NAG Library Function Document

nag_heston_term (s30ncc)

1 Purpose

nag_heston_term (s30ncc) computes the European option price given by Heston's stochastic volatility model with term structure.

2 Specification

```
#include <nag.h>
#include <nags.h>
void nag_heston_term (Nag_CallPut option, Integer m, Integer numts,
     const double x[], double fwd, double disc, const double ts[], double t,
     const double alpha[], const double lambda[], const double corr[],
     const double sigmat[], double var0, double p[], NagError *fail)
```

3 Description

nag_heston_term (s30ncc) computes the price of a European option for Heston's stochastic volatility model with time-dependent parameters which are piecewise constant. Starting from the stochastic volatility model given by the Stochastic Differential Equation (SDE) system defined by Heston (1993), a scaling of the variance process is introduced, together with a normalization, setting the long run variance, η, equal to 1. This leads to

$$\frac{\mathrm{d}S_t}{S_t} = \mu_t \mathrm{d}t + \sigma_t \sqrt{\nu_t} \mathrm{d}W_t^{(1)}, \tag{1}$$

$$\mathrm{d}\nu_t = \lambda_t (1 - \nu_t)\mathrm{d}t + \alpha_t \sqrt{\nu_t} \mathrm{d}W_t^{(2)}, \tag{2}$$

$$\mathrm{Cov}\left[\mathrm{d}W_t^{(1)}, \mathrm{d}W_t^{(2)}\right] = \rho_t \mathrm{d}t, \tag{3}$$

where $\mu_t = r_t - q_t$ is the drift term representing the contribution of interest rates, r_t, and dividends, q_t, while σ_t is the scaling parameter, ν_t is the scaled variance, λ_t is the mean reversion rate and α_t is the volatility of the scaled volatility, $\sqrt{\nu_t}$. Then, $W_t^{(i)}$, for $i = 1, 2$, are two standard Brownian motions with correlation parameter ρ_t. Without loss of generality, the drift term, μ_t, is eliminated by modelling the forward price, F_t, directly, instead of the spot price, S_t, with

$$F_t = S_0 \exp\left(\int_0^t \mu_s \mathrm{d}s\right). \tag{4}$$

If required, the spot can be expressed as, $S_0 = DF_t$, where D is the discount factor.

The option price is computed by dividing the time to expiry, T, into n_s subintervals $[t_0, t_1], \ldots, [t_{i-1}, t_i], \ldots, [t_{n_s-1}, T]$ and applying the method of characteristic functions to each subinterval, with appropriate initial conditions. Thus, a pair of ordinary differential equations (one of which is a Riccati equation) is solved on each subinterval as outlined in Elices (2008) and Mikhailov and Nögel (2003). Reversing time by taking $\tau = T - t$, the characteristic function solution for the first time subinterval, starting at $\tau = 0$, is given by Heston (1993), while the solution on each following subinterval uses the solution of the preceding subinterval as initial condition to compute the value of the characteristic function.

In the case of a 'flat' term structure, i.e., the parameters are constant over the time of the option, T, the form of the SDE system given by Heston (1993)

$$\frac{\mathrm{d}S_t}{S_t} = \mu_t \mathrm{d}t + \sqrt{V_t}\mathrm{d}W_t^{(1)}, \tag{5}$$

$$\mathrm{d}V_t = \kappa(\eta - V_t)\mathrm{d}t + \sigma_v\sqrt{V_t}\mathrm{d}W_t^{(2)}, \tag{6}$$

can be recovered by setting $V_0 = \eta = \sigma_t^2$, $\kappa = \lambda_t$, $\sigma_v = \sigma_t\alpha_t$.

Conversely, given the Heston form of the SDE pair, to get the term structure form set $\sigma = \sqrt{\eta}$, $\alpha_t = \sigma_v/\sqrt{\eta}$, $\lambda_t = \kappa$.

4 References

Bain A (2011) *Private communication*

Elices A (2008) Models with time-dependent parameters using transform methods: application to Heston's model *arXiv:0708.2020v2*

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

Mikhailov S and Nögel U (2003) Heston's Stochastic Volatility Model Implementation, Calibration and Some Extensions *Wilmott Magazine July/August* 74–79

5 Arguments

1: **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.

2: **m** – Integer *Input*

On entry: m, the number of strike prices to be used.

Constraint: **m** ≥ 1.

3: **numts** – Integer *Input*

On entry: n_s, the number of subintervals into which the time to expiry, T, is divided.

Constraint: **numts** ≥ 1.

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

On entry: **x**$[i-1]$ contains the ith strike price, for $i = 1, 2, \ldots, m$.

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

5: **fwd** – double *Input*

On entry: the forward price of the asset.

Constraint: **fwd** $\geq z$ and **fwd** $\leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter.

6: **disc** – double *Input*

On entry: the discount factor, where the current price of the underlying asset, S_0, is given as $S_0 = \textbf{disc} \times \textbf{fwd}$.

Constraint: **disc** $\geq z$ and **disc** $\leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter.

7: **ts**[**numts**] – const double *Input*

On entry: **ts**$[i-1]$ must contain the length of the time intervals for which the corresponding element of **alpha**, **lambda**, **corr** and **sigmat** apply. These should be ordered as they occur in time i.e., $\Delta t_i = t_i - t_{i-1}$.

Constraint: **ts**$[i-1] \geq z$ and **ts**$[i-1] \leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots,$ **numts**.

8: **t** – double *Input*

On entry: **t** contains the time to expiry. If $T > \sum \Delta t_i$ then the parameters associated with the last time interval are extended to the expiry time. If $T < \sum \Delta t_i$ then the parameters specified are used up until the expiry time. The rest are ignored.

Constraint: **t** $\geq z$, where $z =$ nag_real_safe_small_number, the safe range parameter.

9: **alpha**[**numts**] – const double *Input*

On entry: **alpha**$[i-1]$ must contain the value of α_t, the value of the volatility of the scaled volatility, $\sqrt{\nu}$, over time subinterval Δt_i.

Constraint: **alpha**$[i-1] \geq z$ and **alpha**$[i-1] \leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots,$ **numts**.

10: **lambda**[**numts**] – const double *Input*

On entry: **lambda**$[i-1]$ must contain the value, λ_t, of the mean reversion parameter over the time subinterval Δt_i.

Constraint: **lambda**$[i-1] \geq z$ and **lambda**$[i-1] \leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots,$ **numts**.

11: **corr**[**numts**] – const double *Input*

On entry: **corr**$[i-1]$ must contain the value, ρ_t, of the correlation parameter over the time subinterval Δt_i.

Constraint: $-1.0 \leq$ **corr**$[i-1] \leq 1.0$, for $i = 1, 2, \ldots,$ **numts**.

12: **sigmat**[**numts**] – const double *Input*

On entry: **sigmat**$[i-1]$ must contain the value, σ_t, of the variance scale factor over the time subinterval Δt_i.

Constraint: **sigmat**$[i-1] \geq z$ and **sigmat**$[i-1] \leq 1/z$, where $z =$ nag_real_safe_small_number, the safe range parameter, for $i = 1, 2, \ldots,$ **numts**.

13: **var0** – double *Input*

On entry: ν_0, the initial scaled variance.

Constraint: **var0** ≥ 0.0.

14: **p**[**m**] – double *Output*

On exit: **p**$[i-1]$ contains the computed option price at the expiry time, T, corresponding to strike **x**$[i-1]$ for the specified term structure, for $i = 1, 2, \ldots,$ **m**.

15: **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 specified accuracy. However, the result should be a reasonable approximation.

NE_INT

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

On entry, **numts** = ⟨*value*⟩.
Constraint: **numts** ≥ 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, **disc** = ⟨*value*⟩.
Constraint: ⟨*value*⟩ $\leq$ **disc** $\leq$ ⟨*value*⟩.

On entry, **fwd** = ⟨*value*⟩.
Constraint: ⟨*value*⟩ $\leq$ **fwd** $\leq$ ⟨*value*⟩.

On entry, **t** = ⟨*value*⟩.
Constraint: **t** $\geq$ ⟨*value*⟩.

On entry, **var0** = ⟨*value*⟩.
Constraint: **var0** > 0.0.

NE_REAL_ARRAY

On entry, **alpha**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ $\leq$ **alpha**$[i-1] \leq$ ⟨*value*⟩.

On entry, **corr**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: $|$**corr**$[i-1]| \leq 1.0$.

On entry, **lambda**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ ≤ **lambda**[$i-1$] ≤ ⟨*value*⟩.

On entry, **sigmat**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ ≤ **sigmat**[$i-1$] ≤ ⟨*value*⟩.

On entry, **ts**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ ≤ **ts**[$i-1$] ≤ ⟨*value*⟩.

On entry, **x**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: ⟨*value*⟩ ≤ **x**[$i-1$] ≤ ⟨*value*⟩.

7 Accuracy

The solution is obtained by integrating the pair of ordinary differential equations over each subinterval in time. The accuracy is controlled by a relative tolerance over each time subinterval, which is set to 10^{-8}. Over a number of subintervals in time the error may accumulate and so the overall error in the computation may be greater than this. A threshold of 10^{-10} is used and solutions smaller than this are not accurately evaluated.

8 Parallelism and Performance

nag_heston_term (s30ncc) 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 call using Heston's stochastic volatility model with a term structure of interest rates.

10.1 Program Text

```
/* nag_heston_term (s30ncc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <nag.h>
#include <nag_stdlib.h>
#include <nags.h>
#include <nagf16.h>

int main(void)
{
  /* Scalars */
  Integer       exit_status = 0;
  double        disc, fwd, t, var0;
  Integer       i, j, m, numts;
  /* Arrays */
  double        *alpha = 0, *corr = 0, *lambda = 0, *p = 0, *sigmat = 0,
                *ts = 0, *x = 0;
  char          option_str[8+1];
  /* Nag Types */
  Nag_CallPut   option;
```

```
  NagError      fail;

  printf("nag_heston_term (s30ncc) Example Program Results\n");

  /* Initialise fail */
  INIT_FAIL(fail);
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%8s%*[^\n]", option_str, _countof(option_str));
#else
  scanf("%8s%*[^\n]", option_str);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  option = (Nag_CallPut) nag_enum_name_to_value(option_str);

#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %"NAG_IFMT"%*[^\n] ", &m, &numts);
#else
  scanf("%"NAG_IFMT" %"NAG_IFMT"%*[^\n] ", &m, &numts);
#endif
  if (!(p      = NAG_ALLOC(m,     double))||
      !(ts     = NAG_ALLOC(numts, double))||
      !(x      = NAG_ALLOC(m,     double))||
      !(alpha  = NAG_ALLOC(numts, double))||
      !(corr   = NAG_ALLOC(numts, double))||
      !(lambda = NAG_ALLOC(numts, double))||
      !(sigmat = NAG_ALLOC(numts, double))
      )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#ifdef _WIN32
  scanf_s("%lf %lf %lf%*[^\n] ", &fwd, &disc, &var0);
#else
  scanf("%lf %lf %lf%*[^\n] ", &fwd, &disc, &var0);
#endif
  for (j=0; j<m; j++) {
#ifdef _WIN32
    scanf_s("%lf", &x[j]);
#else
    scanf("%lf", &x[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (j=0; j<numts; j++) {
#ifdef _WIN32
    scanf_s("%lf", &ts[j]);
#else
    scanf("%lf", &ts[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
```

```
  for (j=0; j<numts; j++) {
#ifdef _WIN32
    scanf_s("%lf", &alpha[j]);
#else
    scanf("%lf", &alpha[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (j=0; j<numts; j++) {
#ifdef _WIN32
    scanf_s("%lf", &corr[j]);
#else
    scanf("%lf", &corr[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (j=0; j<numts; j++) {
#ifdef _WIN32
    scanf_s("%lf", &lambda[j]);
#else
    scanf("%lf", &lambda[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  for (j=0; j<numts; j++) {
#ifdef _WIN32
    scanf_s("%lf", &sigmat[j]);
#else
    scanf("%lf", &sigmat[j]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

 /* nag_dsum (f16elc).
  * Sum elements of a vector of doubles
  */
  t = nag_dsum(numts, ts, 1, &fail);

  /* nag_heston_term (s30nc).
   * Heston's Stochastic volatility Model with Term Structure.
   */
  nag_heston_term(option, m, numts, x, fwd, disc, ts, t, alpha, lambda, corr,
                  sigmat, var0, p, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_heston_term (s30ncc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nHeston's Stochastic volatility Model with Term Structure\n");
  switch (option) {
  case Nag_Call:
    printf("European Call :\n");
    break;
  case Nag_Put:
```

```
    printf("European Put :\n");
  }

  printf("  Forward               = %9.4f\n", fwd);
  printf("  Discount Factor       = %9.4f\n", disc);
  printf("  Variance              = %9.4f\n", var0);
  printf("\n   ts        alpha     lambda    corr      sigmat\n");
  for ( i=0; i<numts; i++) {
    printf("%9.4f %9.4f %9.4f %9.4f %9.4f\n", ts[i], alpha[i], lambda[i],
           corr[i], sigmat[i]);
  }
  printf("\n   Strike    Expiry       Option Price\n");
  for ( i=0; i<m; i++)
    printf("%9.4f %9.4f %12.4f\n", x[i], t, p[i]);
END:
  NAG_FREE(alpha);
  NAG_FREE(corr);
  NAG_FREE(lambda);
  NAG_FREE(p);
  NAG_FREE(sigmat);
  NAG_FREE(ts);
  NAG_FREE(x);
  return exit_status;
}
```

10.2 Program Data

```
nag_heston_term (s30ncc) Example Program Data
 Nag_Call                  : CallPut option
   1     2                 : m, numts
 100.0   1.0  1.0          : fwd, disc, var0
 100.0                     : x[],      m     values
   0.35  0.65              : ts[],     numts values
   2.25  1.5               : alpha[],  numts values
  -0.05  0.1               : corr[],   numts values
   2.0   1.5               : lambda[], numts values
   0.04  0.13              : sigmat[], numts values
```

10.3 Program Results

```
nag_heston_term (s30ncc) Example Program Results

Heston's Stochastic volatility Model with Term Structure
European Call :
  Forward               =  100.0000
  Discount Factor       =    1.0000
  Variance              =    1.0000

   ts        alpha     lambda    corr      sigmat
   0.3500    2.2500    2.0000   -0.0500    0.0400
   0.6500    1.5000    1.5000    0.1000    0.1300

   Strike    Expiry       Option Price
 100.0000    1.0000       4.0074
```