NAG Library Function Document

nag_lookback_fls_price (s30bac)

1 Purpose

nag_lookback_fls_price (s30bac) computes the price of a floating-strike lookback option.

2 Specification

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

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

3 Description

nag_lookback_fls_price (s30bac) computes the price of a floating-strike lookback call or put option. A call option of this type confers the right to buy the underlying asset at the lowest price, S_{min}, observed during the lifetime of the contract. A put option gives the holder the right to sell the underlying asset at the maximum price, S_{max}, observed during the lifetime of the contract. Thus, at expiry, the payoff for a call option is $S - S_{\text{min}}$, and for a put, $S_{\text{max}} - S$.

For a given minimum value the price of a floating-strike lookback call with underlying asset price, S, and time to expiry, T, is

$$P_{\text{call}} = Se^{-qT}\Phi(a_1) - S_{\text{min}}e^{-rT}\Phi(a_2) + Se^{-rT}\frac{\sigma^2}{2b}\left[\left(\frac{S}{S_{\text{min}}}\right)^{-2b/\sigma^2}\Phi\left(-a_1 + \frac{2b}{\sigma}\sqrt{T}\right) - e^{bT}\Phi(-a_1)\right],$$

where $b = r - q \neq 0$. The volatility, σ, risk-free interest rate, r, and annualised dividend yield, q, are constants. When $r = q$, the option price is given by

$$P_{\text{call}} = Se^{-qT}\Phi(a_1) - S_{\text{min}}e^{-rT}\Phi(a_2) + Se^{-rT}\sigma\sqrt{T}[\phi(a_1) + a_1(\Phi(a_1) - 1)].$$

The corresponding put price is (for $b \neq 0$),

$$P_{\text{put}} = S_{\text{max}}e^{-rT}\Phi(-a_2) - Se^{-qT}\Phi(-a_1) + Se^{-rT}\frac{\sigma^2}{2b}\left[-\left(\frac{S}{S_{\text{max}}}\right)^{-2b/\sigma^2}\Phi\left(a_1 - \frac{2b}{\sigma}\sqrt{T}\right) + e^{bT}\Phi(a_1)\right].$$

When $r = q$,

$$P_{\text{put}} = S_{\text{max}}e^{-rT}\Phi(-a_2) - Se^{-qT}\Phi(-a_1) + Se^{-rT}\sigma\sqrt{T}[\phi(a_1) + a_1\Phi(a_1)].$$

In the above, Φ denotes the cumulative Normal distribution function,

$$\Phi(x) = \int_{-\infty}^{x}\phi(y)dy$$

where ϕ denotes the standard Normal probability density function

$$\phi(y) = \frac{1}{\sqrt{2\pi}}\exp\left(-y^2/2\right)$$

and

$$a_1 = \frac{\ln(S/S_{\text{m}}) + \left(b + \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

$$a_2 = a_1 - \sigma\sqrt{T}$$

where S_m is taken to be the minimum price attained by the underlying asset, S_{min}, for a call and the maximum price, S_{max}, for a put.

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

4 References

Goldman B M, Sosin H B and Gatto M A (1979) Path dependent options: buy at the low, sell at the high *Journal of Finance* **34** 1111–1127

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 minimum or maximum 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: **sm**[**m**] – const double *Input*

On entry: **sm**$[i-1]$ must contain $S_{\text{min}}(i)$, the ith minimum observed price of the underlying asset when **option** = Nag_Call, or $S_{\text{max}}(i)$, the maximum observed price when **option** = Nag_Put, for $i = 1, 2, \ldots,$ **m**.

Constraints:

sm$[i-1] \geq z$ and **sm**$[i-1] \leq 1/z$, where z = nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots,$ **m**;
if **option** = Nag_Call, **sm**$[i-1] \leq S$, for $i = 1, 2, \ldots,$ **m**;
if **option** = Nag_Put, **sm**$[i-1] \geq S$, for $i = 1, 2, \ldots,$ **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: $\mathbf{t}[i-1]$ must contain T_i, the ith time, in years, to expiry, for $i = 1, 2, \ldots, \mathbf{n}$.

Constraint: $\mathbf{t}[i-1] \geq z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots, \mathbf{n}$.

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: $\mathbf{sigma} > 0.0$.

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: $\mathbf{r} \geq 0.0$.

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: $\mathbf{q} \geq 0.0$.

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 minimum or maximum observed price $S_{\text{min}}(i)$ or $S_{\text{max}}(i)$ 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: **q** ≥ 0.0.

On entry, **r** = ⟨*value*⟩.
Constraint: **r** ≥ 0.0.

On entry, **s** = ⟨*value*⟩.
Constraint: **s** $\geq$ ⟨*value*⟩ and **s** $\leq$ ⟨*value*⟩.

On entry, **sigma** = ⟨*value*⟩.
Constraint: **sigma** > 0.0.

NE_REAL_ARRAY

On entry, **sm**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ $\leq$ **sm**[i] $\leq$ ⟨*value*⟩ for all i.

On entry, **t**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: **t**[i] $\geq$ ⟨*value*⟩ for all i.

On entry with a call option, **sm**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: for call options, **sm**[i] $\leq$ ⟨*value*⟩ for all i.

On entry with a put option, **sm**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: for put options, **sm**[i] $\geq$ ⟨*value*⟩ for all i.

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_lookback_fls_price (s30bac) 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 price of a floating-strike lookback call with a time to expiry of 6 months and a stock price of 120. The minimum price observed so far is 100. The risk-free interest rate is 10% per year and the volatility is 30% per year with an annual dividend return of 6%.

10.1 Program Text

```
/* nag_lookback_fls_price (s30bac) Example Program.
 *
 * Copyright 2008, 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, *sm = 0, *t = 0;
  /* Character scalar and array declarations */
  char          put[8+1];
  Nag_OrderType order;

  INIT_FAIL(fail);

  printf("nag_lookback_fls_price (s30bac) Example Program Results\n");
  printf("Floating-strike Lookback\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 s, sigma, r, q */
  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)) ||
      !(sm = NAG_ALLOC(m, double)) ||
      !(t = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* Read array of min/max prices, SM */
  for (i = 0; i < m; i++)
    scanf("%lf ", &sm[i]);
  scanf("%*[^\n] ");
  /* Read array of times to expiry */
  for (i = 0; i < n; i++)
    scanf("%lf ", &t[i]);
  scanf("%*[^\n] ");
  /*
   * nag_lookback_fls_price (s30bac)
   * Floating-strike lookback option pricing formula
   */
  nag_lookback_fls_price(order, putnum, m, n, sm, s, t, sigma, r, q, p,
                         &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_lookback_fls_price (s30bac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }
```

```
  if (putnum == Nag_Call)
    printf("European Call :\n\n");
  else if (putnum == Nag_Put)
    printf("European Put :\n\n");
  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 %12.4f\n", sm[i-1], t[j-1], P(i, j));

 END:
  NAG_FREE(p);
  NAG_FREE(sm);
  NAG_FREE(t);

  return exit_status;
}
```

10.2 Program Data

```
nag_lookback_fls_price (s30bac) Example Program Data
 Nag_Call              : Nag_Call or Nag_Put
 120.0 0.3 0.1 0.06    : s, sigma, r, q
 1 1                   : m, n
 100.0                 : SM(I), I = 1,2,...m
 0.5                   : T(I), I = 1,2,...n
```

10.3 Program Results

```
nag_lookback_fls_price (s30bac) Example Program Results
Floating-strike Lookback

European Call :

  Spot        = 120.0000
  Volatility =   0.3000
  Rate        =   0.1000
  Dividend    =   0.0600

  Strike      Expiry     Option Price
 100.0000    0.5000      25.3534
```