NAG Library Function Document

nag_bsm_price (s30aac)

1 Purpose

nag_bsm_price (s30aac) computes the European option price given by the Black–Scholes–Merton formula.

2 Specification

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

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

3 Description

nag_bsm_price (s30aac) computes the price of a European call (or put) option for constant volatility, σ, and risk-free interest rate, r, with a possible dividend yield, q, using the Black–Scholes–Merton formula (see Black and Scholes (1973) and Merton (1973)). For a given strike price, X, the price of a European call with underlying price, S, and time to expiry, T, is

$$P_{\text{call}} = Se^{-qT}\Phi(d_1) - Xe^{-rT}\Phi(d_2)$$

and the corresponding European put price is

$$P_{\text{put}} = Xe^{-rT}\Phi(-d_2) - Se^{-qT}\Phi(-d_1)$$

and where Φ denotes the cumulative Normal distribution function,

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

and

$$d_1 = \frac{\ln(S/X) + \left(r - q + \sigma^2/2\right)T}{\sigma\sqrt{T}},$$

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

The option price $P_{ij} = P\left(X = X_i, T = T_j\right)$ 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

Black F and Scholes M (1973) The pricing of options and corporate liabilities *Journal of Political Economy* **81** 637–654

Merton R C (1973) Theory of rational option pricing *Bell Journal of Economics and Management Science* **4** 141–183

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 ith strike price, for $i = 1, 2, \dots,$ **m**.

Constraint: **x**$[i-1] \geq z$ and **x**$[i-1] \leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \dots,$ **m**.

6: **s** – double *Input*

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

Constraint: **s** $\geq z$ and **s** $\leq 1.0/z$, 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 ith time, in years, to expiry, for $i = 1, 2, \dots,$ **n**.

Constraint: **t**$[i-1] \geq z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \dots,$ **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** ≥ 0.0.

10: **q** – double *Input*

On entry: q, the annual continuous yield rate. Note that a rate of 8% should be entered as 0.08.

Constraint: **q** ≥ 0.0.

11: **p**[**m** × **n**] – double *Output*

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

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

On exit: $\mathbf{P}(i,j)$ contains P_{ij}, the option price evaluated for the strike price $\mathbf{x}_i$ at expiry $\mathbf{t}_j$ for $i=1,2,\ldots,\mathbf{m}$ and $j=1,2,\ldots,\mathbf{n}$.

12: **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: $\mathbf{m}\geq 1$.

On entry, **n** = ⟨*value*⟩.
Constraint: $\mathbf{n}\geq 1$.

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.

NE_REAL

On entry, **q** = ⟨*value*⟩.
Constraint: $\mathbf{q}\geq 0.0$.

On entry, **r** = ⟨*value*⟩.
Constraint: $\mathbf{r}\geq 0.0$.

On entry, **s** = ⟨*value*⟩.
Constraint: **s** ≥ ⟨*value*⟩ and **s** ≤ ⟨*value*⟩.

On entry, **sigma** = ⟨*value*⟩.
Constraint: $\mathbf{sigma}>0.0$.

NE_REAL_ARRAY

On entry, **t**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **t**[*i*] ≥ ⟨*value*⟩.

On entry, **x**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **x**[*i*] ≥ ⟨*value*⟩ and **x**[*i*] ≤ ⟨*value*⟩.

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 Parallelism and Performance

nag_bsm_price (s30aac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

This example computes the prices for six European call options using two expiry times and three strike prices as input. The times to expiry are taken as 0.7 and 0.8 years respectively. The stock price is 55, with strike prices, 58, 60 and 62. The risk-free interest rate is 10% per year and the volatility is 30% per year.

10.1 Program Text

```
/* nag_bsm_price (s30aac) Example Program.
 *
 * Copyright 2009, Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.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        q, 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_bsm_price (s30aac) Example Program Results\n");
  printf("Black-Scholes-Merton formula\n\n");
  /* Skip heading in data file */
  scanf("%*[^\n] ");
  /* Read put */
  scanf("%8s%*[^\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 sigma, r */
  scanf("%lf%lf%lf%lf%*[^\n] ", &s, &sigma, &r, &q);
  /* Read m, n */
  scanf("%ld%ld%*[^\n] ", &m, &n);
    #ifdef NAG_COLUMN_MAJOR
    #define P(I, J) p[(J-1)*m + I-1]
```

```
  order = Nag_ColMajor;
    #else
    #define P(I, J) p[(I-1)*n + J-1]
  order = Nag_RowMajor;
    #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++)
    scanf("%lf ", &x[i]);
  scanf("%*[^\n] ");
  for (i = 0; i < n; i++)
    scanf("%lf ", &t[i]);
  scanf("%*[^\n] ");
  /*
   * nag_bsm_price (s30aac)
   * Black-Scholes-Merton option pricing formula
   */
  nag_bsm_price(order, putnum, m, n, x, s, t, sigma, r, q, p, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_bsm_price (s30aac).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  if (putnum == Nag_Call)
    printf("%s\n", "European Call :");
  else if (putnum == Nag_Put)
    printf("%s\n", "European Put :");
  printf("%s%8.4f\n", "  Spot       = ", s);
  printf("%s%8.4f\n", "  Volatility = ", sigma);
  printf("%s%8.4f\n", "  Rate       = ", r);
  printf("%s%8.4f\n", "  Dividend   = ", q);
  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 %9.4f\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_bsm_price (s30aac) Example Program Data
Nag_Call               : Nag_Call or Nag_Put
55.0 0.3 0.1 0.0       : s, sigma, r, q
3 2                    : m, n
58.0
60.0
62.0                   : x(i), i = 1,2,...m
0.7
0.8                    : t(i), i = 1,2,...n
```

10.3 Program Results

```
nag_bsm_price (s30aac) Example Program Results
Black-Scholes-Merton formula

European Call :
  Spot       =  55.0000
  Volatility =   0.3000
  Rate       =   0.1000
  Dividend   =   0.0000

  Strike     Expiry    Option Price
  58.0000    0.7000    5.9198
  58.0000    0.8000    6.5506
  60.0000    0.7000    5.0809
  60.0000    0.8000    5.6992
  62.0000    0.7000    4.3389
  62.0000    0.8000    4.9379
```