NAG Library Function Document

nag_asian_geom_price (s30sac)

1 Purpose

nag_asian_geom_price (s30sac) computes the Asian geometric continuous average-rate option price.

2 Specification

```c
#include <nag.h>
#include <nags.h>

void nag_asian_geom_price (Nag_OrderType order, Nag_CallPut option,
                         Integer m, Integer n, const double x[], double s, const double t[],
                         double sigma, double r, double b, double p[], NagError *fail)
```

3 Description

nag_asian_geom_price (s30sac) computes the price of an Asian geometric continuous average-rate option for constant volatility, $\sigma$, risk-free rate, $r$, and cost of carry, $b$ (see Kemna and Vorst (1990)). For a given strike price, $X$, the price of a call option with underlying price, $S$, and time to expiry, $T$, is

$$P_{\text{call}} = S e^{(b-r)T} \Phi(d_1) - X e^{-rT} \Phi(d_2),$$

and the corresponding put option price is

$$P_{\text{put}} = X e^{-rT} \Phi(-d_2) - S e^{(b-r)T} \Phi(-d_1),$$

where

$$d_1 = \frac{\ln(S/X) + (b + \sigma^2/2)T}{\sigma \sqrt{T}}$$

and

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

with

$$\bar{\sigma} = \frac{\sigma}{\sqrt{3}}, \quad \bar{b} = \frac{1}{2} \left( r - \frac{\sigma^2}{3} \right).$$

$\Phi$ is the cumulative Normal distribution function,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp(-y^2/2) dy.$$

The option price $P_{ij} = P(X = X_i, T = T_j)$ is computed for each strike price in a set $X_i$, $i = 1, 2, \ldots, m$, and for each expiry time in a set $T_j$, $j = 1, 2, \ldots, n$.

4 References

5 Arguments

1: order – Nag_OrderType Input

On entry: the order argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by order = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: option – Nag_CallPut Input

On entry: determines whether the option is a call or a put.

option = Nag_Call
   A call; the holder has a right to buy.

option = Nag_Put
   A put; the holder has a right to sell.

Constraint: option = Nag_Call or Nag_Put.

3: m – Integer Input

On entry: the number of strike prices to be used.

Constraint: m ≥ 1.

4: n – Integer Input

On entry: the number of times to expiry to be used.

Constraint: n ≥ 1.

5: x[m] – const double Input

On entry: x[i - 1] must contain X_i, the i-th strike price, for i = 1, 2, . . . , m.

Constraint: x[i - 1] ≥ z and x[i - 1] ≤ 1/z, where z = nag_real_safe_small_number, the safe range parameter, for i = 1, 2, . . . , m.

6: s – double Input

On entry: S, the price of the underlying asset.

Constraint: s ≥ z and s ≤ 1.0/z, where z = nag_real_safe_small_number, the safe range parameter.

7: t[n] – const double Input

On entry: t[i - 1] must contain T_i, the i-th time, in years, to expiry, for i = 1, 2, . . . , n.

Constraint: t[i - 1] ≥ z, where z = nag_real_safe_small_number, the safe range parameter, for i = 1, 2, . . . , n.

8: sigma – double Input

On entry: \sigma, the volatility of the underlying asset. Note that a rate of 15% should be entered as 0.15.

Constraint: sigma > 0.0.

9: r – double Input

On entry: r, the annual risk-free interest rate, continuously compounded. Note that a rate of 5% should be entered as 0.05.

Constraint: r ≥ 0.0.
10: \( b \) – double

*Input*

On entry: \( b \), the annual cost of carry rate. Note that a rate of 8% should be entered as 0.08.

11: \( p[m \times n] \) – double

*Output*

Note: where \( P(i,j) \) appears in this document, it refers to the array element

\[
p[(j-1) \times m + i - 1] \text{ when order = Nag_ColMajor;}
p[(i-1) \times n + j - 1] \text{ when order = Nag_RowMajor.}
\]

On exit: \( P(i,j) \) contains \( P_{ij} \), the option price evaluated for the strike price \( x_i \) at expiry \( t_j \) for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

12: \( \text{fail} \) – NagError *

*Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

**NE_INT**

On entry, \( m = \langle \text{value} \rangle \).

Constraint: \( m \geq 1 \).

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 1 \).

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 3.6.6 in the Essential Introduction for further information.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL**

On entry, \( r = \langle \text{value} \rangle \).

Constraint: \( r \geq 0.0 \).

On entry, \( s = \langle \text{value} \rangle \).

Constraint: \( s \geq \langle \text{value} \rangle \) and \( s \leq \langle \text{value} \rangle \).

On entry, \( \text{sigma} = \langle \text{value} \rangle \).

Constraint: \( \text{sigma} > 0.0 \).

**NE_REAL_ARRAY**

On entry, \( t[i] = \langle \text{value} \rangle \).

Constraint: \( t[i] \geq \langle \text{value} \rangle \).
On entry, \( x[\text{value}] = \langle \text{value} \rangle \).
Constraint: \( x[i] \geq \langle \text{value} \rangle \) and \( x[i] \leq \langle \text{value} \rangle \).

7 Accuracy

The accuracy of the output is dependent on the accuracy of the cumulative Normal distribution function, \( \Phi \). This is evaluated using a rational Chebyshev expansion, chosen so that the maximum relative error in the expansion is of the order of the machine precision (see nag_cumul_normal (s15abc) and nag_erfc (s15adc)). An accuracy close to machine precision can generally be expected.

8 Parallelism and Performance

nag_asian_geom_price (s30sac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users’ Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example computes the price of an Asian geometric continuous average-rate put with a time to expiry of 3 months, a stock price of 80 and a strike price of 85. The risk-free interest rate is 5% per year, the cost of carry is 8% and the volatility is 20% per year.

10.1 Program Text

/* nag_asian_geom_price (s30sac) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 9, 2009. */
*/
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <nag.h>
#include <nags.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, j, m, n;
    NagError fail;
    Nag_CallPut putnum;
    /* Double scalar and array declarations */
    double b, r, s, sigma;
    double *p = 0, *t = 0, *x = 0;
    /* Character scalar and array declarations */
    char put[8+1];
    Nag_OrderType order;
    INIT_FAIL(fail);

    printf("nag_asian_geom_price (s30sac) Example Program Results\n");
    printf("Asian Option: Geometric Continuous Average-Rate\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
s30sac.4
NAG Library Manual
s30sac.4
NAG Library Manual
/* Read put */
#ifdef _WIN32
    scanf_s("%8s%*\[\n\] ", put, _countof(put));
#else
    scanf("%8s%*\[\n\] ", put);
#endif

/* Read sigma, r */
#ifdef _WIN32
    scanf_s("%lf%lf%lf%lf%*\[\n\] ", &s, &sigma, &r, &b);
#else
    scanf("%lf%lf%lf%lf%*\[\n\] ", &s, &sigma, &r, &b);
#endif

/* Read m, n */
#ifdef _WIN32
    scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*\[\n\] ", &m, &n);
#else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*\[\n\] ", &m, &n);
#endif

#if defined NAG_COLUMN_MAJOR
    #define P(I, J) p[(J-1)*m + I-1]
#else
    #define P(I, J) p[(I-1)*n + J-1]
#endif

if (!(p = NAG_ALLOC(m*n, double)) ||
    !(t = NAG_ALLOC(n, double)) ||
    !(x = NAG_ALLOC(m, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read array of strike/exercise prices, X */
for (i = 0; i < m; i++)
#if defined _WIN32
    scanf_s("%lf ", &x[i]);
#else
    scanf("%lf ", &x[i]);
#endif

/* Read array of strike/exercise prices, T */
for (i = 0; i < n; i++)
#if defined _WIN32
    scanf_s("%lf ", &t[i]);
#else
    scanf("%lf ", &t[i]);
#endif

nag_asian_geom_price(order, putnum, m, n, x, s, t, sigma, r, b, p, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_asian_geom_price (s30sac).\n%sn\n", fail.message);
    exit_status = 1;
    goto END;
}
if (putnum == Nag_Call)
    printf("%s\n\n", "Asian Call :");
else if (putnum == Nag_Put)
    printf("%s\n\n", "Asian Put :");
printf("%8.4f\n", " Spot = ", s);
printf("%8.4f\n", " Volatility = ", sigma);
printf("%8.4f\n", " Rate = ", r);
printf("%8.4f\n", " Cost of carry = ", b);
printf("\n")
printf("%s\n", " Strike Expiry Option Price");
for (i = 1; i <= m; i++)
    for (j = 1; j <= n; j++)
        printf("%9.4f %9.4f %11.4f\n", x[i-1], t[j-1], P(i, j));
END:
    NAG_FREE(p);
    NAG_FREE(t);
    NAG_FREE(x);
    return exit_status;
}

10.2 Program Data

nag_asian_geom_price (s30sac) Example Program Data
Nag_Put : Nag_Call or Nag_Put
80.0 0.2 0.05 0.08 : s, sigma, r, b
1 1 : m, n
85.0 : X(I), I = 1,2,...m
0.25 : T(I), I = 1,2,...n

10.3 Program Results

nag_asian_geom_price (s30sac) Example Program Results
Asian Option: Geometric Continuous Average-Rate

Asian Put :
Spot = 80.0000
Volatility = 0.2000
Rate = 0.0500
Cost of carry = 0.0800

<table>
<thead>
<tr>
<th>Strike</th>
<th>Expiry</th>
<th>Option Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>85.0000</td>
<td>0.2500</td>
<td>4.6922</td>
</tr>
</tbody>
</table>