# NAG Library Function Document

# nag_binary_con_price (s30cac)

## 1 Purpose

nag_binary_con_price (s30cac) computes the price of a binary or digital cash-or-nothing option.

## 2 Specification

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

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

## 3 Description

nag_binary_con_price (s30cac) computes the price of a binary or digital cash-or-nothing option which pays a fixed amount, $K$, at expiration if the option is in-the-money (see Section 2.4 in the s Chapter Introduction). For a strike price, $X$, underlying asset price, $S$, and time to expiry, $T$, the payoff is therefore $K$, if $S > X$ for a call or $S < X$ for a put. Nothing is paid out when this condition is not met.

The price of a call with volatility, $\sigma$, risk-free interest rate, $r$, and annualised dividend yield, $q$, is

$$P_{\text{call}} = Ke^{-rT}\Phi(d_2)$$

and for a put,

$$P_{\text{put}} = Ke^{-rT}\Phi(-d_2)$$

where $\Phi$ is the cumulative Normal distribution function,

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

and

$$d_2 = \frac{\ln(S/X) + \left(r - q - \sigma^2/2\right)T}{\sigma\sqrt{T}}.$$

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

Reiner E and Rubinstein M (1991) Unscrambling the binary code *Risk* **4**

## 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*: $\mathbf{m} \geq 1$.

4: **n** – Integer *Input*

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

*Constraint*: $\mathbf{n} \geq 1$.

5: **x**[**m**] – const double *Input*

*On entry*: $\mathbf{x}[i-1]$ must contain $X_i$, the $i$th strike price, for $i = 1, 2, \ldots, \mathbf{m}$.

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

6: **s** – double *Input*

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

*Constraint*: $\mathbf{s} \geq z$ and $\mathbf{s} \leq 1.0/z$, where $z =$ nag_real_safe_small_number, the safe range parameter.

7: **k** – double *Input*

*On entry*: the amount, $K$, to be paid at expiration if the option is in-the-money, i.e., if $\mathbf{s} > \mathbf{x}[i-1]$ when **option** = Nag_Call, or if $\mathbf{s} < \mathbf{x}[i-1]$ when **option** = Nag_Put, for $i = 1, 2, \ldots, m$.

*Constraint*: $\mathbf{k} \geq 0.0$.

8: **t**[**n**] – const double *Input*

*On entry*: $\mathbf{t}[i-1]$ must contain $T_i$, the $i$th 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 $i = 1, 2, \ldots, \mathbf{n}$.

9: **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*: $\mathbf{sigma} > 0.0$.

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

11: **q** – double *Input*

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

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

12: $\mathbf{p}[\mathbf{m} \times \mathbf{n}]$ – 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 $j = 1, 2, \ldots, \mathbf{n}$.

13: **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, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 1$.

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

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

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

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

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

**NE_REAL_ARRAY**

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

On entry, $\mathbf{x}[\langle value \rangle] = \langle value \rangle$.
Constraint: $\mathbf{x}[i] \geq \langle value \rangle$ and $\mathbf{x}[i] \leq \langle value \rangle$.

## 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_binary_con_price (s30cac) 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 cash-or-nothing put with a time to expiry of 0.75 years, a stock price of 100 and a strike price of 80. The risk-free interest rate is 6% per year and the volatility is 35% per year. If the option is in-the-money at expiration, i.e., if $S > X$, the payoff is 10.

### 10.1 Program Text

```
/* nag_binary_con_price (s30cac) 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        k, 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_binary_con_price (s30cac) Example Program Results\n");
  printf("Binary (Digital): Cash-or-Nothing\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, k, sigma, r, q */
  scanf("%lf%lf%lf%lf%lf%*[^\n] ", &s, &k, &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_binary_con_price (s30cac)
   * Binary option: cash-or-nothing pricing formula
   */
  nag_binary_con_price(order, putnum, m, n, x, s, k, t, sigma, r, q, p,
                       &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_binary_con_price (s30cac).\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", "  Payout     = ", k);
  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", 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_binary_con_price (s30cac) Example Program Data
 Nag_Put                   : Nag_Call or Nag_Put
 100.0 10.0 0.35 0.06 0.0 : s, k, sigma, r, q
 1 1                       : m, n
 80.0                      : X(I), I = 1,2,...m
 0.75                      : T(I), I = 1,2,...n
```

## 10.3 Program Results

```
nag_binary_con_price (s30cac) Example Program Results
Binary (Digital): Cash-or-Nothing

European Put :

  Spot       = 100.0000
  Payout     =  10.0000
  Volatility =   0.3500
  Rate       =   0.0600
  Dividend   =   0.0000

   Strike    Expiry    Option Price
  80.0000    0.7500       2.2155
```