

# Package ‘AsianOption’

May 6, 2026

**Type** Package

**Title** Asian Option Pricing under Price Impact

**Version** 0.2.0

**Date** 2026-03-10

**Maintainer** Priyanshu Tiwari <tiwari.priyanshu.iitk@gmail.com>

**Description** Implements the framework of Tiwari and Majumdar (2025) <[doi:10.48550/arXiv.2512.07154](https://doi.org/10.48550/arXiv.2512.07154)> for valuing arithmetic and geometric Asian options under transient and permanent market impact. Provides three pricing approaches: Kemna-Vorst frictionless benchmarks, exogenous diffusion pricing (closed-form for geometric, Monte Carlo for arithmetic), and endogenous Hamilton-Jacobi-Bellman valuation via a tree-based Bellman scheme producing indifference bid-ask prices.

**License** GPL (>= 3)

**URL** <https://github.com/plato-12/AsianOption>

**BugReports** <https://github.com/plato-12/AsianOption/issues>

**Encoding** UTF-8

**Depends** R (>= 4.0.0)

**Imports** Rcpp (>= 1.0.0)

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0), covr

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

**Author** Priyanshu Tiwari [aut, cre] (ORCID:  
<<https://orcid.org/0009-0007-8917-4689>>),  
Sourav Majumdar [ctb]

**Repository** CRAN

**Date/Publication** 2026-03-10 09:50:19 UTC

## Contents

arithmetic_asian_bounds . . . . .	2
arithmetic_asian_bounds_transient . . . . .	4
check_no_arbitrage . . . . .	6
compute_adjusted_factors . . . . .	7
compute_p_adj . . . . .	8
price_arithmetic_asian_diffusion . . . . .	8
price_arithmetic_asian_hjb . . . . .	11
price_black_scholes_call . . . . .	13
price_black_scholes_put . . . . .	14
price_european . . . . .	15
price_geometric_asian . . . . .	17
price_geometric_asian_diffusion . . . . .	19
price_geometric_asian_hjb . . . . .	20
price_geometric_asian_transient . . . . .	22
price_kemna_vorst_arithmetic . . . . .	24
price_kemna_vorst_geometric . . . . .	26
print.arithmetic_bounds . . . . .	27
print.hjb_asian . . . . .	28
print.kemna_vorst_arithmetic . . . . .	28
summary.kemna_vorst_arithmetic . . . . .	29

<b>Index</b>	<b>30</b>
--------------	-----------

---

arithmetic\_asian\_bounds

*Bounds for Arithmetic Asian Option with Price Impact*

---

### Description

Computes lower and upper bounds for the arithmetic Asian option (call or put) using the relationship between arithmetic and geometric means (Jensen's inequality).

### Usage

```
arithmetic_asian_bounds(
  S0,
  K,
  r,
  u,
  d,
  lambda,
  v_u,
  v_d,
  n,
  option_type = "call",
  compute_path_specific = FALSE,
```

```

    validate = TRUE
  )

```

### Arguments

<code>S0</code>	Initial stock price (must be positive)
<code>K</code>	Strike price (must be positive)
<code>r</code>	Gross risk-free rate per period (e.g., 1.05)
<code>u</code>	Base up factor in CRR model (must be > d)
<code>d</code>	Base down factor in CRR model (must be positive)
<code>lambda</code>	Price impact coefficient (non-negative)
<code>v_u</code>	Hedging volume on up move (non-negative)
<code>v_d</code>	Hedging volume on down move (non-negative)
<code>n</code>	Number of time steps (positive integer, recommended $n \leq 20$ )
<code>option_type</code>	Character; either "call" (default) or "put"
<code>compute_path_specific</code>	Logical. If TRUE, computes the tighter path-specific upper bound using exact enumeration of all $2^n$ paths. Default is FALSE.
<code>validate</code>	Logical; if TRUE, performs input validation (default TRUE)

### Details

Computes rigorous upper and lower bounds for arithmetic Asian options using Jensen's inequality. The lower bound is the geometric Asian option price (from AM-GM inequality). Two types of upper bounds are available:

**Global upper bound:** Uses a worst-case spread parameter applicable to all paths.

**Path-specific upper bound:** Computes tighter bounds by using path-specific spread parameters. This requires exact enumeration of all  $2^n$  paths in the binomial tree (no sampling or approximation). The path-specific bound is typically much tighter than the global bound.

For detailed mathematical formulations, see the package vignettes and the reference paper.

### Value

List containing:

**lower\_bound** Lower bound for arithmetic option (= geometric option price)

**upper\_bound** Upper bound for arithmetic option (global bound, for backward compatibility)

**upper\_bound\_global** Global upper bound using  $\rho^*$

**upper\_bound\_path\_specific** Path-specific upper bound (only if `compute_path_specific=TRUE`, otherwise NA)

**rho\_star** Spread parameter  $\rho^*$

**EQ\_G** Expected geometric average under risk-neutral measure

**V0\_G** Geometric Asian option price (same as `lower_bound`)

**n\_paths\_used** Number of paths used for path-specific bound ( $2^n$  if computed, 0 otherwise)

**References**

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*.  
[doi:10.48550/arXiv.2512.07154](https://doi.org/10.48550/arXiv.2512.07154)

**See Also**

[price\\_geometric\\_asian](#)

**Examples**

```
# Compute basic bounds (global bound only) for call option
bounds <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 3, option_type = "call"
)

print(bounds)

# Compute bounds for put option
bounds_put <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 3, option_type = "put"
)

# Compute with path-specific bound (uses exact enumeration of all 2^n paths)
bounds_ps <- arithmetic_asian_bounds(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 5,
  compute_path_specific = TRUE
)

print(bounds_ps)

# Estimate arithmetic option price as midpoint of path-specific bounds
if (!is.na(bounds_ps$upper_bound_path_specific)) {
  estimated_price <- mean(c(bounds_ps$lower_bound,
                           bounds_ps$upper_bound_path_specific))
  cat("Estimated price:", estimated_price, "\n")
}
```

---

arithmetic\_asian\_bounds\_transient

*Bounds for Arithmetic Asian Option with Transient Price Impact*

---

**Description**

Computes upper and lower bounds for the arithmetic Asian option price using the transient impact model.

**Usage**

```

arithmetic_asian_bounds_transient(
  S0,
  K,
  r,
  u,
  d,
  lambda_P,
  lambda_T,
  alpha,
  psi,
  volumes,
  option_type = "call",
  compute_path_specific = FALSE,
  validate = TRUE
)

```

**Arguments**

S0	Initial stock price (must be positive)
K	Strike price (must be positive)
r	Gross risk-free rate per period (e.g., 1.05)
u	Base up factor in CRR model (must be > d)
d	Base down factor in CRR model (must be positive)
lambda_P	Permanent price impact coefficient (non-negative)
lambda_T	Transient price impact coefficient (non-negative)
alpha	Decay rate for transient impact (must be in [0, 1])
psi	Power-law exponent for volume (typically in [0.5, 1])
volumes	Vector of hedging volumes at each time step (length n)
option_type	Character; either "call" (default) or "put"
compute_path_specific	Logical. If TRUE, computes tighter path-specific upper bound (requires enumeration of all $2^n$ paths). Default is FALSE.
validate	Logical; if TRUE, performs input validation

**Details**

Computes bounds for arithmetic Asian options under transient impact:

- Lower bound: Geometric Asian option price (AM-GM inequality)
- Upper bound (global): Uses worst-case spread parameter
- Upper bound (path-specific): Uses path-by-path spread (if requested)

The path-specific bound is tighter but requires full path enumeration.

**Value**

List containing:

**lower\_bound** Lower bound (geometric Asian price)  
**upper\_bound\_global** Global upper bound  
**upper\_bound\_path\_specific** Path-specific upper bound (if computed)  
**rho\_star** Global spread parameter  
**EQ\_G** Expected geometric average  
**V0\_G** Geometric Asian price  
**n\_paths\_used** Number of paths enumerated

**See Also**

[price\\_geometric\\_asian\\_transient](#)

**Examples**

```
# Basic bounds with constant volumes
volumes <- rep(1, 10)
bounds <- arithmetic_asian_bounds_transient(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda_P = 0.05, lambda_T = 0.05,
  alpha = 0.5, psi = 1,
  volumes = volumes
)
print(bounds)

# With path-specific bound (tighter but more expensive)
volumes <- rep(1, 8)
bounds_ps <- arithmetic_asian_bounds_transient(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda_P = 0.05, lambda_T = 0.05,
  alpha = 0.5, psi = 1,
  volumes = volumes,
  compute_path_specific = TRUE
)
```

---

check\_no\_arbitrage      *Check No-Arbitrage Condition*

---

**Description**

Verifies that the no-arbitrage condition  $\tilde{d} < r < \tilde{u}$  holds.

**Usage**

```
check_no_arbitrage(r, u, d, lambda, v_u, v_d)
```

**Arguments**

r	Gross risk-free rate per period
u	Base up factor
d	Base down factor
lambda	Price impact coefficient
v_u	Hedging volume on up move
v_d	Hedging volume on down move

**Value**

Logical: TRUE if condition holds, FALSE otherwise

**Examples**

```
check_no_arbitrage(r = 1.05, u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

---

```
compute_adjusted_factors
```

*Compute Adjusted Up and Down Factors*

---

**Description**

Calculates the modified up and down factors after incorporating price impact from hedging.

**Usage**

```
compute_adjusted_factors(u, d, lambda, v_u, v_d)
```

**Arguments**

u	Base up factor
d	Base down factor
lambda	Price impact coefficient
v_u	Hedging volume on up move
v_d	Hedging volume on down move

**Value**

List with elements `u_tilde` and `d_tilde`

**Examples**

```
compute_adjusted_factors(u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

---

compute\_p\_adj                      *Compute Adjusted Risk-Neutral Probability*

---

### Description

Calculates the adjusted risk-neutral probability incorporating price impact from hedging activities.

### Usage

```
compute_p_adj(r, u, d, lambda, v_u, v_d)
```

### Arguments

r	Gross risk-free rate per period
u	Base up factor
d	Base down factor
lambda	Price impact coefficient
v_u	Hedging volume on up move
v_d	Hedging volume on down move

### Value

Adjusted risk-neutral probability (numeric)

### Examples

```
compute_p_adj(r = 1.05, u = 1.2, d = 0.8, lambda = 0.1, v_u = 1, v_d = 1)
```

---

price\_arithmetic\_asian\_diffusion  
*Arithmetic Asian Option Price via Euler-Maruyama Monte Carlo (Exogenous Diffusion)*

---

### Description

Prices an arithmetic Asian option under exogenous transient price impact using Euler-Maruyama Monte Carlo simulation.

**Usage**

```
price_arithmetic_asian_diffusion(
    S0,
    K,
    r,
    sigma,
    T,
    lambda_T,
    I0,
    kappa,
    eta,
    rho = 0,
    option_type = "call",
    n_steps = 252,
    n_sims = 1e+05,
    use_control_variate = TRUE,
    seed = 0,
    n_quad = 1000
)
```

**Arguments**

<code>S0</code>	Initial stock price (positive).
<code>K</code>	Strike price (positive).
<code>r</code>	Risk-free rate (positive).
<code>sigma</code>	Volatility parameter (positive).
<code>T</code>	Time to maturity (positive).
<code>lambda_T</code>	Transient impact coefficient (non-negative).
<code>I0</code>	Initial transient impact state (real number).
<code>kappa</code>	Mean reversion rate for transient impact (positive).
<code>eta</code>	Noise amplitude for transient impact process. Can be: - A single non-negative number (constant eta) - A function of time $t$ in $[0,T]$ returning a non-negative value
<code>rho</code>	Correlation between stock and impact Brownian motions (in $[-1,1]$ ). Default is 0.
<code>option_type</code>	Character string: "call" (default) or "put".
<code>n_steps</code>	Number of time steps in the Euler-Maruyama discretisation (default: 252).
<code>n_sims</code>	Number of Monte Carlo simulation paths (default: 100000).
<code>use_control_variate</code>	Logical. If TRUE (default), uses the geometric Asian diffusion closed-form price as a control variate to reduce variance.
<code>seed</code>	Integer seed for reproducibility. 0 means no seed (default: 0).
<code>n_quad</code>	Number of quadrature points for the geometric closed-form computation when using control variates (default: 1000).

**Details**

In the exogenous regime with no active trading ( $\nu \equiv 0$ ), the stock price dynamics are:

$$\begin{aligned} dS_t &= S_t(r + \bar{\lambda}_T I_t) dt + \sigma S_t dW_t \\ dI_t &= -\kappa I_t dt + \eta(t) dW_t^I \end{aligned}$$

where  $W$  and  $W^I$  are Brownian motions with instantaneous correlation  $\rho$ .

The arithmetic Asian payoff is  $\Phi_A(Y_T) = (Y_T/T - K)^+$  where  $Y_t = \int_0^t S_u du$ .

The Euler-Maruyama scheme discretises the SDEs on a uniform grid with step  $\Delta t = T/N$ . The log-Euler method is used for  $S$  to ensure positivity.

When `use_control_variate = TRUE`, the geometric Asian diffusion closed-form from [price\\_geometric\\_asian\\_diffusion](#) is used as a control variate, which typically reduces the standard error substantially.

**Value**

An object of class "arithmetic\_asian\_diffusion" (a list) with:

**price** Estimated option price.

**std\_error** Standard error of the estimate.

**lower\_ci** Lower bound of 95% confidence interval.

**upper\_ci** Upper bound of 95% confidence interval.

**geometric\_price** Closed-form geometric Asian price (benchmark).

**correlation** Correlation between arithmetic and geometric MC payoffs.

**use\_control\_variate** Whether control variate was used.

**n\_sims** Number of simulations.

**n\_steps** Number of time steps.

**References**

Section 3.2.1 of "Asian option valuation under price impact"

**See Also**

[price\\_geometric\\_asian\\_diffusion](#) for the geometric Asian closed-form in the same diffusion limit.

**Examples**

```
# Basic call pricing
price_arithmetic_asian_diffusion(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2, T = 1,
  lambda_T = 0.01, I0 = 0, kappa = 1, eta = 0.1, rho = 0,
  n_steps = 100, n_sims = 10000, seed = 42
)

# With time-dependent eta
```

```

eta_func <- function(t) 0.1 * (1 + 0.5 * t)
price_arithmetic_asian_diffusion(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2, T = 1,
  lambda_T = 0.01, I0 = 0.5, kappa = 2, eta = eta_func, rho = 0.3,
  n_steps = 100, n_sims = 10000, seed = 42
)

```

---

```
price_arithmetic_asian_hjb
```

*Price Arithmetic Asian Option via HJB Bellman Scheme (Endogenous Impact)*

---

### Description

Computes bid and ask prices for an arithmetic Asian option under transient price impact using a Bellman (HJB) scheme.

### Usage

```

price_arithmetic_asian_hjb(
  S0,
  K,
  T,
  N,
  sigma,
  r_cont,
  kappa,
  lambda_bar_T,
  lambda_bar_P,
  k_A,
  k_B,
  psi_cost,
  eta = 1,
  p = 0.5,
  I0 = 0,
  control_set = NULL,
  nu_min = -5,
  nu_max = 5,
  n_controls = 31,
  n_logS = NULL,
  n_I = 51,
  n_Y = 51,
  option_type = "call",
  validate = TRUE
)

```

**Arguments**

$S_0$	Initial stock price (positive).
$K$	Strike price (positive).
$T$	Time to maturity (positive).
$N$	Number of time steps (positive integer).
$\sigma$	Volatility (positive).
$r_{\text{cont}}$	Continuous risk-free rate.
$\kappa$	Mean reversion rate for impact (non-negative).
$\lambda_{\text{bar}_T}$	Transient impact coefficient (non-negative).
$\lambda_{\text{bar}_P}$	Permanent impact coefficient (non-negative).
$k_A, k_B$	Cost coefficients (non-negative).
$\psi_{\text{cost}}$	Cost exponent (in $(0, 2]$ ).
$\eta$	Noise trader intensity (scalar or length- $N$ vector).
$p$	Probability of up move (in $(0, 1)$ ).
$I_0$	Initial impact state.
$\text{control\_set}$	Optional numeric vector of controls; otherwise built from $\text{nu\_min}$ , $\text{nu\_max}$ , $\text{n\_controls}$ .
$\text{nu\_min}, \text{nu\_max}, \text{n\_controls}$	Control grid (used if $\text{control\_set}$ is NULL).
$\text{n\_logS}, \text{n\_I}$	Grid sizes for log-price and impact state.
$\text{n\_Y}$	Grid size for running state (integral of log $S$ ). Ignored if $\text{n\_Z}$ is provided.
$\text{option\_type}$	"call" or "put".
$\text{validate}$	Whether to validate inputs.

**Details**

Bid and ask are defined via value-function differences: a baseline problem (no option), a long-option problem, and a short-option problem are solved internally in C++.

**Value**

A list with S3 class "hjb\_asian" containing:

**ask\_price** Ask (seller's indifference) price at  $t=0$

**bid\_price** Bid (buyer's indifference) price at  $t=0$

**mid\_price** Mid price

**spread** Ask minus bid

**optimal\_nu** Optimal trading rate (seller/short) per period, length  $N$

**optimal\_volumes** Seller volumes per period ( $\text{optimal\_nu} * \text{dt}$ )

**optimal\_nu\_buyer** Optimal trading rate (buyer/long) per period

**optimal\_volumes\_buyer** Buyer volumes per period

**asian\_type** Type of Asian option ("arithmetic")  
**option\_type** Option type  
**params** List of input parameters  
**grid\_sizes** Grid sizes used in the computation

price\_black\_scholes\_call

*Black-Scholes European Call Option Price***Description**

Computes the exact price of a European call option using the classical Black-Scholes (1973) analytical formula. This is the continuous-time benchmark for comparison with discrete binomial models.

**Usage**

```
price_black_scholes_call(S0, K, r, sigma, time_to_maturity)
```

**Arguments**

S0	Initial stock price (must be positive)
K	Strike price (must be positive)
r	Continuously compounded risk-free rate (e.g., 0.05 for 5% annual rate)
sigma	Volatility (annualized standard deviation, must be non-negative)
time_to_maturity	Time to maturity in years (must be positive)

**Details**

The Black-Scholes formula for a European call option is:

$$C = S_0 N(d_1) - K e^{-rT} N(d_2)$$

where:

$$d_1 = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

and  $N(\cdot)$  is the cumulative standard normal distribution function.

This formula assumes:

- Stock price follows geometric Brownian motion:  $dS_t = rS_t dt + \sigma S_t dW_t$
- No dividends
- Constant risk-free rate and volatility
- Continuous trading with no transaction costs or price impact

**Value**

European call option price (numeric)

**References**

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637-654. doi:10.1086/260062

**Examples**

```
price_black_scholes_call(S0 = 100, K = 100, r = 0.05, sigma = 0.2,
                        time_to_maturity = 1)
```

---

```
price_black_scholes_put
```

*Black-Scholes European Put Option Price*

---

**Description**

Computes the exact price of a European put option using the classical Black-Scholes (1973) analytical formula.

**Usage**

```
price_black_scholes_put(S0, K, r, sigma, time_to_maturity)
```

**Arguments**

S0	Initial stock price (must be positive)
K	Strike price (must be positive)
r	Continuously compounded risk-free rate (e.g., 0.05 for 5% annual rate)
sigma	Volatility (annualized standard deviation, must be non-negative)
time_to_maturity	Time to maturity in years (must be positive)

**Details**

The Black-Scholes formula for a European put option is:

$$P = Ke^{-rT}N(-d_2) - S_0N(-d_1)$$

where:

$$d_1 = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

and  $N(\cdot)$  is the cumulative standard normal distribution function.

Alternatively, the put price can be derived from put-call parity:

$$P = C - S_0 + Ke^{-rT}$$

### Value

European put option price (numeric)

### Put-Call Parity

The Black-Scholes put and call prices satisfy:

$$C - P = S_0 - Ke^{-rT}$$

This relationship holds exactly for European options without dividends.

### References

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), 637-654. doi:10.1086/260062

### See Also

[price\\_black\\_scholes\\_call](#)

### Examples

```
price_black_scholes_put(S0 = 100, K = 100, r = 0.05, sigma = 0.2,
  time_to_maturity = 1)
```

---

price\_european

*Price European Option with Price Impact*

---

### Description

Computes the exact price of a European option (call or put) using the Cox-Ross-Rubinstein (CRR) binomial model with price impact from hedging activities.

### Usage

```
price_european(
  S0,
  K,
  r,
  u,
  d,
```

```

    lambda,
    v_u,
    v_d,
    n,
    option_type = "call",
    validate = TRUE
)

```

### Arguments

$S_0$	Initial stock price (must be positive)
$K$	Strike price (must be positive)
$r$	Gross risk-free rate per period (e.g., 1.05 for 5% rate)
$u$	Base up factor in CRR model (must be $> d$ )
$d$	Base down factor in CRR model (must be positive)
$\lambda$	Price impact coefficient (non-negative)
$v_u$	Hedging volume on up move (non-negative)
$v_d$	Hedging volume on down move (non-negative)
$n$	Number of time steps (positive integer)
option_type	Character; either "call" (default) or "put"
validate	Logical; if TRUE, performs input validation

### Details

Computes exact prices for European options (call or put) using the binomial model with price impact. Price impact from hedging activities modifies the stock dynamics through adjusted up/down factors and risk-neutral probability.

Unlike path-dependent Asian options, European options only depend on the terminal stock price, allowing for efficient  $O(n)$  computation instead of  $O(2^n)$ . See the package vignettes and reference paper for detailed mathematical formulations.

### Value

European option price (numeric)

### References

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*. doi:10.48550/arXiv.2512.07154

### See Also

[price\\_geometric\\_asian](#), [compute\\_p\\_adj](#)

**Examples**

```

# Call option with no price impact
price_european(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0, v_u = 0, v_d = 0, n = 10, option_type = "call"
)

# Put option with price impact
price_european(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 10, option_type = "put"
)

# Verify put-call parity
call <- price_european(100, 100, 1.05, 1.2, 0.8, 0.1, 1, 1, 10, "call")
put <- price_european(100, 100, 1.05, 1.2, 0.8, 0.1, 1, 1, 10, "put")

```

---

price\_geometric\_asian *Price Geometric Asian Option with Price Impact*

---

**Description**

Computes the exact price of a geometric Asian option (call or put) using the Cox-Ross-Rubinstein (CRR) binomial model with price impact from hedging activities. Uses exact enumeration of all  $2^n$  paths.

**Usage**

```

price_geometric_asian(
  S0,
  K,
  r,
  u,
  d,
  lambda,
  v_u,
  v_d,
  n,
  option_type = "call",
  validate = TRUE
)

```

**Arguments**

S0	Initial stock price (must be positive)
K	Strike price (must be positive)

r	Gross risk-free rate per period (e.g., 1.05)
u	Base up factor in CRR model (must be > d)
d	Base down factor in CRR model (must be positive)
lambda	Price impact coefficient (non-negative)
v_u	Hedging volume on up move (non-negative)
v_d	Hedging volume on down move (non-negative)
n	Number of time steps (positive integer)
option_type	Character; either "call" (default) or "put"
validate	Logical; if TRUE, performs input validation

### Details

Computes exact prices for geometric Asian options using complete path enumeration in a binomial tree. Price impact from hedging activities modifies the stock dynamics through adjusted up/down factors and risk-neutral probability.

This function enumerates all  $2^n$  possible paths in the binomial tree for exact pricing (no approximation or sampling). For large  $n$  ( $> 20$ ), this requires significant computation time and memory. See the package vignettes and reference paper for detailed mathematical formulations.

### Value

Geometric Asian option price (numeric).

### References

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*. [doi:10.48550/arXiv.2512.07154](https://doi.org/10.48550/arXiv.2512.07154)

### See Also

[arithmetic\\_asian\\_bounds](#), [compute\\_p\\_adj](#)

### Examples

```
# Basic example
price_geometric_asian(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0, v_u = 0, v_d = 0, n = 10
)

# With price impact
price_geometric_asian(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda = 0.1, v_u = 1, v_d = 1, n = 15
)

# Put option
price_geometric_asian(
```

```

S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
lambda = 0.1, v_u = 1, v_d = 1, n = 10,
option_type = "put"
)

```

---

price\_geometric\_asian\_diffusion

*Geometric Asian Option Price in Exogenous Diffusion Limit*

---

### Description

Computes the closed-form price for a geometric Asian call option in the exogenous diffusion limit with transient price impact.

### Usage

```

price_geometric_asian_diffusion(
  S0,
  K,
  r,
  sigma,
  T,
  lambda_T,
  I0,
  kappa,
  eta,
  rho = 0,
  option_type = "call",
  n_quad = 1000
)

```

### Arguments

S0	Initial stock price (positive).
K	Strike price (positive).
r	Risk-free rate (positive, typically close to 1 in discrete models).
sigma	Volatility parameter (positive).
T	Time to maturity (positive).
lambda_T	Transient impact coefficient (non-negative).
I0	Initial transient impact state (can be any real number).
kappa	Mean reversion rate for transient impact (positive).
eta	Noise amplitude for transient impact process. Can be: - A single positive number (constant eta) - A function of time t in [0,T] returning a non-negative value
rho	Correlation between stock and impact Brownian motions (in [-1,1]).
option_type	Character string: "call" (default) or "put".
n_quad	Number of quadrature points for numerical integration (default: 1000).

**Details**

In the exogenous regime with no trading control ( $\nu = 0$ ), the stock price dynamics are:

$$dS_t = S_t * (r + \lambda_T * I_t) dt + \sigma * S_t * dW_t \quad dI_t = -\kappa * I_t dt + \eta(t) * dW^I_t$$

where  $W$  and  $W^I$  are Brownian motions with correlation  $\rho$ .

The running log-integral  $Z_T = \int_0^T \log(S_u) du$  is Gaussian, so  $G_T = \exp(Z_T/T)$  is lognormal. This yields a Black-Scholes type closed form:

$$U(0) = \exp(-r*T) * [\exp(\mu_G + \sigma_G^2/2) * \Phi(d1) - K * \Phi(d2)]$$

where:  $-\mu_G = m_Z / T - \sigma_G^2 = v_Z / T^2 - m_Z$  and  $v_Z$  are the mean and variance of  $Z_T$  -  $\Phi$  is the standard normal CDF -  $d1 = (\mu_G - \log(K) + \sigma_G^2) / \sigma_G$  -  $d2 = d1 - \sigma_G$

**Value**

The price of the geometric Asian option in the exogenous diffusion limit.

**References**

Section 3.2.2 of "Asian option valuation under price impact"

**Examples**

```
# Example 1: Constant eta, no correlation
price_geometric_asian_diffusion(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2, T = 1,
  lambda_T = 0.01, I0 = 0, kappa = 1, eta = 0.1, rho = 0
)

# Example 2: Time-dependent eta
eta_func <- function(t) 0.1 * (1 + 0.5 * t)
price_geometric_asian_diffusion(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2, T = 1,
  lambda_T = 0.01, I0 = 0.5, kappa = 2, eta = eta_func, rho = 0.3
)
```

---

```
price_geometric_asian_hjb
```

*Price Geometric Asian Option via HJB Bellman Scheme (Endogenous Impact)*

---

**Description**

Computes bid and ask prices for a geometric Asian option under transient price impact using a Bellman (HJB) scheme. Same interface as [price\\_arithmetic\\_asian\\_hjb](#); the running average is geometric (average of log-price, then exp).

**Usage**

```
price_geometric_asian_hjb(
  S0,
  K,
  T,
  N,
  sigma,
  r_cont,
  kappa,
  lambda_bar_T,
  lambda_bar_P,
  k_A,
  k_B,
  psi_cost,
  eta = 1,
  p = 0.5,
  I0 = 0,
  control_set = NULL,
  nu_min = -5,
  nu_max = 5,
  n_controls = 31,
  n_logS = NULL,
  n_I = 51,
  n_Y = 51,
  n_Z = NULL,
  option_type = "call",
  validate = TRUE
)
```

**Arguments**

S0	Initial stock price (positive).
K	Strike price (positive).
T	Time to maturity (positive).
N	Number of time steps (positive integer).
sigma	Volatility (positive).
r_cont	Continuous risk-free rate.
kappa	Mean reversion rate for impact (non-negative).
lambda_bar_T	Transient impact coefficient (non-negative).
lambda_bar_P	Permanent impact coefficient (non-negative).
k_A, k_B	Cost coefficients (non-negative).
psi_cost	Cost exponent (in (0, 2]).
eta	Noise trader intensity (scalar or length-N vector).
p	Probability of up move (in (0, 1)).

I0	Initial impact state.
control_set	Optional numeric vector of controls; otherwise built from nu_min, nu_max, n_controls.
nu_min, nu_max, n_controls	Control grid (used if control_set is NULL).
n_logS, n_I	Grid sizes for log-price and impact state.
n_Y	Grid size for running state (integral of log S). Ignored if n_Z is provided.
n_Z	Grid size for running state (alias for geometric case). If provided, used instead of n_Y.
option_type	"call" or "put".
validate	Whether to validate inputs.

**Value**

List with S3 class "hjb\_asian" (same structure as arithmetic), with asian\_type = "geometric".

---

```
price_geometric_asian_transient
```

*Price Geometric Asian Option with Transient and Permanent Price Impact*

---

**Description**

Computes the price of a geometric Asian option using the binomial model with both transient and permanent price impact from hedging activities.

**Usage**

```
price_geometric_asian_transient(
  S0,
  K,
  r,
  u,
  d,
  lambda_P,
  lambda_T,
  alpha,
  psi,
  volumes,
  option_type = "call",
  validate = TRUE
)
```

**Arguments**

<code>S0</code>	Initial stock price (must be positive)
<code>K</code>	Strike price (must be positive)
<code>r</code>	Gross risk-free rate per period (e.g., 1.05)
<code>u</code>	Base up factor in CRR model (must be > d)
<code>d</code>	Base down factor in CRR model (must be positive)
<code>lambda_P</code>	Permanent price impact coefficient (non-negative)
<code>lambda_T</code>	Transient price impact coefficient (non-negative)
<code>alpha</code>	Decay rate for transient impact (must be in [0, 1])
<code>psi</code>	Power-law exponent for volume (typically in [0.5, 1])
<code>volumes</code>	Vector of hedging volumes at each time step (length n)
<code>option_type</code>	Character; either "call" (default) or "put"
<code>validate</code>	Logical; if TRUE, performs input validation

**Details**

This function implements the transient impact model where price impact has both permanent and transient components. The transient component decays exponentially with rate alpha.

The stock price dynamics are:

$$S_{m+1} = u \cdot S_m \cdot \exp(\lambda_P v_m^\psi + \lambda_T \sum_{k=0}^m \alpha^{m-k} \epsilon_k v_k^\psi)$$

for an up move, and similarly for a down move.

Key parameters:

- `lambda_P`: Permanent impact (persistent price change)
- `lambda_T`: Transient impact (temporary price change)
- `alpha`: Decay rate (alpha = 0 means no memory, alpha close to 1 means long memory)
- `psi`: Volume power-law (psi = 1 is linear, psi = 0.5 is square-root)

Unlike the permanent impact model, the risk-neutral probability remains path-dependent even for constant volumes, due to the transient accumulator term in the numerator of the probability formula.

**Value**

Numeric value of the option price.

**References**

Tiwari, P., & Majumdar, S. (2025). Asian option valuation under price impact. *arXiv preprint*.  
doi:10.48550/arXiv.2512.07154

**See Also**

[arithmetic\\_asian\\_bounds\\_transient](#), [price\\_geometric\\_asian](#)

**Examples**

```
# Example 1: Constant volumes with transient impact
volumes <- rep(1, 10)
price <- price_geometric_asian_transient(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda_P = 0.05, lambda_T = 0.05,
  alpha = 0.5, psi = 1,
  volumes = volumes
)

# Example 2: Time-varying volumes
volumes <- seq(0.5, 1.5, length.out = 10)
price <- price_geometric_asian_transient(
  S0 = 100, K = 100, r = 1.05, u = 1.2, d = 0.8,
  lambda_P = 0.08, lambda_T = 0.02,
  alpha = 0.3, psi = 0.5,
  volumes = volumes
)
```

---

```
price_kemna_vorst_arithmetic
```

*Kemna-Vorst Arithmetic Average Asian Option*

---

**Description**

Calculates the price of an arithmetic average Asian option using Monte Carlo simulation with variance reduction via the geometric average control variate. This implements the Kemna & Vorst (1990) method WITHOUT price impact.

**Usage**

```
price_kemna_vorst_arithmetic(
  S0,
  K,
  r,
  sigma,
  T0,
  T_mat,
  n,
  M = 10000,
  option_type = "call",
  use_control_variate = TRUE,
  seed = NULL,
  return_diagnostics = FALSE
)
```

**Arguments**

<code>S0</code>	Numeric. Initial stock price at time $T_0$ (start of averaging period). Must be positive.
<code>K</code>	Numeric. Strike price. Must be positive.
<code>r</code>	Numeric. Continuously compounded risk-free rate (e.g., 0.05 for 5%). Use <code>log(r_gross)</code> to convert from gross rate.
<code>sigma</code>	Numeric. Volatility (annualized standard deviation). Must be non-negative.
<code>T0</code>	Numeric. Start time of averaging period. Must be non-negative.
<code>T_mat</code>	Numeric. Maturity time. Must be greater than $T_0$ .
<code>n</code>	Integer. Number of averaging points (observations). Must be positive.
<code>M</code>	Integer. Number of Monte Carlo simulations. Default is 10000. Larger values give more accurate results but take longer.
<code>option_type</code>	Character. Type of option: "call" (default) or "put".
<code>use_control_variate</code>	Logical. If TRUE (default), uses the geometric average as a control variate for variance reduction. This dramatically improves accuracy.
<code>seed</code>	Integer. Random seed for reproducibility. Default is NULL (no seed).
<code>return_diagnostics</code>	Logical. If TRUE, returns additional diagnostic information including confidence intervals, correlation, and variance reduction factor. Default is FALSE.

**Value**

If `return_diagnostics = FALSE`, returns a numeric value (the estimated option price). If `return_diagnostics = TRUE`, returns a list with components:

**price** Estimated option price  
**std\_error** Standard error of the estimate  
**lower\_ci** Lower 95% confidence interval  
**upper\_ci** Upper 95% confidence interval  
**geometric\_price** Analytical geometric average price (control variate)  
**correlation** Correlation between arithmetic and geometric payoffs  
**variance\_reduction\_factor** Ratio of variances (with/without control)  
**n\_simulations** Number of Monte Carlo simulations used  
**n\_steps** Number of time steps in each simulation

**References**

Kemna, A.G.Z. and Vorst, A.C.F. (1990). "A Pricing Method for Options Based on Average Asset Values." *Journal of Banking and Finance*, 14, 113-129.

**Examples**

```
price_kemna_vorst_arithmetic(
  S0 = 100, K = 100, r = 0.05, sigma = 0.2,
  T0 = 0, T_mat = 1, n = 50, M = 10000
)
```

---

```
price_kemna_vorst_geometric
```

*Kemna-Vorst Geometric Average Asian Option*

---

**Description**

Calculates the price of a geometric average Asian call option using the closed-form analytical solution from Kemna & Vorst (1990). This is the standard benchmark implementation WITHOUT price impact.

**Usage**

```
price_kemna_vorst_geometric(S0, K, r, sigma, T0, T_mat, option_type = "call")
```

**Arguments**

S0	Numeric. Initial stock price at time T0 (start of averaging period). Must be positive.
K	Numeric. Strike price. Must be positive.
r	Numeric. Gross risk-free interest rate per period (e.g., 1.05 for 5 Must be positive).
sigma	Numeric. Volatility (annualized standard deviation). Must be non-negative.
T0	Numeric. Start time of averaging period. Must be non-negative.
T_mat	Numeric. Maturity time. Must be greater than T0.
option_type	Character. Type of option: "call" (default) or "put".

**Details**

The geometric average at maturity is defined as:

$$G_T = \exp\left(\frac{1}{T - T_0} \int_{T_0}^T \log(S(\tau)) d\tau\right)$$

For the discrete case with n+1 observations:

$$G_T = \left(\prod_{i=0}^n S(T_i)\right)^{1/(n+1)}$$

The closed-form solution for a call option is:

$$C = S_0 e^{d^*} N(d) - KN(d - \sigma_G \sqrt{T - T_0})$$

where:

$$d^* = \frac{1}{2} \left( r - \frac{\sigma^2}{6} \right) (T - T_0)$$

$$d = \frac{\log(S_0/K) + \frac{1}{2} \left( r + \frac{\sigma^2}{6} \right) (T - T_0)}{\sigma \sqrt{(T - T_0)/3}}$$

and  $N(\cdot)$  is the cumulative standard normal distribution function.

### Value

Numeric. The analytical price of the geometric average Asian option.

### References

Kemna, A.G.Z. and Vorst, A.C.F. (1990). "A Pricing Method for Options Based on Average Asset Values." *Journal of Banking and Finance*, 14, 113-129.

### Examples

```
price_kemna_vorst_geometric(
    S0 = 100, K = 100, r = 0.05, sigma = 0.2,
    T0 = 0, T_mat = 1, option_type = "call"
)
```

---

```
print.arithmetic_bounds
```

*Print Method for Arithmetic Asian Bounds*

---

### Description

Print Method for Arithmetic Asian Bounds

### Usage

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

### Arguments

```
x          Object of class arithmetic_bounds
...        Additional arguments (unused)
```

### Value

Invisible x

---

`print.hjb_asian`      *Print method for HJB Asian option results*

---

**Description**

Print method for HJB Asian option results

**Usage**

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

**Arguments**

`x`                    Object of class `hjb_asian` from `price_arithmetic_asian_hjb`.  
`...`                 Additional arguments (unused).

**Value**

Invisible `x`.

---

`print.kemna_vorst_arithmetic`  
*Print Method for Kemna-Vorst Arithmetic Results*

---

**Description**

Print Method for Kemna-Vorst Arithmetic Results

**Usage**

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

**Arguments**

`x`                    Object of class "kemna\_vorst\_arithmetic"  
`...`                 Additional arguments (ignored)

**Value**

Invisibly returns the input object `x`. Called for side effects (printing).

---

`summary.kemna_vorst_arithmetic`*Summary Method for Kemna-Vorst Arithmetic Results*

---

**Description**

Summary Method for Kemna-Vorst Arithmetic Results

**Usage**

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

**Arguments**

<code>object</code>	Object of class "kemna_vorst_arithmetic"
<code>...</code>	Additional arguments (ignored)

**Value**

Invisibly returns the input object `object`. Called for side effects (printing).

# Index

arithmetic\_asian\_bounds, [2](#), [18](#)  
arithmetic\_asian\_bounds\_transient, [4](#),  
[23](#)

check\_no\_arbitrage, [6](#)  
compute\_adjusted\_factors, [7](#)  
compute\_p\_adj, [8](#), [16](#), [18](#)

price\_arithmetic\_asian\_diffusion, [8](#)  
price\_arithmetic\_asian\_hjb, [11](#), [20](#)  
price\_black\_scholes\_call, [13](#), [15](#)  
price\_black\_scholes\_put, [14](#)  
price\_european, [15](#)  
price\_geometric\_asian, [4](#), [16](#), [17](#), [23](#)  
price\_geometric\_asian\_diffusion, [10](#), [19](#)  
price\_geometric\_asian\_hjb, [20](#)  
price\_geometric\_asian\_transient, [6](#), [22](#)  
price\_kemna\_vorst\_arithmetic, [24](#)  
price\_kemna\_vorst\_geometric, [26](#)  
print.arithmetic\_bounds, [27](#)  
print.hjb\_asian, [28](#)  
print.kemna\_vorst\_arithmetic, [28](#)

summary.kemna\_vorst\_arithmetic, [29](#)