

NAG Library Function Document

nag_asian_geom_greeks (s30sbc)

1 Purpose

nag_asian_geom_greeks (s30sbc) computes the Asian geometric continuous average-rate option price together with its sensitivities (Greeks).

2 Specification

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

void nag_asian_geom_greeks (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[], double delta[], double gamma[],
    double vega[], double theta[], double rho[], double crho[], double vanna[],
    double charm[], double speed[], double colour[], double zomma[],
    double vomma[], NagError *fail)
```

3 Description

nag_asian_geom_greeks (s30sbc) computes the price of an Asian geometric continuous average-rate option, together with the Greeks or sensitivities, which are the partial derivatives of the option price with respect to certain of the other input parameters. The annual volatility, σ , risk-free rate, r , and cost of carry, b , are constants (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

$$P_{\text{call}} = Se^{(\bar{b}-r)T}\Phi(\bar{d}_1) - Xe^{-rT}\Phi(\bar{d}_2),$$

and the corresponding put option price is

$$P_{\text{put}} = Xe^{-rT}\Phi(-\bar{d}_2) - Se^{(\bar{b}-r)T}\Phi(-\bar{d}_1),$$

where

$$\bar{d}_1 = \frac{\ln(S/X) + (\bar{b} + \bar{\sigma}^2/2)T}{\bar{\sigma}\sqrt{T}}$$

and

$$\bar{d}_2 = \bar{d}_1 - \bar{\sigma}\sqrt{T},$$

with

$$\bar{\sigma} = \frac{\sigma}{\sqrt{3}}, \quad \bar{b} = \frac{1}{2}\left(b - \frac{\sigma^2}{6}\right).$$

Φ is the cumulative Normal distribution function,

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

4 References

Kemna A and Vorst A (1990) A pricing method for options based on average asset values *Journal of Banking and Finance* **14** 113–129

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** \geq 1.
- 4: **n** – Integer *Input*
On entry: the number of times to expiry to be used.
Constraint: **n** \geq 1.
- 5: **x[m]** – const double *Input*
On entry: **x**[*i* – 1] must contain X_i , the *i*th strike price, for $i = 1, 2, \dots, \mathbf{m}$.
Constraint: **x**[*i* – 1] $\geq z$ and **x**[*i* – 1] $\leq 1/z$, where $z = \text{nag_real_safe_small_number}$, the safe range parameter, for $i = 1, 2, \dots, \mathbf{m}$.
- 6: **s** – double *Input*
On entry: S , the price of the underlying asset.
Constraint: **s** $\geq z$ and **s** $\leq 1/z$, where $z = \text{nag_real_safe_small_number}$, 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 $i = 1, 2, \dots, \mathbf{n}$.
Constraint: **t**[*i* – 1] $\geq z$, where $z = \text{nag_real_safe_small_number}$, the safe range parameter, for $i = 1, 2, \dots, \mathbf{n}$.
- 8: **sigma** – double *Input*
On entry: σ , 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: $r \geq 0.0$.
- 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** × **n**] – double *Output*
Note: the (i, j) th element of the matrix P is stored in
 $\mathbf{p}[(j - 1) \times \mathbf{m} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{p}[(i - 1) \times \mathbf{n} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the $m \times n$ array **p** contains the computed option prices.
- 12: **delta**[**m** × **n**] – double *Output*
Note: the (i, j) th element of the matrix is stored in
 $\mathbf{delta}[(j - 1) \times \mathbf{m} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{delta}[(i - 1) \times \mathbf{n} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the $m \times n$ array **delta** contains the sensitivity, $\frac{\partial P}{\partial S}$, of the option price to change in the price of the underlying asset.
- 13: **gamma**[**m** × **n**] – double *Output*
Note: the (i, j) th element of the matrix is stored in
 $\mathbf{gamma}[(j - 1) \times \mathbf{m} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{gamma}[(i - 1) \times \mathbf{n} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the $m \times n$ array **gamma** contains the sensitivity, $\frac{\partial^2 P}{\partial S^2}$, of **delta** to change in the price of the underlying asset.
- 14: **vega**[**m** × **n**] – double *Output*
Note: the (i, j) th element of the matrix is stored in
 $\mathbf{vega}[(j - 1) \times \mathbf{m} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{vega}[(i - 1) \times \mathbf{n} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the $m \times n$ array **vega** contains the sensitivity, $\frac{\partial P}{\partial \sigma}$, of the option price to change in the volatility of the underlying asset.
- 15: **theta**[**m** × **n**] – double *Output*
Note: the (i, j) th element of the matrix is stored in
 $\mathbf{theta}[(j - 1) \times \mathbf{m} + i - 1]$ when **order** = Nag_ColMajor;
 $\mathbf{theta}[(i - 1) \times \mathbf{n} + j - 1]$ when **order** = Nag_RowMajor.
On exit: the $m \times n$ array **theta** contains the sensitivity, $-\frac{\partial P}{\partial T}$, of the option price to change in the time to expiry of the option.
- 16: **rho**[**m** × **n**] – double *Output*
Note: the (i, j) th element of the matrix is stored in

rho[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

rho[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **rho** contains the sensitivity, $\frac{\partial P}{\partial r}$, of the option price to change in the annual risk-free interest rate.

17: **crho**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

crho[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

crho[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **crho** containing the sensitivity, $\frac{\partial P}{\partial b}$, of the option price to change in the annual cost of carry rate, b .

18: **vanna**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

vanna[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

vanna[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **vanna** contains the sensitivity, $\frac{\partial^2 P}{\partial S \partial \sigma}$, of **vega** to change in the price of the underlying asset or, equivalently, the sensitivity of **delta** to change in the volatility of the asset price.

19: **charm**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

charm[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

charm[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **charm** contains the sensitivity, $-\frac{\partial^2 P}{\partial S \partial T}$, of **delta** to change in the time to expiry of the option.

20: **speed**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

speed[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

speed[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **speed** contains the sensitivity, $\frac{\partial^3 P}{\partial S^3}$, of **gamma** to change in the price of the underlying asset.

21: **colour**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

colour[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

colour[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **colour** contains the sensitivity, $-\frac{\partial^3 P}{\partial S^2 \partial T}$, of **gamma** to change in the time to expiry of the option.

22: **zomma**[**m** \times **n**] – double *Output*

Note: the (i, j)th element of the matrix is stored in

zomma[($j - 1$) \times **m** + $i - 1$] when **order** = Nag_ColMajor;

zomma[($i - 1$) \times **n** + $j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **zomma** contains the sensitivity, $\frac{\partial^3 P}{\partial S^2 \partial \sigma}$, of **gamma** to change in the volatility of the underlying asset.

23: **vomma**[$m \times n$] – double

Output

Note: the (i, j) th element of the matrix is stored in

vomma[($j - 1$) \times $m + i - 1$] when **order** = Nag_ColMajor;

vomma[($i - 1$) \times $n + j - 1$] when **order** = Nag_RowMajor.

On exit: the $m \times n$ array **vomma** contains the sensitivity, $\frac{\partial^2 P}{\partial \sigma^2}$, of **vega** to change in the volatility of the underlying asset.

24: **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.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **m** = $\langle value \rangle$.

Constraint: **m** ≥ 1 .

On entry, **n** = $\langle value \rangle$.

Constraint: **n** ≥ 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.

NE_REAL

On entry, **r** = $\langle value \rangle$.

Constraint: **r** ≥ 0.0 .

On entry, **s** = $\langle value \rangle$.

Constraint: **s** $\geq \langle value \rangle$ and **s** $\leq \langle value \rangle$.

On entry, **sigma** = $\langle value \rangle$.

Constraint: **sigma** > 0.0 .

NE_REAL_ARRAY

On entry, **t**[$\langle value \rangle$] = $\langle value \rangle$.

Constraint: **t**[i] $\geq \langle value \rangle$.

On entry, **x**[$\langle value \rangle$] = $\langle value \rangle$.

Constraint: **x**[i] $\geq \langle value \rangle$ and **x**[i] $\leq \langle value \rangle$.

7 Accuracy

The accuracy of the output is dependent on the accuracy of the cumulative Normal distribution function, Φ . 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 Further Comments

None.

9 Example

This example computes the price of an Asian geometric continuous average-rate call with a time to expiry of 3 months, a stock price of 80 and a strike price of 97. The risk-free interest rate is 5% per year, the cost of carry is 8% and the volatility is 20% per year.

9.1 Program Text

```

/* nag_asian_geom_greeks (s30sbc) Example Program.
 *
 * Copyright 2009, Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nags.h>

int main(int argc, char *argv[])
{
    FILE          *fpin, *fpout;
    /*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        *charm = 0, *colour = 0, *crho = 0, *delta = 0, *gamma = 0;
    double        *p = 0, *rho = 0, *speed = 0, *t = 0, *theta = 0, *vanna = 0;
    double        *vega = 0, *vomma = 0, *x = 0, *zomma = 0;
    /*Character scalar and array declarations */
    char          put[8+1];
    Nag_OrderType order;

    INIT_FAIL(fail);

    /* Check for command-line IO options */
    fpin = nag_example_file_io(argc, argv, "-data", NULL);
    fpout = nag_example_file_io(argc, argv, "-results", NULL);

    fprintf(fpout, "nag_asian_geom_greeks (s30sbc) Example Program Results\n");
    fprintf(fpout, "Asian Option: Geometric Continuous Average-Rate\n\n");
    /* Skip heading in data file*/
    fscanf(fpin, "%*[\n] ");
    /* Read put*/
    fscanf(fpin, "%s*[\n] ", put);
    /*
     * nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    putnum = (Nag_CallPut) nag_enum_name_to_value(put);
    /* Read s, sigma, r, b*/
    fscanf(fpin, "%lf%lf%lf%lf*[\n] ", &s, &sigma, &r, &b);
    /* Read m, n*/

```

```

fscanf(fpin, "%ld%ld%*[\n] ", &m, &n);
#ifdef NAG_COLUMN_MAJOR
#define CHARM(I, J)  charm[(J-1)*m + I-1]
#define COLOUR(I, J) colour[(J-1)*m + I-1]
#define CRHO(I, J)  crho[(J-1)*m + I-1]
#define DELTA(I, J) delta[(J-1)*m + I-1]
#define GAMMA(I, J) gamma[(J-1)*m + I-1]
#define P(I, J)     p[(J-1)*m + I-1]
#define RHO(I, J)   rho[(J-1)*m + I-1]
#define SPEED(I, J) speed[(J-1)*m + I-1]
#define THETA(I, J) theta[(J-1)*m + I-1]
#define VANNA(I, J) vanna[(J-1)*m + I-1]
#define VEGA(I, J)  vega[(J-1)*m + I-1]
#define VOMMA(I, J) vomma[(J-1)*m + I-1]
#define ZOMMA(I, J) zomma[(J-1)*m + I-1]
order = Nag_ColMajor;
#else
#define CHARM(I, J)  charm[(I-1)*n + J-1]
#define COLOUR(I, J) colour[(I-1)*n + J-1]
#define CRHO(I, J)  crho[(I-1)*n + J-1]
#define DELTA(I, J) delta[(I-1)*n + J-1]
#define GAMMA(I, J) gamma[(I-1)*n + J-1]
#define P(I, J)     p[(I-1)*n + J-1]
#define RHO(I, J)   rho[(I-1)*n + J-1]
#define SPEED(I, J) speed[(I-1)*n + J-1]
#define THETA(I, J) theta[(I-1)*n + J-1]
#define VANNA(I, J) vanna[(I-1)*n + J-1]
#define VEGA(I, J)  vega[(I-1)*n + J-1]
#define VOMMA(I, J) vomma[(I-1)*n + J-1]
#define ZOMMA(I, J) zomma[(I-1)*n + J-1]
order = Nag_RowMajor;
#endif
if (
    !(charm = NAG_ALLOC(m*n, double)) ||
    !(colour = NAG_ALLOC(m*n, double)) ||
    !(crho = NAG_ALLOC(m*n, double)) ||
    !(delta = NAG_ALLOC(m*n, double)) ||
    !(gamma = NAG_ALLOC(m*n, double)) ||
    !(p = NAG_ALLOC(m*n, double)) ||
    !(rho = NAG_ALLOC(m*n, double)) ||
    !(speed = NAG_ALLOC(m*n, double)) ||
    !(t = NAG_ALLOC(n, double)) ||
    !(theta = NAG_ALLOC(m*n, double)) ||
    !(vanna = NAG_ALLOC(m*n, double)) ||
    !(vega = NAG_ALLOC(m*n, double)) ||
    !(vomma = NAG_ALLOC(m*n, double)) ||
    !(x = NAG_ALLOC(m, double)) ||
    !(zomma = NAG_ALLOC(m*n, double))
    )
{
    fprintf(fpout, "Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read array of strike/exercise prices, X*/
for (i = 0; i < m; i++)
    fscanf(fpin, "%lf ", &x[i]);
fscanf(fpin, "%*[\n] ");
/* Read array of times to expiry*/
for (i = 0; i < n; i++)
    fscanf(fpin, "%lf ", &t[i]);
fscanf(fpin, "%*[\n] ");
/*
 * nag_asian_geom_greeks (s30sbc)
 * Asian option: geometric continuous average rate pricing formula
 * with Greeks
 */
nag_asian_geom_greeks(order, putnum, m, n, x, s, t, sigma, r, b, p,
    delta, gamma, vega, theta, rho, crho, vanna,
    charm, speed, colour, zomma, vomma, &fail);
if (fail.code != NE_NOERROR)
    {

```

```

        fprintf(fpout, "Error from nag_asian_geom_greeks (s30sbc).\n%s\n",
                fail.message);
        exit_status = 1;
        goto END;
    }
    if (putnum == Nag_Call)
    {
        fprintf(fpout, "%s\n\n", "Asian Call :");
    }
    else if (putnum == Nag_Put)
    {
        fprintf(fpout, "%s\n\n", "Asian Put :");
    }
    fprintf(fpout, " Spot           = %8.4f\n", s);
    fprintf(fpout, " Volatility      = %8.4f\n", sigma);
    fprintf(fpout, " Rate           = %8.4f\n", r);
    fprintf(fpout, " Cost of carry = %8.4f\n", b);
    fprintf(fpout, "\n");
    for (j = 1; j <= n; j++)
    {
        fprintf(fpout, "\n Time to Expiry :   %8.4f\n", t[j-1]);
        fprintf(fpout, " Strike Price Delta Gamma Vega "
                "Theta Rho CRho\n");
        for (i = 1; i <= m; i++)
            fprintf(fpout, "%8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n",
                    x[i-1], P(i, j), DELTA(i, j), GAMMA(i, j), VEGA(i, j),
                    THETA(i, j), RHO(i, j), CRHO(i, j));
        fprintf(fpout, " Vanna Charm Speed "
                "Colour Zomma Vomma\n");
        for (i = 1; i <= m; i++)
            fprintf(fpout, "%26.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n", VANNA(i, j),
                    CHARM(i, j), SPEED(i, j), COLOUR(i, j), ZOMMA(i, j),
                    VOMMA(i, j));
    }

END:
    if (fpin != stdin) fclose(fpin);
    if (fpout != stdout) fclose(fpout);
    if (charm) NAG_FREE(charm);
    if (colour) NAG_FREE(colour);
    if (crho) NAG_FREE(crho);
    if (delta) NAG_FREE(delta);
    if (gamma) NAG_FREE(gamma);
    if (p) NAG_FREE(p);
    if (rho) NAG_FREE(rho);
    if (speed) NAG_FREE(speed);
    if (t) NAG_FREE(t);
    if (theta) NAG_FREE(theta);
    if (vanna) NAG_FREE(vanna);
    if (vega) NAG_FREE(vega);
    if (vomma) NAG_FREE(vomma);
    if (x) NAG_FREE(x);
    if (zomma) NAG_FREE(zomma);

    return exit_status;
}

```

9.2 Program Data

```

nag_asian_geom_greeks (s30sbc) Example Program Data
Nag_Call           : Nag_Call or Nag_Put
80.0 0.2 0.05 0.08 : s, sigma, r, b
1 1                : m, n
97.0               : X(I), I = 1,2,...m
0.25               : T(I), I = 1,2,...n

```

9.3 Program Results

nag_asian_geom_greeks (s30sbc) Example Program Results
 Asian Option: Geometric Continuous Average-Rate

Asian Call :

Spot = 80.0000
 Volatility = 0.2000
 Rate = 0.0500
 Cost of carry = 0.0800

Time to Expiry :	0.2500						
Strike Price		Delta	Gamma	Vega	Theta	Rho	CRho
97.0000	0.0010	0.0008	0.0006	0.0638	-0.0281	0.0079	0.0081
		Vanna	Charm	Speed	Colour	Zomma	Vomma
		0.0443	-0.0196	0.0004	-0.0122	0.0272	3.1893
