# Package ‘fAsianOptions’

September 30, 2009

**Version** 2100.76

**Revision** 4057

**Date** 2009-09-28

**Title** EBM and Asian Option Valuation

**Author** Diethelm Wuertz and many others, see the SOURCE file

**Depends** R (>= 2.4.0), timeDate, timeSeries, fBasics, fOptions

**Suggests** RUnit

**Maintainer** Rmetrics Core Team <Rmetrics-core@r-project.org>

**Description** Environment for teaching “Financial Engineering and Computational Finance”

**NOTE** SEVERAL PARTS ARE STILL PRELIMINARY AND MAY BE CHANGED IN THE FUTURE. THIS TYPICALLY INCLUDES FUNCTION AND ARGUMENT NAMES, AS WELL AS DEFAULTS FOR ARGUMENTS AND RETURN VALUES.

**LazyLoad** yes

**LazyData** yes

**License** GPL (>= 2)

**URL** http://www.rmetrics.org

**Repository** CRAN

**Date/Publication** 2009-09-30 19:40:52

## R topics documented:

## `BesselFunctions` *Modified Bessel Functions*

### Description

A collection and description of special mathematical functions which compute the modified Bessel functions of integer order of the first and second kind as well as their derivatives.

The functions are:

| | |
|---|---|
| `BesselI` | modified Bessel function of the 1st Kind, |
| `BesselDI` | its derivative, |
| `BesselK` | the modified Bessel function of the 3nd Kind, |
| `BesselDK` | its derivative. |

### Usage

```
BesselI(x, nu, expon.scaled = FALSE)
BesselK(x, nu, expon.scaled = FALSE)
BesselDI(x, nu)
BesselDK(x, nu)
```

### Arguments

| | |
|---|---|
| `expon.scaled` | a logical; if TRUE, the results are exponentially scaled. |
| `nu` | an integer value greater or equal to zero, the integer order of the modified Bessel function. |
| `x` | a positive numeric value or a vector of positive numerical values. |

### Value

The functions return the values of the selected special mathematical function.

### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

### References

Abramowitz M., Stegun I.A. (1972); *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York, Dover Publishing.

Weisstein E.W. (2004); *MathWorld – A Wolfram Web Resource*, http://mathworld.wolfram.com

### Examples

```
## Bessel I0 and K0 -
   # Abramowitz-Stegun: Table 9.8, p. 416-422
   x = c(0.0, 0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50)
   data.frame(x, I = exp(-x)*BesselI(x, 0), K = exp(x)*BesselK(x, 0))
   # Compare with R's internal function:
   # data.frame(x, ratio = BesselI(x, 0) / besselI(x, 0))
   # data.frame(x, ratio = BesselK(x, 0) / besselK(x, 0))

## x = 0:
   c(BesselI(0, 0), BesselI(0, 1), BesselI(0, 2), BesselI(0, 5))
   # Compare with R's internal function:
   # c(besselI(0, 0), besselI(0, 1), besselI(0, 2), besselI(0, 5))
   c(BesselK(0, 0), BesselK(0, 1), BesselK(0, 2), BesselK(0, 5))
   # Compare with R's internal function:
   # c(besselK(0, 0), besselK(0, 1), besselK(0, 2), besselK(0, 5))

## Bessel I2 and K2 -
   # Abramowitz-Stegun: Table 9.8, p. 416-422
   x = c(0.0, 0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50)
   data.frame(x, I = BesselI(x, 2)/x^2, K = BesselK(x, 2)*x^2)
   # Compare with R's internal function:
   # data.frame(x, ratio = BesselI(x, 0) / besselI(x, 0))
   # data.frame(x, ratio = BesselK(x, 0) / besselK(x, 0))
   # data.frame(x, ratio = BesselI(x, 1) / besselI(x, 1))
   # data.frame(x, ratio = BesselK(x, 1) / besselK(x, 1))
   # data.frame(x, ratio = BesselI(x, 5) / besselI(x, 5))
   # data.frame(x, ratio = BesselK(x, 5) / besselK(x, 5))
   # data.frame(x, ratio = BesselI(x,50) / besselI(x,50))
   # data.frame(x, ratio = BesselK(x,50) / besselK(x,50))
```

---

## `EBMAsianOptions` *Exponential Brownian Motion Distributions*

---

### Description

A collection and description of functions used in the theory of exponential Brownian Motion and in the valuation of Asian options.

The functions for Moment matching and Series Expansions are:

| | |
|---|---|
| `MomentMatchedAsianOption` | Valuate moment matched option prices, |
| `... method="LN"` | Log-Normal Approximation of Levy, Turnbull and Wakeman, |
| `... method="RG"` | Reciprocal-Gamma Approximation of Milevski and Posner, |
| `... method="JI"` | Johnson Type I Approximation of Posner and Milevsky, |
| `MomentMatchedAsianDensity` | Valuate moment matched option densities, |
| `... method="LN"` | Log-Normal Approximation, |
| `... method="RG"` | Reciprocal-Gamma Approximation, |
| `... method="JI"` | Johnson Type I Approximation, |
| `GramCharlierAsianOption` | Calculate Gram-Charlier option prices. |

| | |
|---|---|
| `AsianOptionMoments` | Methods to calculate Asian Moments, |
| `... method="A"` | Moments from Abrahamson's Formula, |
| `... method="D"` | Moments from Dufresne's Formula, |
| `... method="TW"` | First 2 Moments from Turnbull-Wakeman, |
| `... method="T"` | Asymptotic Behavior after Tolmatz. |

| | |
|---|---|
| `ZhangAsianOption` | Asian option price by Zhang's 1D PDE, |
| `VecerAsianOption` | Asian option price by Vecer's 1D PDE. |

| | |
|---|---|
| `gGemanYor` | Function to be Laplace inverted, |
| `GemanYorAsianOption` | Asian option price by Laplace Inversion, |
| `gLinetzky` | Function to be integrated, |
| `LinetzkyAsianOption` | Asian option price by Spectral Expansion. |

| | |
|---|---|
| `BoundsOnAsianOption` | Lower and upper bonds on Asian calls, |
| `CurranThompsonAsianOption` | From Thompson's continuous limit, |
| `RogerShiThompsonAsianOption` | From Thompson's single integral formula, |
| `ThompsonAsianOption` | Thompson's upper bound, |
| `TolmatzAsianOption` | Lower Bound from Tolmatz' symptotics. |

| | |
|---|---|
| `CallPutParityAsianOption` | Call-Put parity Relation, |
| `WithDividendsAsianOption` | Adds dividends to Asian option formula. |

| | |
|---|---|
| `FuMadanWangTable` | Table from Fu, Madan and Wang's paper, |
| `FusaiTaglianiTable` | Table from Fusai und tagliani's paper, |
| `GemanTable` | Table from Geman's paper, |
| `LinetzkyTable` | Table from Linetzky's paper, |
| `ZhangTable` | Table from Zhang's paper, |
| `ZhangLongTable` | Long Table from Zhang's paper, |
| `ZhangShortTable` | Short Table from Zhang's paper. |

Sorry - The Documentation is still Uncomplete.

### Usage

```
MomentMatchedAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, table = NA, method = c("LN", "RG", "JI"))
MomentMatchedAsianDensity(x, Time = 1, r = 0.09, sigma = 0.30,
    method = c("LN", "RG", "JI"))
GramCharlierAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, table = NA, method = c("LN", "RG", "JI"))

AsianOptionMoments(M = 4, Time = 1, r = 0.045, sigma = 0.30, log = FALSE,
```

```
    method = c("A", "D", "TW", "T"))

ZhangAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, table = NA, correction = TRUE, nint = 800,
    eps = 1.0e-8, dt = 1.0e-10)
VecerAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, table = NA, nint = 800, eps = 1.0e-8,
    dt = 1.0e-10)

gGemanYor(lambda, S = 100, X = 100, Time = 1, r = 0.05, sigma = 0.30,
    log = FALSE, doplot = FALSE)
GemanYorAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, doprint = FALSE)
gLinetzky(x, y, tau, nu, ip = 0)
LinetzkyAsianOption(TypeFlag = c("c", "p"), S = 2, X = 2, Time = 1,
    r = 0.02, sigma = 0.1, table = NA, lower = 0, upper = 100,
    method = "adaptive", subdivisions = 100, ip = 0, doprint = TRUE,
    doplot = TRUE, ...)

BoundsOnAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30, table = NA, method = c("CT", "RST", "T"))
CurranThompsonAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100,
    Time = 1, r = 0.09, sigma = 0.30)
RogerShiThompsonAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100,
    Time = 1, r = 0.09, sigma = 0.30)
ThompsonAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30)
TolmatzAsianOption(TypeFlag = c("c", "p"), S = 100, X = 100, Time = 1,
    r = 0.09, sigma = 0.30)

CallPutParityAsianOption(TypeFlag = "p", Price = 8.828759, S = 100,
    X = 100, Time = 1, r = 0.09, sigma = 0.3, table = NA)
WithDividendsAsianOption(TypeFlag = "c", Dividends = 0.45, S = 100,
    X = 100, Time = 1, r = 0.09, sigma = 0.3,
    calculator = MomentMatchedAsianOption, method = "LN")

FuMadanWangTable()
FusaiTaglianiTable()
GemanTable()
LinetzkyTable()
ZhangTable()
ZhangLongTable()
ZhangShortTable()
```

## Arguments

`calculator` [WithDividendsAsianOption] -
the name of the function selecting the option calculator to be used.

`correction` [ZhangAsianOption] -
xxx.

`Dividends` [WithDividendsAsianOption] -
xxx.

`doplot` [gGemanYor][LinetzkyAsianOption] -
xxx.

`doprint` [GemanYorAsianOption][LinetzkyAsianOption] -
xxx.

`dt` [VecerAsianOption][ZhangAsianOption] -
xxx.

`eps` [VecerAsianOption][ZhangAsianOption] -
xxx.

`ip` [gLinetzky] -
xxx.

`lambda` [gGemanYor] -
xxx.

`log` [AsianOptionMoments][gGemanYor] -
xxx.

`lower, upper` [LinetzkyAsianOption] -
xxx.

`M` [*] -
xxx.

`method` [*] -
xxx.

`nint` [*] -
xxx.

`nu` [*] -
xxx.

`Price` [*] -
xxx.

`r` a numeric value, the annualized rate of interest; e.g. 0.25 means 25% pa.

`S` the asset price, a numeric value.

`sigma` a numeric value, the annualized volatility of the underlying security; e.g. 0.3 means 30% volatility pa.

`subdivisions` [*] -
xxx.

`table` [*] -
xxx.

`tau` [*] -
xxx.

`Time` a numeric value, the time to maturity measured in years; e.g. 0.5 means 6 months.

| | |
|---|---|
| `TypeFlag` | a character string either `"c"` for a call option or a `"p"` for a put option. |
| `x` | [*] -<br>xxx. |
| `X` | a numeric value, the exercise price. |
| `y` | [*] -<br>xxx. |
| `...` | [*] -<br>xxx. |

### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

### Examples

```
## Examples:
   # none ...
```

---

`EBMDistribution` *Exponential Brownian Motion Distributions*

---

### Description

A collection and description of distributions and related functions which are useful in the theory of exponential Brownian motion and Asian option valuation. The functions compute densities and probabilities for the log-Normal distribution, the Gamma distribution, the Reciprocal-Gamma distribution, and the Johnson Type-I distribution. Functions are made available for the compution of moments including the Normal, the log-Normal, the Reciprocal-Gamma, and the Asian-Option Density. In addition a function is given to compute numerically first and second derivatives of a given function.

The functions are:

| | |
|---|---|
| `dlognorm` | the log-Normal density and derivatives, |
| `plognorm` | the log-Normal, a synonyme for R's plnorm, |
| `dgam` | the Gamma density, a synonyme for R's dgamma, |
| `pgam` | the Gamma probability, a synonyme for R's pgamma, |
| `drgam` | the Reciprocal-Gamma density, |
| `prgam` | the Reciprocal-Gamma probability, |
| `djohnson` | the Johnson Type I density, |
| `pjohnson` | the Johnson Type I probability, |
| `mnorm` | the Moments of Normal density, |
| `mlognorm` | the Moments of log-Normal density, |
| `mrgam` | the Moments of reciprocal-Gamma density, |
| `masian` | the Moments of Asian Option density, |
| `derivative` | the First and second numerical derivative. |

### Usage

```
dlognorm(x, meanlog = 0, sdlog = 1, deriv = c(0, 1, 2))
plognorm(q, meanlog = 0, sdlog = 1)
dgam(x, alpha, beta)
pgam(q, alpha, beta, lower.tail = TRUE)
drgam(x, alpha, beta, deriv = c(0, 1, 2))
prgam(q, alpha, beta, lower.tail = TRUE)
djohnson(x, a = 0, b = 1, c = 0, d = 1, deriv = c(0, 1, 2))
pjohnson(q, a = 0, b = 1, c = 0, d = 1)

mnorm(mean = 0, sd = 1)
mlognorm(meanlog = 0, sdlog = 1)
mrgam(alpha = 1/2, beta = 1)
mjohnson(a, b, c, d)
masian(Time = 1, r = 0.045, sigma = 0.30)

derivative(x, y, deriv = c(1, 2))

dEBM(u, t = 1)
pEBM(u, t = 1)
d2EBM(u, t = 1)
dasymEBM(u, t = 1)
```

### Arguments

`a, b, c, d` [*johnson] -
the parameters of the Johnson Type I distribution. The default values are `a=1`, `b=1`, `c=0`, and `d=1`.

`alpha, beta` [*gam] -
the parameters of the Gamma distribution.

`deriv` an integer value, the degree of differentiation, either 0, 1 or 2.

`lower.tail` a logical, if `TRUE`, the default, then the probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

`mean, sd` [*lognorm] -
the parameters of the Normal distribution, the mean and the standard deviation respectively. The default values are `mean=0` and `sd=1`.

`meanlog, sdlog`
[*lognorm] -
the parameters of the Log Normal distribution, the mean and the standard deviation respectively. The default values are `mean=0` and `sd=1`.

`q` a real numeric value or vector.

`t` ...

`Time, r, sigma`
the parameters of the Asian Option distribution.

`u` ...

`x` a real numeric value or vector.

`y` [derivative] -
a real numeric value or vector, the function values from which to compute the first and second derivative.

### Value

The functions `d*` and `p*` return the values or numeric vectors of the density and probability of the the corresponding distribution.

The functions `m*` return a list with three elements, the values of the first four moments `rawMoments`, the values of the first four central moments `centralMoments`, and the skewness and kurtosis `fisher`, also called Fisher parameters.

The function `derivative` returns a list of two elemtes, `$x` and `$y`, where `$y($x)` is either the first or second derivative of `y(x)` as selected by the argument `deriv`.

### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

### Examples

```
## dlognorm -
   # Calculate Log-Normal Density and its Derivaties:
   x = exp(seq(-2.8, 1.2, length = 100))
   y0 = dlognorm(x, deriv = 0)
   y1 = dlognorm(x, deriv = 1)
   y2 = dlognorm(x, deriv = 2)

## derivative -
   # Compare with Numerical Differentiation:
   par(mfrow = c(2, 2))
   xa = exp(seq(-2.5, 1.5, length = 20))
   plot(x, y0, type = "l", main = "Log-Normal Density")
   plot(x, y1, type = "l", main = "1st Derivative")
   z = derivative(xa, dlognorm(xa, deriv = 0), deriv = 1)
   points(z$x, z$y, col = "steelblue")
   plot(x, y2, type = "l", main = "2nd Derivative")
   z = derivative(xa, dlognorm(xa, deriv = 0), deriv = 2)
   points(z$x, z$y, col = "steelblue")
```

---

## `GammaFunctions` *Gamma and Related Functions*

---

### Description

A collection and description of special mathematical functions. The functions include the error function, the Psi function, the incomplete Gamma function, the Gamma function for complex argument, and the Pochhammer symbol. The Gamma function the logarithm of the Gamma function, their first four derivatives, and the Beta function and the logarithm of the Beta function are part of

R's base package. For example, these functions are required to valuate Asian Options based on the theory of exponential Brownian motion.

The functions are:

| | |
|---|---|
| `erf` | the Error function, |
| `gamma*` | the Gamma function, |
| `lgamma*` | the logarithm of the Gamma function, |
| `digamma*` | the first derivative of the Log Gamma function, |
| `trigamma*` | the second derivative of the Log Gamma function, |
| `tetragamma*` | the third derivative of the Log Gamma function, |
| `pentagamma*` | the fourth derivative of the Log Gammafunction, |
| `beta*` | the Beta function, |
| `lbeta*` | the logarithm of the Beta function, |
| `Psi` | Psi(x) the Psi or Digamma function, |
| `igamma` | P(a,x) the incomplete Gamma function, |
| `cgamma` | Gamma function for complex argument, |
| `Pochhammer` | the Pochhammer symbol. |

The functions marked by an asterisk are part of R's base package.

### Usage

```
erf(x)
Psi(x)
igamma(x, a)
cgamma(x, log = FALSE)
Pochhammer(x, n)
```

### Arguments

| | |
|---|---|
| `x` | [erf] -<br>a real numeric value or vector.<br>[Psi][*gamma][Pochhammer] -<br>a complex numeric value or vector. |
| `a` | a complex numeric value or vector. |
| `n` | an integer value `n >= 0`. A notation used in the theory of special functions for the rising factorial, also known as the rising factorial power, Graham et al. 1994. |
| `log` | a logical, if `TRUE` the logarithm of the complex Gamma function is calculated otherwise if `FALSE`, the complex Gamma function itself will be calculated. |

### Value

The functions return the values of the selected special mathematical function.

### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

### References

Abramowitz M., Stegun I.A. (1972); *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York, Dover Publishing.

Artin, E. (1964); *The Gamma Function*, New York, Holt, Rinehart, and Winston Publishing.

Weisstein E.W. (2004); *MathWorld–A Wolfram Web Resource*, http://mathworld.wolfram.com

### Examples

```
## Calculate Error, Gamma and Related Functions

## gamma -
   # Abramowitz-Stegun: Figure 6.1
   x = seq(-4.01, 4.01, by = 0.011)
   plot(x, gamma(x), ylim = c(-5,5), type = "l", main = "Gamma Function")
   lines(x = c(-4, 4), y = c(0, 0))

## Psi -
   # Abramowitz-Stegun: Figure 6.1
   x = seq(-4.01, 4.01, by = 0.011)
   plot(x, Psi(x), ylim = c(-5, 5), type = "l", main = "Psi Function")
   lines(x = c(-4, 4), y = c(0, 0))
   # Note: Is digamma defined for positive values only ?

## igamma -
   # Abramowitz-Stegun: Figure 6.3.
   gammaStar = function(x, a) { igamma(x,a)/x^a }
   # ... create Figure as an exercise.

## igamma -
   # Abramowitz-Stegun: Formula 6.5.12
   # Relation to Confluent Hypergeometric Functions
   a = sqrt(2)
   x = pi
   Re ( (x^a/a) * kummerM(-x, a, 1+a) )
   Re ( (x^a*exp(-x)/a) * kummerM(x, 1, 1+a) )
   pgamma(x,a) * gamma(a)
   igamma(x, a)

## cgamma -
   # Abramowitz-Stegun: Tables 6.7
   x = 1
   y = seq(0, 5, by = 0.1); x = rep(x, length = length(y))
   z = complex(real = x, imag = y)
   c = cgamma(z, log = TRUE)
   cbind(y, Re(c), Im(c))

## cgamma -
   # Abramowitz-Stegun: Examples 4-8:
   options(digits = 10)
   gamma(6.38); lgamma(56.38)                              # 1/2
   Psi(6.38); Psi(56.38)                                   # 3/4
```

```
cgamma(complex(real = 1, imag = -1), log = TRUE )      # 5
cgamma(complex(real = 1/2, imag = 1/2), log = TRUE )   # 6
cgamma(complex(real = 3, imag = 7), log = TRUE )       # 7/8
```

---

`HypergeometricFunctions` *Confluent Hypergeometric Functions*

---

## Description

A collection and description of special mathematical functions which compute the confluent hypergeometric and related functions. For example, these functions are required to valuate Asian Options based on the theory of exponential Brownian motion.

The functions are:

| | |
|---|---|
| `kummerM` | the Confluent Hypergeometric Function of the 1st Kind, |
| `kummerU` | the Confluent Hypergeometric Function of the 2nd Kind, |
| `whittakerM` | the Whittaker M Function, |
| `whittakerW` | the Whittaker W Function, |
| `hermiteH` | the Hermite Polynomials. |

## Usage

```
kummerM(x, a, b, lnchf = 0, ip = 0)
kummerU(x, a, b, ip = 0)
whittakerM(x, kappa, mu, ip = 0)
whittakerW(x, kappa, mu, ip = 0)
hermiteH(x, n, ip = 0)
```

## Arguments

| | |
|---|---|
| `x` | [kummer*] -<br>a complex numeric value or vector. |
| `a, b` | [kummer*] -<br>complex numeric indexes of the Kummer functions. |
| `ip` | an integer value that specifies how many array positions are desired, usually 10 is sufficient. Setting `ip=0` causes the function to estimate the number of array positions. |
| `kappa, mu` | complex numeric indexes of the Whittaker functions. |
| `lnchf` | an integer value which selects how the result should be represented. A `0` will return the value in standard exponential form, a `1` will return the LOG of the result. |
| `n` | [hermiteH] -<br>the index of the Hermite polynomial, a positive integer value. |

### Details

The functions use the TOMS707 Algorithm by M. Nardin, W.F. Perger and A. Bhalla (1989). A numerical evaluator for the confluent hypergeometric function for complex arguments with large magnitudes using a direct summation of the Kummer series. The method used allows an accuracy of up to thirteen decimal places through the use of large real arrays and a single final division.

The confluent hypergeometric function is the solution to the differential equation:

zf"(z) + (a-z)f'(z) - bf(z) = 0

The Whittaker functions and the Hermite Polynomials are dervived from Kummer's functions.

### Value

The functions return the values of the selected special mathematical function.

### Author(s)

Diethelm Wuertz for the Rmetrics R-port.

### References

Abramowitz M., Stegun I.A. (1972); *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, 9th printing, New York, Dover Publishing.

Weisstein E.W. (2004); *MathWorld – A Wolfram Web Resource*, http://mathworld.wolfram.com

### Examples

```
## kummerM -
   # Abramowitz-Stegun: Formula 13.6.3/13.6.21
   x = c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
   nu = 1; a = nu+1/2; b = 2*nu+1
   M = Re ( kummerM(x = 2*x, a = a, b = b) )
   Bessel = gamma(1+nu) * exp(x)*(x/2)^(-nu) * BesselI(x, nu)
   cbind(x, M, Bessel)

## kummerM -
   # Abramowitz-Stegun: Formula 13.6.14
   x = c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
   M = Re ( kummerM(2*x, a = 1, b = 2) )
   Sinh = exp(x)*sinh(x)/(x)
   cbind(x, M, Sinh)
   # Now the same for complex x:
   y = rep(1, length = length(x))
   x = complex(real = x, imag = y)
   M = kummerM(2*x, a = 1, b = 2)
   Sinh = exp(x)*sinh(x)/(x)
   cbind(x, M, Sinh)

## kummerU -
   # Abramowitz-Stegun: Formula 13.1.3
   x = c(0.001, 0.01, 0.1, 1, 10, 100, 1000)
   a = 1/3; b = 2/3
```

```
   U = Re ( kummerU(x, a = a, b = b) )
   cbind(x, U)

## whittakerM -
   # Abramowitz-Stegun: Example 13
   AS = c(1.10622, 0.57469)
   W = c(
     whittakerM(x = 1, kappa = 0, mu = -0.4),
     whittakerW(x = 1, kappa = 0, mu = -0.4) )
   data.frame(AS, W)

## kummerM
   # Abramowitz-Stegun: Example 17
   x = seq(0, 16, length = 200)
   plot(x = x, y = kummerM(x, -4.5, 1), type = "l", ylim = c(-25,125),
     main = "Figure 13.2:  M(-4.5, 1, x)")
   lines(x = c(0, 16), y = c(0, 0), col = 2)
```

# Index