Solve the elliptic curve $y(x)$*

Description

Solve the elliptic curve $y(x)$ by constructing a cubic polynomial from ecd object. Then solve it and take the smallest real root.

Usage

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

## S4 method for signature 'ecd'
solve(a, b, ...)
```

Arguments

`a`	An object of ecd class
`b`	A vector of x values
`...`	Not used. Only here to match the generic signature.

Value

A vector of roots for $y(x)$

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.1)
y <- solve(d,x)
```

`solve_sym.ecd` *Analytic solution for a symmetric elliptic curve*

Description

Analytic solution for a symmetric elliptic curve $y(x)$

Usage

```
## S3 method for class 'ecd'
solve_sym(object, x)

solve_sym(object, x)

## S4 method for signature 'ecd'
solve_sym(object, x)
```

Arguments

object	an object of ecd class
x	array of x dimension

Value

array of y

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.01)
y <- solve_sym(d,x)
```

---

solve_trig.ecd *Trigonometric solution for a elliptic curve*

Description

Use Chebyshev trigonometry for a depressed cube to solve a elliptic curve $y(x)$.

Usage

```
## S3 method for class 'ecd'
solve_trig(object, x)

solve_trig(object, x)

## S4 method for signature 'ecd'
solve_trig(object, x)
```

Arguments

object	an object of ecd class
x	array of x dimension

Value

array of y

Author(s)

Stephen H-T. Lihn

Examples

```
d <- ecd()
x <- seq(-100,100,by=0.1)
y <- solve_trig(d,x)
```

summary.ecdb *Summary for the Elliptic DB (ECDB)*

Description

Summary for the Elliptic DB (ECDB)

Usage

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

summary(object, ...)

## S4 method for signature 'ecdb'
summary(object, ...)
```

Arguments

object	an object of ecdb class.
...	more arguments for summary. Currently not used.

Author(s)

Stephen H-T. Lihn

Examples

```
summary(ecdb())
```

write.ecdb *Write API for the ecdb for a list of basic ecdattr objects*

Description

It takes a list of basic ecdattr objects, enrich them in parallel, then save them to ecdb.

Usage

```
## S3 method for class 'ecdb'
write(x, object)

write(x, object)

## S4 method for signature 'list,ecdb'
write(x, object)
```

Arguments

x	a list of basic ecdattr objects
object	an object of ecdb class

Value

The row count

`y_slope.ecd` *Slope of* $y(x)$

Description

Slope of $y(x)$, that is, dy/dx.

Usage

```
## S3 method for class 'ecd'
y_slope(object, x)

y_slope(object, x)

## S4 method for signature 'ecd'
y_slope(object, x)
```

Arguments

`object`	an object of ecd class
`x`	a numeric vector of x dimension

Value

a numeric vector of dy/dx

Author(s)

Stephen H. Lihn

Examples

```
d <- ecd(0,1)
x <- seq(-20,20,by=0.01)
yp <- y_slope(d,x)
```

Index