NAG Library Function Document

nag_binary_aon_greeks (s30cdc)

1 Purpose

nag_binary_aon_greeks (s30cdc) computes the price of a binary or digital asset-or-nothing option together with its sensitivities (Greeks).

2 Specification

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

void nag_binary_aon_greeks (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[], 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_binary_aon_greeks (s30cdc) computes the price of a binary or digital asset-or-nothing 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. This option pays the underlying asset itself, $S$, at expiration if the option is in-the-money (see Section 2.4 in the Chapter Introduction). For a strike price, $X$, underlying asset price, $S$, and time to expiry, $T$, the payoff is therefore $S$, 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

$$P_{\text{call}} = S e^{-qT} \Phi(d_1)$$

and for a put,

$$P_{\text{put}} = S e^{-qT} \Phi(-d_1)$$

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

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

and

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

The option price $P_{ij} = P(X = X_i, T = T_j)$ 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:* *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]** must contain *X*, the *i*th strike price, for *i* = 1, 2, ..., *m*.

*Constraint:* **x[i]** ≥ *z* and **x[i]** ≤ 1/*z*, where *z* = nag_real_safe_small_number, the safe range parameter, for *i* = 1, 2, ..., *m*.

6: **s** – double  
*Input*

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

*Constraint:* *s* ≥ *z* and *s* ≤ 1.0/*z*, where *z* = nag_real_safe_small_number, the safe range parameter.

7: **t[n]** – const double  
*Input*

*On entry:* **t[i]** must contain *T*, the *i*th time, in years, to expiry, for *i* = 1, 2, ..., *n*.

*Constraint:* **t[i]** ≥ *z*, where *z* = nag_real_safe_small_number, the safe range parameter, for *i* = 1, 2, ..., *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:  \( q \) \quad double

\textit{Input}

\textit{On entry:} \( q \), the annual continuous yield rate. Note that a rate of 8\% should be entered as 0.08.

\textit{Constraint:} \( q \geq 0.0 \).

11:  \( p[m \times n] \) \quad double

\textit{Output}

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

\( p[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( p[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{On exit:} \( P(i, j) \) contains \( P_{ij} \), the option price evaluated for the strike price \( x_i \) at expiry \( t_j \) for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

12:  \( \delta[m \times n] \) \quad double

\textit{Output}

\textit{Note:} the \((i, j)\)th element of the matrix is stored in

\( \delta[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( \delta[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{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:  \( \gamma[m \times n] \) \quad double

\textit{Output}

\textit{Note:} the \((i, j)\)th element of the matrix is stored in

\( \gamma[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( \gamma[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{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:  \( \nu[m \times n] \) \quad double

\textit{Output}

\textit{Note:} where \( \text{VEGA}(i, j) \) appears in this document, it refers to the array element

\( \nu[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( \nu[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{On exit:} \( \text{VEGA}(i, j) \), contains the first-order Greek measuring the sensitivity of the option price \( P_{ij} \) to change in the volatility of the underlying asset, i.e., \( \frac{\partial^2 P}{\partial \sigma^2} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

15:  \( \theta[m \times n] \) \quad double

\textit{Output}

\textit{Note:} where \( \text{THETA}(i, j) \) appears in this document, it refers to the array element

\( \theta[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( \theta[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{On exit:} \( \text{THETA}(i, j) \), contains the first-order Greek measuring the sensitivity of the option price \( P_{ij} \) to change in time, i.e., \( \frac{\partial P}{\partial t} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \), where \( b = r - q \).

16:  \( \rho[m \times n] \) \quad double

\textit{Output}

\textit{Note:} where \( \text{RHO}(i, j) \) appears in this document, it refers to the array element

\( \rho[(j - 1) \times m + i - 1] \) when \( \text{order} = \text{Nag\_ColMajor} \);

\( \rho[(i - 1) \times n + j - 1] \) when \( \text{order} = \text{Nag\_RowMajor} \).

\textit{On exit:} \( \text{RHO}(i, j) \), contains the first-order Greek measuring the sensitivity of the option price \( P_{ij} \) to change in the annual risk-free interest rate, i.e., \( \frac{\partial P}{\partial r} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).
17: \( \text{crho}[m \times n] \) – double

**Output**

Note: where CRHO\((i, j)\) appears in this document, it refers to the array element

\[
\text{crho}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{crho}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** CRHO\((i, j)\), contains the first-order Greek measuring the sensitivity of the option price \( P_j \) to change in the annual cost of carry rate, i.e., \( -\frac{\partial P_j}{\partial r} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \), where \( b = r - q \).

18: \( \text{vanna}[m \times n] \) – double

**Output**

Note: where VANNA\((i, j)\) appears in this document, it refers to the array element

\[
\text{vanna}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{vanna}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** VANNA\((i, j)\), contains the second-order Greek measuring the sensitivity of the first-order Greek \( \Delta_{ij} \) to change in the volatility of the asset price, i.e., \( -\frac{\partial \Delta_{ij}}{\partial \sigma} = -\frac{\partial P_j}{\partial \sigma} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

19: \( \text{charm}[m \times n] \) – double

**Output**

Note: where CHARM\((i, j)\) appears in this document, it refers to the array element

\[
\text{charm}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{charm}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** CHARM\((i, j)\), contains the second-order Greek measuring the sensitivity of the first-order Greek \( \Delta_{ij} \) to change in the time, i.e., \( -\frac{\partial \Delta_{ij}}{\partial t} = -\frac{\partial P_j}{\partial t} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

20: \( \text{speed}[m \times n] \) – double

**Output**

Note: where SPEED\((i, j)\) appears in this document, it refers to the array element

\[
\text{speed}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{speed}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** SPEED\((i, j)\), contains the third-order Greek measuring the sensitivity of the second-order Greek \( \Gamma_{ij} \) to change in the price of the underlying asset, i.e., \( -\frac{\partial \Gamma_{ij}}{\partial s} = -\frac{\partial P_j}{\partial s} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

21: \( \text{colour}[m \times n] \) – double

**Output**

Note: where COLOUR\((i, j)\) appears in this document, it refers to the array element

\[
\text{colour}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{colour}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** COLOUR\((i, j)\), contains the third-order Greek measuring the sensitivity of the second-order Greek \( \Gamma_{ij} \) to change in the time, i.e., \( -\frac{\partial \Gamma_{ij}}{\partial t} = -\frac{\partial P_j}{\partial t} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

22: \( \text{zomma}[m \times n] \) – double

**Output**

Note: where ZOMMA\((i, j)\) appears in this document, it refers to the array element

\[
\text{zomma}(j - 1) \times m + i - 1 \text{ when order} = \text{Nag ColMajor};
\]

\[
\text{zomma}(i - 1) \times n + j - 1 \text{ when order} = \text{Nag RowMajor}.
\]

**On exit:** ZOMMA\((i, j)\), contains the third-order Greek measuring the sensitivity of the second-order Greek \( \Gamma_{ij} \) to change in the volatility of the underlying asset, i.e., \( -\frac{\partial \Gamma_{ij}}{\partial \sigma} = -\frac{\partial P_j}{\partial \sigma} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).
vomma[m × n] – double

Output

Note: where VOMMA(i, j) appears in this document, it refers to the array element
vomma[(j - 1) × m + i - 1] when order = Nag_ColMajor;
vomma[(i - 1) × n + j - 1] when order = Nag_RowMajor.

On exit: VOMMA(i, j), contains the second-order Greek measuring the sensitivity of the first-order Greek \( \Delta_{ij} \) to change in the volatility of the underlying asset, i.e., \( -\frac{\partial \Delta_{ij}}{\partial \sigma} = -\frac{\partial^2 \pi_j}{\partial \sigma^2} \), for \( i = 1, 2, \ldots, m \) and \( j = 1, 2, \ldots, n \).

fail – NagError*

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 value \rangle had an illegal value.

**NE_INT**
On entry, \( m = \langle value \rangle \).
Constraint: \( m \geq 1 \).

On entry, \( n = \langle value \rangle \).
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, \( q = \langle value \rangle \).
Constraint: \( q \geq 0.0 \).

On entry, \( r = \langle value \rangle \).
Constraint: \( r \geq 0.0 \).

On entry, \( s = \langle value \rangle \).
Constraint: \( s \geq \langle value \rangle \) and \( s \leq \langle value \rangle \).

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

**NE_REAL_ARRAY**
On entry, \( t = \langle \text{value} \rangle \).
Constraint: \( t[i] \geq \langle \text{value} \rangle \).
On entry, \( x[(\text{value})] = (\text{value}) \).
Constraint: \( x[i] \geq (\text{value}) \) and \( x[i] \leq (\text{value}) \).

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 \textit{machine precision} (see \texttt{nag_cumul_normal (s15abc)} and \texttt{nag_erfc (s15adc)}). An accuracy close to \textit{machine precision} can generally be expected.

8 Parallelism and Performance
\texttt{nag_binary_aon_greeks (s30cdc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the \texttt{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 an asset-or-nothing put with a time to expiry of 292 days, a stock price of 70 and a strike price of 65. The risk-free interest rate is 5\% per year, there is an annual dividend return of 3\% and the volatility is 15\% per year.

10.1 Program Text
/* \texttt{nag_binary_aon_greeks (s30cdc)} 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 *charm = 0, *colour = 0, *delta = 0, *gamma = 0, *p = 0;
  double *rho = 0, *rhoq = 0, *speed = 0, *t = 0, *theta = 0;
  double *vanna = 0, *vega = 0, *vomma = 0, *x = 0, *zomma = 0;
  /* Character scalar and array declarations */
  char put[8+1];
  Nag_OrderType order;

  INIT_FAIL(fail);

  printf("nag_binary_aon_greeks (s30cdc) Example Program Results\n");
  printf("Binary (Digital): Asset-or-Nothing\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
/* nag_alloc (m*n, double) */
if (!(charm = NAG_ALLOC(m*n, double)) ||
    !(colour = 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)) ||
    !(rhoq = 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)) ||
    !(zomma = NAG_ALLOC(m*n, double)))
    
#define CHARM(I, J) charm[(J-1)*m + I-1]
#define COLOUR(I, J) colour[(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 RHOQ(I, J) rhoq[(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 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 RHOQ(I, J) rhoq[(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
!/ (vega = NAG_ALLOC(m*n, double)) ||
!/ (vomma = NAG_ALLOC(m*n, double)) ||
!/ (x = NAG_ALLOC(m, double)) ||
!/ (zomma = NAG_ALLOC(m*n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read array of strike/exercise prices, X */
for (i = 0; i < m; i++)
    ifdef _WIN32
        scanf_s("%lf ", &x[i]);
    else
        scanf("%lf ", &x[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_binary_aon_greeks (s30cdc)
* Binary option: asset-or-nothing pricing formula with Greeks */
nag_binary_aon_greeks(order, putnum, m, n, x, s, t, sigma, r, q, p,
    delta, gamma, vega, theta, rho, rhoq, vanna, charm,
    speed, colour, zomma, vomma, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_binary_aon_greeks (s30cdc).
    %s
", fail.message);
    exit_status = 1;
    goto END;
}
if (putnum == Nag_Call)
    printf("European Call :
    
    %s%8.4f
", " Spot = ", s);
else if (putnum == Nag_Put)
    printf("European Put :
    
    %s%8.4f
", " Spot = ", s);
    printf("%s8.4f\n", " Volatility = ", sigma);
    printf("%s8.4f\n", " Rate = ", r);
    printf("%s8.4f\n", " Dividend = ", q);
    printf("\n");
for (j = 1; j <= n; j++)
{
    printf("\n");
    printf(" Time to Expiry : %8.4f\n", t[j-1]);
    printf(" Strike Price Delta Gamma Vega"
    " Theta Rho Rhoq\n");
    for (i = 1; i <= m; i++)
        printf("%9.4f %9.4f %9.4f %9.4f %9.4f %9.4f %9.4f %9.4f\n", 
            x[i-1], P(i, j), DELTA(i, j), GAMMA(i, j), VEGA(i, j), 
            THETA(i, j), RHO(i, j), RHOQ(i, j));
    printf(" Vanna Charm Speed"
    " Colour Zomma Vomma\n");
    for (i = 1; i <= m; i++)
        printf("%29.4f %9.4f %9.4f %9.4f %9.4f %9.4f %9.4f\n", 
            VANNA(i, j), CHARM(i, j), SPEED(i, j), COLOUR(i, j), 
            ZOMMA(i, j), VOMMA(i, j),
```c
VOMMA(i, j));
}
END:
NAG_FREE(charm);
NAG_FREE(colour);
NAG_FREE(delta);
NAG_FREE(gamma);
NAG_FREE(p);
NAG_FREE(rho);
NAG_FREE(rhoq);
NAG_FREE(speed);
NAG_FREE(t);
NAG_FREE(theta);
NAG_FREE(vanna);
NAG_FREE(vega);
NAG_FREE(vomma);
NAG_FREE(x);
NAG_FREE(zomma);
return exit_status;
}

10.2 Program Data

nag_binary_aon_greeks (s30cdc) Example Program Data
Nag_Put : Nag_Call or Nag_Put
70.0 0.15 0.05 0.03 : s, sigma, r, q
1 1 : m, n
65.0 : X(I), I = 1,2,...m
0.8 : T(I), I = 1,2,...n

10.3 Program Results

nag_binary_aon_greeks (s30cdc) Example Program Results
Binary (Digital): Asset-or-Nothing

European Put :

<table>
<thead>
<tr>
<th>Spot</th>
<th>Volatility</th>
<th>Rate</th>
<th>Dividend</th>
</tr>
</thead>
<tbody>
<tr>
<td>70.0000</td>
<td>0.1500</td>
<td>0.0500</td>
<td>0.0300</td>
</tr>
</tbody>
</table>

Time to Expiry : 0.8000

<table>
<thead>
<tr>
<th>Strike Price</th>
<th>Delta</th>
<th>Gamma</th>
<th>Vega</th>
<th>Theta</th>
<th>Rho</th>
<th>Rhoq</th>
</tr>
</thead>
<tbody>
<tr>
<td>65.0000</td>
<td>-1.9852</td>
<td>0.1422</td>
<td>83.6424</td>
<td>-4.2761</td>
<td>-123.7497</td>
<td>-111.1728</td>
</tr>
<tr>
<td>Vanna</td>
<td>9.3479</td>
<td>-1.1351</td>
<td>0.0118</td>
<td>0.2316</td>
<td>-2.6319</td>
<td>989.9610</td>
</tr>
</tbody>
</table>
```