

# NAG Library Function Document

## nag\_bsm\_greeks (s30abc)

### 1 Purpose

nag\_bsm\_greeks (s30abc) computes the European option price given by the Black–Scholes–Merton formula together with its sensitivities (Greeks).

### 2 Specification

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

void nag_bsm_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\_bsm\_greeks (s30abc) computes the price of a European call (or put) 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, by the Black–Scholes–Merton formula (see Black and Scholes (1973) and Merton (1973)). The annual volatility,  $\sigma$ , risk-free interest rate,  $r$ , and dividend yield,  $q$ , must be supplied as input. 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  $\Phi$  denotes the cumulative Normal distribution function,

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

and

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

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

The option price  $P_{ij} = P(X = X_i, T = T_j)$  is computed for each strike price in a set  $X_i$ ,  $i = 1, 2, \dots, m$ , and for each expiry time in a set  $T_j$ ,  $j = 1, 2, \dots, n$ .

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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**  $\geq$  1.
- 4: **n** – Integer *Input*  
*On entry:* the number of times to expiry to be used.  
*Constraint:* **n**  $\geq$  1.
- 5: **x[m]** – const double *Input*  
*On entry:* **x**[ $i - 1$ ] must contain  $X_i$ , the  $i$ th strike price, for  $i = 1, 2, \dots, \mathbf{m}$ .  
*Constraint:* **x**[ $i - 1$ ]  $\geq z$  and **x**[ $i - 1$ ]  $\leq 1/z$ , where  $z = \text{nag\_real\_safe\_small\_number}$ , the safe range parameter, 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**[ $m \times n$ ] – double *Output*  
**Note:** where  $\mathbf{P}(i, j)$  appears in this document, it refers to the array element  
 $\mathbf{p}[(j - 1) \times m + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{p}[(i - 1) \times n + j - 1]$  when **order** = Nag\_RowMajor.  
*On exit:*  $\mathbf{P}(i, j)$  contains  $P_{ij}$ , the option price evaluated for the strike price  $x_i$  at expiry  $t_j$  for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ .
- 12: **delta**[ $m \times n$ ] – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix is stored in  
 $\mathbf{delta}[(j - 1) \times m + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{delta}[(i - 1) \times n + j - 1]$  when **order** = Nag\_RowMajor.  
*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**[ $m \times n$ ] – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix is stored in  
 $\mathbf{gamma}[(j - 1) \times m + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{gamma}[(i - 1) \times n + j - 1]$  when **order** = Nag\_RowMajor.  
*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: **vega**[ $m \times n$ ] – double *Output*  
**Note:** where  $\mathbf{VEGA}(i, j)$  appears in this document, it refers to the array element  
 $\mathbf{vega}[(j - 1) \times m + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{vega}[(i - 1) \times n + j - 1]$  when **order** = Nag\_RowMajor.  
*On exit:*  $\mathbf{VEGA}(i, j)$ , 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.,  $\frac{\partial P_{ij}}{\partial \sigma}$ , for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ .
- 15: **theta**[ $m \times n$ ] – double *Output*  
**Note:** where  $\mathbf{THETA}(i, j)$  appears in this document, it refers to the array element  
 $\mathbf{theta}[(j - 1) \times m + i - 1]$  when **order** = Nag\_ColMajor;  
 $\mathbf{theta}[(i - 1) \times n + j - 1]$  when **order** = Nag\_RowMajor.  
*On exit:*  $\mathbf{THETA}(i, j)$ , contains the first-order Greek measuring the sensitivity of the option price  $P_{ij}$  to change in time, i.e.,  $-\frac{\partial P_{ij}}{\partial T}$ , for  $i = 1, 2, \dots, m$  and  $j = 1, 2, \dots, n$ , where  $b = r - q$ .

16: **rho**[**m** × **n**] – double Output

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

**rho**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**rho**[(*i* – 1) × **n** + *j* – 1] when **order** = Nag\_RowMajor.

*On exit:* **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.,  $-\frac{\partial P_{ij}}{\partial r}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

17: **crho**[**m** × **n**] – double Output

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

**crho**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**crho**[(*i* – 1) × **n** + *j* – 1] when **order** = Nag\_RowMajor.

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

18: **vanna**[**m** × **n**] – double Output

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

**vanna**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**vanna**[(*i* – 1) × **n** + *j* – 1] when **order** = 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.,  $-\frac{\partial \Delta_{ij}}{\partial \sigma} = -\frac{\partial^2 P_{ij}}{\partial \sigma \partial \sigma}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

19: **charm**[**m** × **n**] – double Output

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

**charm**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**charm**[(*i* – 1) × **n** + *j* – 1] when **order** = 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.,  $-\frac{\partial \Delta_{ij}}{\partial T} = -\frac{\partial^2 P_{ij}}{\partial S \partial T}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

20: **speed**[**m** × **n**] – double Output

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

**speed**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**speed**[(*i* – 1) × **n** + *j* – 1] when **order** = 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.,  $-\frac{\partial \Gamma_{ij}}{\partial S} = -\frac{\partial^3 P_{ij}}{\partial S^3}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

21: **colour**[**m** × **n**] – double Output

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

**colour**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**colour**[(*i* – 1) × **n** + *j* – 1] when **order** = 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.,  $-\frac{\partial \Gamma_{ij}}{\partial T} = -\frac{\partial^3 P_{ij}}{\partial S \partial T}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

22: **zomma**[**m** × **n**] – double *Output*

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

**zomma**[(*j* – 1) × **m** + *i* – 1] when **order** = Nag\_ColMajor;  
**zomma**[(*i* – 1) × **n** + *j* – 1] when **order** = 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.,  $-\frac{\partial \Gamma_{ij}}{\partial \sigma} = -\frac{\partial^3 P_{ij}}{\partial \sigma^2 \partial \sigma}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

23: **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.,  $-\frac{\partial \Delta_{ij}}{\partial \sigma} = -\frac{\partial^2 P_{ij}}{\partial \sigma^2}$ , for  $i = 1, 2, \dots, \mathbf{m}$  and  $j = 1, 2, \dots, \mathbf{n}$ .

24: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument *value* had an illegal value.

### NE\_INT

On entry, **m** = *value*.  
 Constraint: **m** ≥ 1.

On entry, **n** = *value*.  
 Constraint: **n** ≥ 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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

### NE\_NO\_LICENCE

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

### NE\_REAL

On entry, **q** = *value*.  
 Constraint: **q** ≥ 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,  $\mathbf{t}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{t}[i] \geq \langle \text{value} \rangle$ .

On entry,  $\mathbf{x}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{x}[i] \geq \langle \text{value} \rangle$  and  $\mathbf{x}[i] \leq \langle \text{value} \rangle$ .

## 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_bsm_greeks` (s30abc) 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 European put with a time to expiry of 0.7 years, a stock price of 55 and a strike price of 60. The risk-free interest rate is 10% per year and the volatility is 30% per year.

### 10.1 Program Text

```

/* nag_bsm_greeks (s30abc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#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, *crho = 0, *delta = 0, *gamma = 0;
double *p = 0, *rho = 0, *speed = 0, *t = 0, *theta = 0, *vanna = 0;
double *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_bsm_greeks (s30abc) Example Program Results\n");
/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n] ");
#else
scanf("%*[\n] ");
#endif
/* Read put */
#ifdef _WIN32
scanf_s("%8s%*[\n] ", put, (unsigned)_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 sigma, r */
#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 CHARM(I, J) charm[(J-1)*m + I-1]
#define COLOUR(I, J) colour[(J-1)*m + I-1]
#define CRHO(I, J) crho[(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 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 CRHO(I, J) crho[(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 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
    if (!(charm = NAG_ALLOC(m * n, double)) ||
        !(colour = NAG_ALLOC(m * n, double)) ||
        !(crho = 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)) ||
        !(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)) ||
        !(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
    /* Read array of times to expiry */
    for (i = 0; i < n; i++)
#ifdef _WIN32
        scanf_s("%*[^\\n] ");
#else
        scanf("%*[^\\n] ");
#endif
    nag_bsm_greeks(order, putnum, m, n, x, s, t, sigma, r, q, p,
                  delta, gamma, vega, theta, rho, crho, vanna, charm,
                  speed, colour, zomma, vomma, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_bsm_greeks (s30abc).\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");
    for (j = 1; j <= n; j++) {
        printf(" Time to Expiry : %8.4f\n", t[j - 1]);
        printf(" Strike Price Delta Gamma Vega "
              "Theta Rho CRho\n");
    }

```



```

    for (i = 1; i <= m; i++)
        printf("%8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n",
            x[i - 1], P(i, j), DELTA(i, j), GAMMA(i, j), VEGA(i, j),
            THETA(i, j), RHO(i, j), CRHO(i, j));
    printf("
        Vanna    Charm    Speed
    "
        "Colour    Zomma    Vomma\n");
    for (i = 1; i <= m; i++)
        printf("%26.4f %8.4f %8.4f %8.4f %8.4f %8.4f\n",
            VANNA(i, j), CHARM(i, j), SPEED(i, j),
            COLOUR(i, j), ZOMMA(i, j), VOMMA(i, j));
}

END:
    NAG_FREE(charm);
    NAG_FREE(colour);
    NAG_FREE(crho);
    NAG_FREE(delta);
    NAG_FREE(gamma);
    NAG_FREE(p);
    NAG_FREE(rho);
    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_bsm_greeks (s30abc) Example Program Data
Nag_Put          : Nag_Call or Nag_Put
55.0 0.3 0.1 0.0 : s, sigma, r, q
1 1              : m, n
60.0             : x(i), i = 1,2,...m
0.7              : t(i), i = 1,2,...n

```

## 10.3 Program Results

```

nag_bsm_greeks (s30abc) Example Program Results
European Put :

```

```

Spot          = 55.0000
Volatility    = 0.3000
Rate          = 0.1000
Dividend      = 0.0000

```

```

Time to Expiry : 0.7000
Strike Price   Delta   Gamma   Vega   Theta   Rho   CRho
60.0000  6.0245 -0.4770  0.0289 18.3273 -0.7014 -22.5811 -18.3639
          Vanna   Charm   Speed   Colour   Zomma   Vomma
          0.2566 -0.2137 -0.0006  0.0215 -0.0972 -0.6816

```

---