

# 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  $S - S_{\min}$ , and for a put,  $S_{\max} - S$ .

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

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

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

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

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

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

When  $r = q$ ,

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

In the above,  $\Phi$  denotes the cumulative Normal distribution function,

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

where  $\phi$  denotes the standard Normal probability density function

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

and

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

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

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  $S_{\min}(i)$  or  $S_{\max}(i)$ ,  $i = 1, 2, \dots, m$ , and for each expiry time in a set  $T_j$ ,  $j = 1, 2, \dots, n$ .

## 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**  $\geq$  1.
  
- 4: **n** – Integer *Input*  
*On entry:* the number of times to expiry to be used.  
*Constraint:* **n**  $\geq$  1.
  
- 5: **sm[m]** – const double *Input*  
*On entry:* **sm**[ $i - 1$ ] must contain  $S_{\min}(i)$ , the  $i$ th minimum observed price of the underlying asset when **option** = Nag\_Call, or  $S_{\max}(i)$ , the maximum observed price when **option** = Nag\_Put, for  $i = 1, 2, \dots, \mathbf{m}$ .  
*Constraints:*  
**sm**[ $i - 1$ ]  $\geq z$  and **sm**[ $i - 1$ ]  $\leq 1/z$ , where  $z = \text{nag\_real\_safe\_small\_number}$ , the safe range parameter, for  $i = 1, 2, \dots, \mathbf{m}$ ;  
if **option** = Nag\_Call, **sm**[ $i - 1$ ]  $\leq S$ , for  $i = 1, 2, \dots, \mathbf{m}$ ;  
if **option** = Nag\_Put, **sm**[ $i - 1$ ]  $\geq S$ , 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**  $\leq 1.0/z$ , 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:*  $\sigma$ , 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**  $\geq 0.0$ .
- 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:* **q**  $\geq 0.0$ .
- 11: **p[m × n]** – double *Output*  
**Note:** where **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:* **P**( $i, j$ ) contains  $P_{ij}$ , the option price evaluated for the minimum or maximum observed price  $S_{\min}(i)$  or  $S_{\max}(i)$  at expiry  $\mathbf{t}_j$  for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .
- 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.  
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,  $\mathbf{q} = \langle \text{value} \rangle$ .  
Constraint:  $\mathbf{q} \geq 0.0$ .

On entry,  $\mathbf{r} = \langle \text{value} \rangle$ .  
Constraint:  $\mathbf{r} \geq 0.0$ .

On entry,  $\mathbf{s} = \langle \text{value} \rangle$ .  
Constraint:  $\mathbf{s} \geq \langle \text{value} \rangle$  and  $\mathbf{s} \leq \langle \text{value} \rangle$ .

On entry,  $\mathbf{\sigma} = \langle \text{value} \rangle$ .  
Constraint:  $\mathbf{\sigma} > 0.0$ .

### NE\_REAL\_ARRAY

On entry,  $\mathbf{sm}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .  
Constraint:  $\langle \text{value} \rangle \leq \mathbf{sm}[i] \leq \langle \text{value} \rangle$  for all  $i$ .

On entry,  $\mathbf{t}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .  
Constraint:  $\mathbf{t}[i] \geq \langle \text{value} \rangle$  for all  $i$ .

On entry with a call option,  $\mathbf{sm}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .  
Constraint: for call options,  $\mathbf{sm}[i] \leq \langle \text{value} \rangle$  for all  $i$ .

On entry with a put option,  $\mathbf{sm}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .  
Constraint: for put options,  $\mathbf{sm}[i] \geq \langle \text{value} \rangle$  for all  $i$ .

## 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_lookback_fls_price` (s30bac) 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 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 2014 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 */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
    /* Read put */
#ifdef _WIN32
    scanf_s("%8s%*[\n] ", put, _countof(put));
#else
    scanf("%8s%*[\n] ", put);
#endif
    /*
     * 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 */
#ifdef _WIN32
    scanf_s("%lf%lf%lf%lf%*[\n] ", &s, &sigma, &r, &q);
#else
    scanf("%lf%lf%lf%lf%*[\n] ", &s, &sigma, &r, &q);
#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
    #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++)
#ifdef _WIN32
        scanf_s("%lf ", &sm[i]);
#else
        scanf("%lf ", &sm[i]);
#endif
#ifdef _WIN32
        scanf_s("%*[\n] ");
#else
        scanf("%*[\n] ");
#endif
    /* Read array of times to expiry */
    for (i = 0; i < n; i++)
#ifdef _WIN32
        scanf_s("%lf ", &t[i]);
#else
        scanf("%lf ", &t[i]);
#endif
#ifdef _WIN32
        scanf_s("%*[\n] ");
#else
        scanf("%*[\n] ");
#endif
    /*
     * 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
```

---