NAG Library Function Document

nag_heston_greeks (s30nbc)

1 Purpose

nag_heston_greeks (s30nbc) computes the European option price given by Heston's stochastic volatility model together with its sensitivities (Greeks).

2 Specification

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

```
void nag_heston_greeks (Nag_OrderType order, Nag_CallPut option, Integer m,
     Integer n, const double x[], double s, const double t[], double sigmav,
     double kappa, double corr, double var0, double eta, double grisk,
     double r, double q, double p[], double delta[], double gamma[],
     double vega[], double theta[], double rho[], double vanna[],
     double charm[], double speed[], double zomma[], double vomma[],
     NagError *fail)
```

3 Description

nag_heston_greeks (s30nbc) computes the price and sensitivities of a European option using Heston's stochastic volatility model. The return on the asset price, S, is

$$\frac{dS}{S} = (r-q)dt + \sqrt{v_t}dW_t^{(1)}$$

and the instantaneous variance, v_t, is defined by a mean-reverting square root stochastic process,

$$dv_t = \kappa(\eta - v_t)dt + \sigma_v\sqrt{v_t}dW_t^{(2)},$$

where r is the risk free annual interest rate; q is the annual dividend rate; v_t is the variance of the asset price; σ_v is the volatility of the volatility, $\sqrt{v_t}$; κ is the mean reversion rate; η is the long term variance. $dW_t^{(i)}$, for $i = 1, 2$, denotes two correlated standard Brownian motions with

$$\mathbb{C}\mathrm{ov}\left[dW_t^{(1)}, dW_t^{(2)}\right] = \rho dt.$$

The option price is computed by evaluating the integral transform given by Lewis (2000) using the form of the characteristic function discussed by Albrecher *et al.* (2007), see also Kilin (2006).

$$P_{\text{call}} = Se^{-qT} - Xe^{-rT}\frac{1}{\pi}\mathrm{Re}\left[\int_{0+i/2}^{\infty+i/2} e^{-ik\bar{X}}\frac{\hat{H}(k,v,T)}{k^2-ik}\mathrm{d}k\right], \tag{1}$$

where $\bar{X} = \ln(S/X) + (r-q)T$ and

$$\hat{H}(k,v,T) = \exp\left(\frac{2\kappa\eta}{\sigma_v^2}\left[t\mathrm{gendgroup} - \ln\left(\frac{1-\mathrm{he}^{-\xi t}}{1-\mathrm{h}}\right)\right] + \mathrm{v_t g}\left[\frac{1-\mathrm{e}^{-\xi t}}{1-\mathrm{he}^{-\xi t}}\right]\right),$$

$$g = \frac{1}{2}(b-\xi), \quad h = \frac{b-\xi}{b+\xi}, \quad t = \sigma_v^2 T/2,$$

$$\xi = \left[b^2 + 4\frac{k^2-ik}{\sigma_v^2}\right]^{\frac{1}{2}},$$

$$b = \frac{2}{\sigma_v^2}\left[(1-\gamma+ik)\rho\sigma_v + \sqrt{\kappa^2 - \gamma(1-\gamma)\sigma_v^2}\right]$$

with $t = \sigma_v^2 T/2$. Here γ is the risk aversion parameter of the representative agent with $0 \le \gamma \le 1$ and $\gamma(1-\gamma)\sigma_v^2 \le \kappa^2$. The value $\gamma = 1$ corresponds to $\lambda = 0$, where λ is the market price of risk in Heston (1993) (see Lewis (2000) and Rouah and Vainberg (2007)).

The price of a put option is obtained by put-call parity.

$$P_{\text{put}} = P_{\text{call}} + Xe^{-rT} - Se^{-qT}.$$

Writing the expression for the price of a call option as

$$P_{\text{call}} = Se^{-qT} - Xe^{-rT}\frac{1}{\pi}\text{Re}\left[\int_{0+i/2}^{\infty+i/2} I(k,r,S,T,v)\text{d}k\right]$$

then the sensitivities or Greeks can be obtained in the following manner,

Delta

$$\frac{\partial P_{\text{call}}}{\partial S} = e^{-qT} + \frac{Xe^{-rT}}{S}\frac{1}{\pi}\text{Re}\left[\int_{0+i/2}^{\infty+i/2}(ik)I(k,r,S,T,v)\text{d}k\right],$$

Vega

$$\frac{\partial P}{\partial v} = -Xe^{-rT}\frac{1}{\pi}\text{Re}\left[\int_{0-i/2}^{0+i/2} f_2 I(k,r,j,S,T,v)\text{d}k\right], \quad \text{where } f_2 = g\left[\frac{1-e^{-\xi t}}{1-he^{-\xi t}}\right],$$

Rho

$$\frac{\partial P_{\text{call}}}{\partial r} = TXe^{-rT}\frac{1}{\pi}\text{Re}\left[\int_{0+i/2}^{\infty+i/2}(1+ik)I(k,r,S,T,v)\text{d}k\right].$$

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

Albrecher H, Mayer P, Schoutens W and Tistaert J (2007) The little Heston trap *Wilmott Magazine* **January 2007** 83–92

Heston S (1993) A closed-form solution for options with stochastic volatility with applications to bond and currency options *Review of Financial Studies* **6** 327–343

Kilin F (2006) Accelerating the calibration of stochastic volatility models *MPRA Paper No. 2975* http://mpra.ub.uni-muenchen.de/2975/

Lewis A L (2000) Option valuation under stochastic volatility *Finance Press, USA*

Rouah F D and Vainberg G (2007) *Option Pricing Models and Volatility using Excel-VBA* John Wiley and Sons, Inc

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 ith 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: **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: **sigmav** – double *Input*

On entry: the volatility, σ_v, of the volatility process, $\sqrt{v_t}$. Note that a rate of 20% should be entered as 0.2.

Constraint: **sigmav** > 0.0.

9: **kappa** – double *Input*

On entry: κ, the long term mean reversion rate of the volatility.

Constraint: **kappa** > 0.0.

10: **corr** – double *Input*

On entry: the correlation between the two standard Brownian motions for the asset price and the volatility.

Constraint: $-1.0 \leq$ **corr** ≤ 1.0.

11: **var0** – double *Input*

On entry: the initial value of the variance, v_t, of the asset price.

Constraint: **var0** ≥ 0.0.

12: **eta** – double *Input*

On entry: η, the long term mean of the variance of the asset price.

Constraint: **eta** > 0.0.

13: **grisk** – double *Input*

On entry: the risk aversion parameter, γ, of the representative agent.

Constraint: $0.0 \leq$ **grisk** ≤ 1.0 and **grisk** $\times (1 -$ **grisk**$) \times$ **sigmav** $\times$ **sigmav** $\leq$ **kappa** $\times$ **kappa**.

14: **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.

15: **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** ≥ 0.0.

16: **p**[**m** $\times$ **n**] – double *Output*

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

p$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
p$[(i-1) \times$ **n** $+ j - 1]$ when **order** = Nag_RowMajor.

On exit: **P**(i,j) contains P_{ij}, the option price evaluated for the strike price $\mathbf{x}_i$ at expiry $\mathbf{t}_j$ for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

17: **delta**[**m** $\times$ **n**] – double *Output*

Note: the (i,j)th element of the matrix is stored in

delta$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
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.

18: **gamma**[**m** $\times$ **n**] – double *Output*

Note: the (i,j)th element of the matrix is stored in

gamma$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
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.

19: **vega**[**m** $\times$ **n**] – double *Output*

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

vega$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
vega$[(i-1) \times$ **n** $+ j - 1]$ when **order** = Nag_RowMajor.

On exit: **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 $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

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

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

theta$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
theta$[(i-1) \times$ **n** $+ j - 1]$ when **order** = Nag_RowMajor.

On exit: **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 $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**, where $b = r - q$.

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

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

rho$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
rho$[(i-1) \times$ **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 $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

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

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

vanna$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
vanna$[(i-1) \times$ **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 Δ_{ij} to change in the volatility of the asset price, i.e., $-\frac{\partial \Delta_{ij}}{\partial T} = -\frac{\partial^2 P_{ij}}{\partial S \partial \sigma}$, for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

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

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

charm$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
charm$[(i-1) \times$ **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 Δ_{ij} to change in the time, i.e., $-\frac{\partial \Delta_{ij}}{\partial T} = -\frac{\partial^2 P_{ij}}{\partial S \partial T}$, for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

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

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

speed$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
speed$[(i-1) \times$ **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 Γ_{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 $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

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

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

zomma$[(j-1) \times$ **m** $+ i - 1]$ when **order** = Nag_ColMajor;
zomma$[(i-1) \times$ **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 Γ_{ij} to change in the volatility of the underlying asset, i.e., $-\frac{\partial \Gamma_{ij}}{\partial \sigma} = -\frac{\partial^3 P_{ij}}{\partial S^2 \partial \sigma}$, for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **n**.

26: **vomma**[**m** $\times$ **n**] – double *Output*

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

vomma$[(j-1) \times$ **m** $+ i - 1]$ when **order** $=$ Nag_ColMajor;
vomma$[(i-1) \times$ **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 Δ_{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 $i = 1, 2, \dots,$ **m** and $j = 1, 2, \dots,$ **n**.

27: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ACCURACY

Solution cannot be computed accurately. Check values of input arguments.

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 ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

Quadrature has not converged to the required accuracy. However, the result should be a reasonable approximation.

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 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, **corr** $=$ ⟨*value*⟩.
Constraint: $|$**corr**$| \leq 1.0$.

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

On entry, **grisk** $= \langle value \rangle$, **sigmav** $= \langle value \rangle$ and **kappa** $= \langle value \rangle$.
Constraint: $0.0 \leq$ **grisk** ≤ 1.0 and **grisk** $\times (1.0 -$ **grisk**$) \times$ **sigmav**$^2 \leq$ **kappa**^{2}.

On entry, **kappa** $= \langle value \rangle$.
Constraint: **kappa** > 0.0.

On entry, **q** $= \langle value \rangle$.
Constraint: **q** ≥ 0.0.

On entry, **r** $= \langle value \rangle$.
Constraint: **r** ≥ 0.0.

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

On entry, **sigmav** $= \langle value \rangle$.
Constraint: **sigmav** > 0.0.

On entry, **var0** $= \langle value \rangle$.
Constraint: **var0** ≥ 0.0.

NE_REAL_ARRAY

On entry, **t**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **t**$[i-1] \geq \langle value \rangle$.

On entry, **x**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **x**$[i-1] \geq \langle value \rangle$ and **x**$[i-1] \leq \langle value \rangle$.

7 Accuracy

The accuracy of the output is determined by the accuracy of the numerical quadrature used to evaluate the integral in (1). An adaptive method is used which evaluates the integral to within a tolerance of $\max\left(10^{-8}, 10^{-10} \times |I|\right)$, where $|I|$ is the absolute value of the integral.

8 Parallelism and Performance

nag_heston_greeks (s30nbc) 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 and sensitivities of a European call using Heston's stochastic volatility model. The time to expiry is 1 year, the stock price is 100 and the strike price is 100. The risk-free interest rate is 2.5% per year, the volatility of the variance, σ_v, is 57.51% per year, the mean reversion parameter, κ, is 1.5768, the long term mean of the variance, η, is 0.0398 and the correlation between the volatility process and the stock price process, ρ, is -0.5711. The risk aversion parameter, γ, is 1.0 and the initial value of the variance, **var0**, is 0.0175.

10.1 Program Text

```
/* nag_heston_greeks (s30nbc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

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

int main(void)
{
#ifdef NAG_COLUMN_MAJOR
#define K(I, J)     (J-1)*pdp + I-1
#else
#define K(I, J)     (I-1)*pdp + J-1
#endif

  /* Scalars */
  Integer       exit_status = 0;
  double        corr, eta, grisk, kappa, q, r, s, sigmav, var0;
  Integer       i, j, pdp, m, n;
  /* Arrays */
  double        *charm = 0, *delta = 0, *gamma = 0, *p = 0, *rho = 0,
                *speed = 0, *t = 0, *theta = 0, *vanna = 0, *vega = 0,
                *vomma = 0, *x = 0, *zomma = 0;
  char          put[8+1];
  /* Nag types */
  Nag_OrderType order;
  Nag_CallPut   putnum;
  NagError      fail;

  INIT_FAIL(fail);

  printf("nag_heston_greeks (s30nbc) 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, _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, r, q */
#ifdef _WIN32
  scanf_s("%lf%lf%lf%*[^\n] ", &s, &r, &q);
#else
  scanf("%lf%lf%lf%*[^\n] ", &s, &r, &q);
#endif
  /* Read kappa,eta,var0,sigmav,corr,grisk */
#ifdef _WIN32
  scanf_s("%lf%lf%lf%*[^\n]", &kappa, &eta, &var0);
#else
  scanf("%lf%lf%lf%*[^\n]", &kappa, &eta, &var0);
#endif
#ifdef _WIN32
  scanf_s("%lf%lf%lf%*[^\n]", &sigmav, &corr, &grisk);
#else
  scanf("%lf%lf%lf%*[^\n]", &sigmav, &corr, &grisk);
```

```
#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
  if (!(charm = 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;
    }
#ifdef NAG_COLUMN_MAJOR
  order = Nag_ColMajor;
  pdp = m;
#else
  order = Nag_RowMajor;
  pdp = n;
#endif

  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
  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_heston_greeks (s30nbc).
     Heston's model option pricing formula with Greeks
   */
  nag_heston_greeks(order, putnum, m, n, x, s, t, sigmav, kappa, corr, var0,
                    eta, grisk, r, q, p, delta, gamma, vega, theta, rho, vanna,
                    charm, speed, zomma, vomma, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_heston_greeks (s30nbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
```

```
  printf("\nHeston’s Stochastic volatility Model\n");
  switch (putnum)
    {
    case Nag_Call:
      printf("European Call :\n\n");
      break;
    case Nag_Put:
      printf("European Put :\n\n");
    }
  printf("  Spot                   = %10.4f\n", s);
  printf("  Volatility of vol      = %10.4f\n", sigmav);
  printf("  Mean reversion         = %10.4f\n", kappa);
  printf("  Correlation            = %10.4f\n", corr);
  printf("  Variance               = %10.4f\n", var0);
  printf("  Mean of variance       = %10.4f\n", eta);
  printf("  Risk aversion          = %10.4f\n", grisk);
  printf("  Rate                   = %10.4f\n", r);
  printf("  Dividend               = %10.4f\n\n", q);

  for (j = 1; j <= n; j++)
    {
      printf("Time to Expiry : %8.4f\n", t[j-1]);

      printf("%10s%11s%11s%11s%11s%11s%11s\n",
             "Strike", "Price", "Delta", "Gamma", "Vega", "Theta", "Rho");
      for (i = 1; i <= m; i++)
        printf("%10.4f %10.4f %10.4f %10.4f %10.4f %10.4f %10.4f\n", x[i-1],
               p[K(i, j)], delta[K(i, j)], gamma[K(i, j)], vega[K(i, j)],
               theta[K(i, j)], rho[K(i, j)]);

      printf("%32s%11s%11s%11s%11s\n",
             "Vanna", "Charm", "Speed", "Zomma", "Vomma");
      for (i = 1; i <= m; i++)
        printf("%21s %10.4f %10.4f %10.4f %10.4f %10.4f\n", "", vanna[K(i, j)],
               charm[K(i, j)], speed[K(i, j)], zomma[K(i, j)], vomma[K(i, j)]);
    }
 END:
  NAG_FREE(charm);
  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_heston_greeks (s30nbc) Example Program Data
 Nag_Call                    : CallPut option
 100.0     0.025  0.0        : s, r, q
   1.5768  0.0398 0.0175     : kappa,  eta,  var0
   0.5751 -0.5711 1.0        : sigmav, corr, grisk
   1       1                 : m, n
 100.0                       : x[i], i = 0,...,n-1
   1.0                       : t[i], i = 0,...,m-1
```

10.3 Program Results

```
nag_heston_greeks (s30nbc) Example Program Results

Heston's Stochastic volatility Model
European Call :

  Spot                   =   100.0000
  Volatility of vol      =     0.5751
  Mean reversion         =     1.5768
  Correlation            =    -0.5711
  Variance               =     0.0175
  Mean of variance       =     0.0398
  Risk aversion          =     1.0000
  Rate                   =     0.0250
  Dividend               =     0.0000

Time to Expiry :   1.0000
    Strike      Price      Delta      Gamma       Vega      Theta        Rho
  100.0000     7.2743     0.6945     0.0251    52.5461    -4.9969    62.1735
                           Vanna      Charm      Speed      Zomma      Vomma
                         -0.5643    -0.0321    -0.0023    -0.1976  -321.0780
```