NAG Library Routine Document

D03NCF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

D03NCF solves the Black–Scholes equation for financial option pricing using a finite difference scheme.

2 Specification

```
SUBROUTINE D03NCF (KOPT, X, MESH, NS, S, NT, T, TDPAR, R, Q, SIGMA,          &
                   ALPHA, NTKEEP, F, THETA, DELTA, GAMMA, LAMBDA, RHO,       &
                   LDF, WORK, IWORK, IFAIL)

INTEGER            KOPT, NS, NT, NTKEEP, LDF, IWORK(NS), IFAIL
REAL (KIND=nag_wp) X, S(NS), T(NT), R(*), Q(*), SIGMA(*), ALPHA,             &
                   F(LDF,NTKEEP), THETA(LDF,NTKEEP),                         &
                   DELTA(LDF,NTKEEP), GAMMA(LDF,NTKEEP),                     &
                   LAMBDA(LDF,NTKEEP), RHO(LDF,NTKEEP), WORK(4*NS)
LOGICAL            TDPAR(3)
CHARACTER(1)       MESH
```

3 Description

D03NCF solves the Black–Scholes equation (see Hull (1989) and Wilmott *et al.* (1995))

$$\frac{\partial f}{\partial t} + (r-q)S\frac{\partial f}{\partial S} + \frac{\sigma^2 S^2}{2}\frac{\partial^2 f}{\partial S^2} = rf \tag{1}$$

$$S_{\min} < S < S_{\max}, \quad t_{\min} < t < t_{\max}, \tag{2}$$

for the value f of a European or American, put or call stock option, with exercise price X. In equation (1) t is time, S is the stock price, r is the risk free interest rate, q is the continuous dividend, and σ is the stock volatility. According to the values in the array TDPAR, the arguments r, q and σ may each be either constant or functions of time. The routine also returns values of various Greeks.

D03NCF uses a finite difference method with a choice of time-stepping schemes. The method is explicit for ALPHA $= 0.0$ and implicit for nonzero values of ALPHA. Second order time accuracy can be obtained by setting ALPHA $= 0.5$. According to the value of the argument MESH the finite difference mesh may be either uniform, or user-defined in both S and t directions.

4 References

Hull J (1989) *Options, Futures and Other Derivative Securities* Prentice–Hall

Wilmott P, Howison S and Dewynne J (1995) *The Mathematics of Financial Derivatives* Cambridge University Press

5 Arguments

1: KOPT – INTEGER *Input*

On entry: specifies the kind of option to be valued.

KOPT $= 1$
A European call option.

KOPT = 2
An American call option.

KOPT = 3
A European put option.

KOPT = 4
An American put option.

Constraint: KOPT = 1, 2, 3 or 4.

2: X – REAL (KIND=nag_wp) *Input*

On entry: the exercise price X.

3: MESH – CHARACTER(1) *Input*

On entry: indicates the type of finite difference mesh to be used:

MESH = 'U'
Uniform mesh.

MESH = 'C'
Custom mesh supplied by you.

Constraint: MESH = 'U' or 'C'.

4: NS – INTEGER *Input*

On entry: the number of stock prices to be used in the finite difference mesh.

Constraint: NS ≥ 2.

5: S(NS) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if MESH = 'C', S(i) must contain the ith stock price in the mesh, for $i = 1, 2, \ldots, \text{NS}$. These values should be in increasing order, with $\text{S}(1) = S_{\min}$ and $\text{S}(\text{NS}) = S_{\max}$.

If MESH = 'U', S(1) must be set to $S_{\min}$ and S(NS) to $S_{\max}$, but $\text{S}(2), \text{S}(3), \ldots, \text{S}(\text{NS}-1)$ need not be initialized, as they will be set internally by the routine in order to define a uniform mesh.

On exit: if MESH = 'U', the elements of S define a uniform mesh over $[S_{\min}, S_{\max}]$.

If MESH = 'C', the elements of S are unchanged.

Constraints:

if MESH = 'C', $\text{S}(1) \geq 0.0$ and $\text{S}(i) < \text{S}(i+1)$, for $i = 1, 2, \ldots, \text{NS}-1$;
if MESH = 'U', $0.0 \leq \text{S}(1) < \text{S}(\text{NS})$.

6: NT – INTEGER *Input*

On entry: the number of time-steps to be used in the finite difference method.

Constraint: NT ≥ 2.

7: T(NT) – REAL (KIND=nag_wp) array *Input/Output*

On entry: if MESH = 'C' then T(j) must contain the jth time in the mesh, for $j = 1, 2, \ldots, \text{NT}$. These values should be in increasing order, with $\text{T}(1) = t_{\min}$ and $\text{T}(\text{NT}) = t_{\max}$.

If MESH = 'U' then T(1) must be set to $t_{\min}$ and T(NT) to $t_{\max}$, but $\text{T}(2), \text{T}(3), \ldots, \text{T}(\text{NT}-1)$ need not be initialized, as they will be set internally by the routine in order to define a uniform mesh.

On exit: if MESH = 'U', the elements of T define a uniform mesh over $[t_{\min}, t_{\max}]$.

If MESH = 'C', the elements of T are unchanged.

Constraints:

if MESH = 'C', $\mathrm{T}(1) \geq 0.0$ and $\mathrm{T}(j) < \mathrm{T}(j+1)$, for $j = 1, 2, \ldots, \mathrm{NT}-1$;
if MESH = 'U', $0.0 \leq \mathrm{T}(1) < \mathrm{T}(\mathrm{NT})$.

8: TDPAR(3) – LOGICAL array *Input*

On entry: specifies whether or not various arguments are time-dependent. More precisely, r is time-dependent if TDPAR(1) = .TRUE. and constant otherwise. Similarly, TDPAR(2) specifies whether q is time-dependent and TDPAR(3) specifies whether σ is time-dependent.

9: R(∗) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array R must be at least NT if TDPAR(1) = .TRUE., and at least 1 otherwise.

On entry: if TDPAR(1) = .TRUE. then R(j) must contain the value of the risk-free interest rate $r(t)$ at the jth time in the mesh, for $j = 1, 2, \ldots, \mathrm{NT}$.

If TDPAR(1) = .FALSE. then R(1) must contain the constant value of the risk-free interest rate r. The remaining elements need not be set.

10: Q(∗) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array Q must be at least NT if TDPAR(2) = .TRUE., and at least 1 otherwise.

On entry: if TDPAR(2) = .TRUE. then Q(j) must contain the value of the continuous dividend $q(t)$ at the jth time in the mesh, for $j = 1, 2, \ldots, \mathrm{NT}$.

If TDPAR(2) = .FALSE. then Q(1) must contain the constant value of the continuous dividend q. The remaining elements need not be set.

11: SIGMA(∗) – REAL (KIND=nag_wp) array *Input*

Note: the dimension of the array SIGMA must be at least NT if TDPAR(3) = .TRUE., and at least 1 otherwise.

On entry: if TDPAR(3) = .TRUE. then SIGMA(j) must contain the value of the volatility $\sigma(t)$ at the jth time in the mesh, for $j = 1, 2, \ldots, \mathrm{NT}$.

If TDPAR(3) = .FALSE. then SIGMA(1) must contain the constant value of the volatility σ. The remaining elements need not be set.

12: ALPHA – REAL (KIND=nag_wp) *Input*

On entry: the value of λ to be used in the time-stepping scheme. Typical values include:

ALPHA = 0.0
Explicit forward Euler scheme.

ALPHA = 0.5
Implicit Crank–Nicolson scheme.

ALPHA = 1.0
Implicit backward Euler scheme.

The value 0.5 gives second-order accuracy in time. Values greater than 0.5 give unconditional stability. Since 0.5 is at the limit of unconditional stability this value does not damp oscillations.

Suggested value: ALPHA = 0.55.

Constraint: $0.0 \leq \mathrm{ALPHA} \leq 1.0$.

13: NTKEEP – INTEGER *Input*

On entry: the number of solutions to be stored in the time direction. The routine calculates the solution backwards from T(NT) to T(1) at all times in the mesh. These time solutions and the

corresponding Greeks will be stored at times T(i), for $i = 1, 2, \ldots,$ NTKEEP, in the arrays F, THETA, DELTA, GAMMA, LAMBDA and RHO. Other time solutions will be discarded. To store all time solutions set NTKEEP = NT.

Constraint: $1 \leq$ NTKEEP $\leq$ NT.

14: F(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*

On exit: F(i, j), for $i = 1, 2, \ldots,$ NS and $j = 1, 2, \ldots,$ NTKEEP, contains the value f of the option at the ith mesh point S(i) at time T(j).

15: THETA(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*
16: DELTA(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*
17: GAMMA(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*
18: LAMBDA(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*
19: RHO(LDF, NTKEEP) – REAL (KIND=nag_wp) array *Output*

On exit: the values of various Greeks at the ith mesh point S(i) at time T(j), as follows:

$$\text{THETA}(i,j) = \frac{\partial f}{\partial t}, \qquad \text{DELTA}(i,j) = \frac{\partial f}{\partial S}, \quad \text{GAMMA}(i,j) = \frac{\partial^2 f}{\partial S^2},$$

$$\text{LAMBDA}(i,j) = \frac{\partial f}{\partial \sigma}, \quad \text{RHO}(i,j) = \frac{\partial f}{\partial r}.$$

20: LDF – INTEGER *Input*

On entry: the first dimension of the arrays F, THETA, DELTA, GAMMA, LAMBDA and RHO as declared in the (sub)program from which D03NCF is called.

Constraint: LDF $\geq$ NS.

21: WORK($4 \times$ NS) – REAL (KIND=nag_wp) array *Workspace*

22: IWORK(NS) – INTEGER array *Workspace*

23: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, KOPT < 1,
or KOPT > 4,
or MESH $\neq$ 'U' or 'C',
or NS < 2,

or NT < 2,
or S$(1) < 0.0$,
or T$(1) < 0.0$,
or ALPHA < 0.0,
or ALPHA > 1.0,
or NTKEEP < 1,
or NTKEEP $>$ NT,
or LDF $<$ NS.

IFAIL $= 2$

MESH $=$ 'U' and the constraints:

S$(1) <$ S(NS),

T$(1) <$ T(NT)

are violated. Thus the end points of the uniform mesh are not in order.

IFAIL $= 3$

MESH $=$ 'C' and the constraints:

S$(i) <$ S$(i+1)$, for $i = 1, 2, \ldots,$ NS $- 1$,

T$(i) <$ T$(i+1)$, for $i = 1, 2, \ldots,$ NT $- 1$

are violated. Thus the mesh points are not in order.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy of the solution f and the various derivatives returned by the routine is dependent on the values of NS and NT supplied, the distribution of the mesh points, and the value of ALPHA chosen. For most choices of ALPHA the solution has a truncation error which is second-order accurate in S and first order accurate in t. For ALPHA $= 0.5$ the truncation error is also second-order accurate in t.

The simplest approach to improving the accuracy is to increase the values of both NS and NT.

8 Parallelism and Performance

D03NCF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

D03NCF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Timing

Each time-step requires the construction and solution of a tridiagonal system of linear equations. To calculate each of the derivatives LAMBDA and RHO requires a repetition of the entire solution process. The time taken for a call to the routine is therefore proportional to NS $\times$ NT.

9.2 Algorithmic Details

D03NCF solves equation (1) using a finite difference method. The solution is computed backwards in time from $t_{\max}$ to $t_{\min}$ using a λ scheme, which is implicit for all nonzero values of λ, and is unconditionally stable for values of $\lambda > 0.5$. For each time-step a tridiagonal system is constructed and solved to obtain the solution at the earlier time. For the explicit scheme ($\lambda = 0$) this tridiagonal system degenerates to a diagonal matrix and is solved trivially. For American options the solution at each time-step is inspected to check whether early exercise is beneficial, and amended accordingly.

To compute the arrays LAMBDA and RHO, which are derivatives of the stock value f with respect to the problem arguments σ and r respectively, the entire solution process is repeated with perturbed values of these arguments.

10 Example

This example, taken from Hull (1989), solves the one-dimensional Black–Scholes equation for valuation of a 5-month American put option on a non-dividend-paying stock with an exercise price of \$50. The risk-free interest rate is 10% per annum, and the stock volatility is 40% per annum.

A fully implicit backward Euler scheme is used, with a mesh of 20 stock price intervals and 10 time intervals.

10.1 Program Text

```
!   D03NCF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module d03ncfe_mod

!     D03NCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: print_greek
!     .. Parameters ..
      Integer, Parameter, Public       :: nin = 5, nout = 6
    Contains
      Subroutine print_greek(ns,ntkeep,nt,s,t,grname,greek)

!       .. Scalar Arguments ..
        Integer, Intent (In)           :: ns, nt, ntkeep
        Character (*), Intent (In)     :: grname
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: greek(ns,ntkeep), s(ns), t(nt)
!       .. Local Scalars ..
        Integer                        :: i, j
!       .. Intrinsic Procedures ..
        Intrinsic                      :: len
```

```
!       .. Executable Statements ..
        Write (nout,*)
        Write (nout,*) grname
        Write (nout,*)('-',i=1,len(grname))
        Write (nout,*) '  Stock Price  |   Time to Maturity (months)'
        Write (nout,99999) '|', (12.0_nag_wp*(t(nt)-t(i)),i=1,ntkeep)
        Write (nout,*) ' -----------------', ('------------',i=1,ntkeep)
        Do i = 1, ns
          Write (nout,99998) s(i), '|', (greek(i,j),j=1,ntkeep)
        End Do

        Return

99999   Format (16X,A,1X,12(1P,E12.4))
99998   Format (1X,1P,E12.4,3X,A,1X,12(1P,E12.4))
      End Subroutine print_greek
    End Module d03ncfe_mod

    Program d03ncfe

!     D03NCF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: d03ncf, nag_wp
      Use d03ncfe_mod, Only: nin, nout, print_greek
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Logical, Parameter               :: gprnt(5) = .True.
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: alpha, x
      Integer                          :: ifail, kopt, ldf, ns, nt, ntkeep
      Character (1)                    :: mesh
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: delta(:,:), f(:,:), gamma(:,:),      &
                                          lambda(:,:), rho(:,:), s(:), t(:),   &
                                          theta(:,:), work(:)
      Real (Kind=nag_wp)               :: q(3), r(3), sigma(3)
      Integer, Allocatable             :: iwork(:)
      Logical                          :: tdpar(3)
!     .. Executable Statements ..
      Write (nout,*) 'D03NCF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) ns, nt, ntkeep
      ldf = ns

      Allocate (delta(ldf,ntkeep),f(ldf,ntkeep),gamma(ldf,ntkeep),             &
        lambda(ldf,ntkeep),rho(ldf,ntkeep),s(ldf),t(nt),theta(ldf,ntkeep),     &
        work(4*ns),iwork(ns))

!     Read problem parameters

      Read (nin,*) kopt
      Read (nin,*) x
      Read (nin,*) mesh
      Read (nin,*) s(1), s(ns)
      Read (nin,*) t(1), t(nt)
      Read (nin,*) alpha

!     Set up input parameters for D03NCF

      Read (nin,*) tdpar(1:3)
      Read (nin,*) q(1), r(1), sigma(1)

!     Call Black-Scholes solver
      ifail = 0
      Call d03ncf(kopt,x,mesh,ns,s,nt,t,tdpar,r,q,sigma,alpha,ntkeep,f,theta,  &
        delta,gamma,lambda,rho,ldf,work,iwork,ifail)
```

```
!     Output option values and possibly Greeks.

      Call print_greek(ns,ntkeep,nt,s,t,'Option Values',f)

      If (gprnt(1)) Then
        Call print_greek(ns,ntkeep,nt,s,t,'Theta',theta)
      End If
      If (gprnt(2)) Then
        Call print_greek(ns,ntkeep,nt,s,t,'Delta',delta)
      End If
      If (gprnt(3)) Then
        Call print_greek(ns,ntkeep,nt,s,t,'Gamma',gamma)
      End If
      If (gprnt(4)) Then
        Call print_greek(ns,ntkeep,nt,s,t,'Lambda',lambda)
      End If
      If (gprnt(5)) Then
        Call print_greek(ns,ntkeep,nt,s,t,'Rho',rho)
      End If

    End Program d03ncfe
```

10.2 Program Data

```
D03NCF Example Program Data
  21  11  4               : ns, nt, ntkeep
  4                       : kopt
  50.                     : x
  'U'                     : mesh
  0.0  100.               : s(1), s(ns)
  0.0  0.4166667          : t(1), t(nt)
  1.0                     : alpha
  .FALSE. .FALSE. .FALSE. : tdpar
  0.0  0.1  0.4           : q(1), r(1), sigma(1)
```

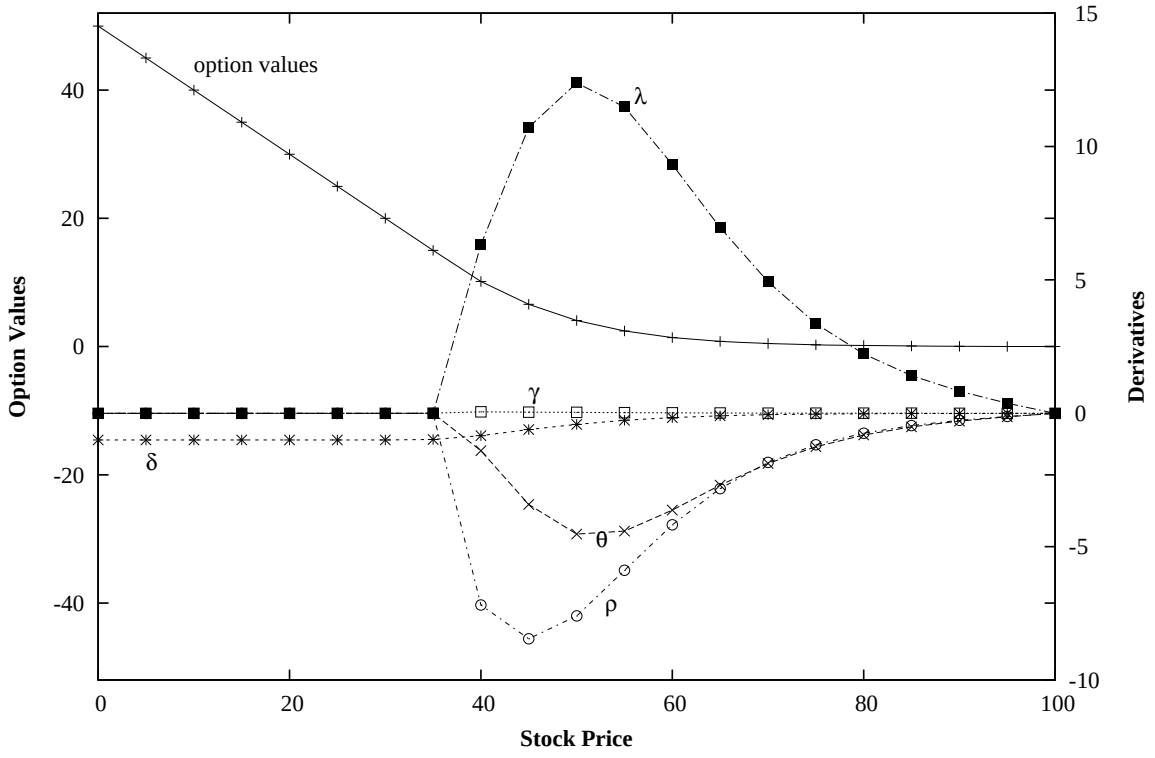
10.3 Program Results

```
 D03NCF Example Program Results


 Option Values
 -------------
   Stock Price  |   Time to Maturity (months)
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
  ------------------------------------------------------------------
   0.0000E+00   |   5.0000E+01  5.0000E+01  5.0000E+01  5.0000E+01
   5.0000E+00   |   4.5000E+01  4.5000E+01  4.5000E+01  4.5000E+01
   1.0000E+01   |   4.0000E+01  4.0000E+01  4.0000E+01  4.0000E+01
   1.5000E+01   |   3.5000E+01  3.5000E+01  3.5000E+01  3.5000E+01
   2.0000E+01   |   3.0000E+01  3.0000E+01  3.0000E+01  3.0000E+01
   2.5000E+01   |   2.5000E+01  2.5000E+01  2.5000E+01  2.5000E+01
   3.0000E+01   |   2.0000E+01  2.0000E+01  2.0000E+01  2.0000E+01
   3.5000E+01   |   1.5000E+01  1.5000E+01  1.5000E+01  1.5000E+01
   4.0000E+01   |   1.0154E+01  1.0096E+01  1.0046E+01  1.0012E+01
   4.5000E+01   |   6.5848E+00  6.4424E+00  6.2916E+00  6.1306E+00
   5.0000E+01   |   4.0672E+00  3.8785E+00  3.6729E+00  3.4463E+00
   5.5000E+01   |   2.4264E+00  2.2423E+00  2.0454E+00  1.8336E+00
   6.0000E+01   |   1.4174E+00  1.2662E+00  1.1096E+00  9.4813E-01
   6.5000E+01   |   8.1951E-01  7.0724E-01  5.9532E-01  4.8515E-01
   7.0000E+01   |   4.7241E-01  3.9411E-01  3.1904E-01  2.4845E-01
   7.5000E+01   |   2.7257E-01  2.2016E-01  1.7174E-01  1.2815E-01
   8.0000E+01   |   1.5725E-01  1.2328E-01  9.2935E-02  6.6682E-02
   8.5000E+01   |   8.9662E-02  6.8478E-02  5.0100E-02  3.4731E-02
   9.0000E+01   |   4.8449E-02  3.6251E-02  2.5901E-02  1.7469E-02
   9.5000E+01   |   2.1100E-02  1.5584E-02  1.0968E-02  7.2680E-03
   1.0000E+02   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00

 Theta
 -----
   Stock Price  |   Time to Maturity (months)
```

```
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
 ---------------------------------------------------------------
  0.0000E+00    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  5.0000E+00    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  1.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  1.5000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  2.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  2.5000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  3.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  3.5000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  4.0000E+01    |  -1.4043E+00 -1.1857E+00 -8.3285E-01 -2.8064E-01
  4.5000E+01    |  -3.4185E+00 -3.6183E+00 -3.8646E+00 -4.1880E+00
  5.0000E+01    |  -4.5285E+00 -4.9339E+00 -5.4387E+00 -6.0796E+00
  5.5000E+01    |  -4.4165E+00 -4.7277E+00 -5.0821E+00 -5.4821E+00
  6.0000E+01    |  -3.6294E+00 -3.7585E+00 -3.8748E+00 -3.9632E+00
  6.5000E+01    |  -2.6946E+00 -2.6860E+00 -2.6441E+00 -2.5561E+00
  7.0000E+01    |  -1.8790E+00 -1.8018E+00 -1.6941E+00 -1.5505E+00
  7.5000E+01    |  -1.2578E+00 -1.1621E+00 -1.0461E+00 -9.0969E-01
  8.0000E+01    |  -8.1539E-01 -7.2821E-01 -6.3006E-01 -5.2314E-01
  8.5000E+01    |  -5.0841E-01 -4.4106E-01 -3.6887E-01 -2.9433E-01
  9.0000E+01    |  -2.9276E-01 -2.4840E-01 -2.0237E-01 -1.5656E-01
  9.5000E+01    |  -1.3237E-01 -1.1079E-01 -8.8802E-02 -6.7378E-02
  1.0000E+02    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00

 Delta
 -----
   Stock Price  |   Time to Maturity (months)
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
 ---------------------------------------------------------------
  0.0000E+00    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  5.0000E+00    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  1.0000E+01    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  1.5000E+01    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  2.0000E+01    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  2.5000E+01    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  3.0000E+01    |  -1.0000E+00 -1.0000E+00 -1.0000E+00 -1.0000E+00
  3.5000E+01    |  -9.8457E-01 -9.9042E-01 -9.9536E-01 -9.9883E-01
  4.0000E+01    |  -8.4152E-01 -8.5576E-01 -8.7084E-01 -8.8694E-01
  4.5000E+01    |  -6.0871E-01 -6.2173E-01 -6.3735E-01 -6.5654E-01
  5.0000E+01    |  -4.1584E-01 -4.2000E-01 -4.2463E-01 -4.2970E-01
  5.5000E+01    |  -2.6498E-01 -2.6123E-01 -2.5633E-01 -2.4982E-01
  6.0000E+01    |  -1.6069E-01 -1.5351E-01 -1.4500E-01 -1.3485E-01
  6.5000E+01    |  -9.4501E-02 -8.7208E-02 -7.9055E-02 -6.9969E-02
  7.0000E+01    |  -5.4694E-02 -4.8708E-02 -4.2358E-02 -3.5699E-02
  7.5000E+01    |  -3.1515E-02 -2.7084E-02 -2.2610E-02 -1.8177E-02
  8.0000E+01    |  -1.8291E-02 -1.5168E-02 -1.2164E-02 -9.3423E-03
  8.5000E+01    |  -1.0880E-02 -8.7026E-03 -6.7034E-03 -4.9214E-03
  9.0000E+01    |  -6.8562E-03 -5.2894E-03 -3.9132E-03 -2.7463E-03
  9.5000E+01    |  -4.8449E-03 -3.6251E-03 -2.5901E-03 -1.7469E-03
  1.0000E+02    |  -4.2199E-03 -3.1168E-03 -2.1936E-03 -1.4536E-03

 Gamma
 -----
   Stock Price  |   Time to Maturity (months)
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
 ---------------------------------------------------------------
  0.0000E+00    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  5.0000E+00    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  1.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  1.5000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  2.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  2.5000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  3.0000E+01    |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
  3.5000E+01    |   6.1726E-03  3.8321E-03  1.8558E-03  4.6773E-04
  4.0000E+01    |   5.1047E-02  5.0031E-02  4.7953E-02  4.4288E-02
  4.5000E+01    |   4.2075E-02  4.3582E-02  4.5444E-02  4.7873E-02
  5.0000E+01    |   3.5072E-02  3.7109E-02  3.9646E-02  4.2863E-02
  5.5000E+01    |   2.5275E-02  2.6400E-02  2.7671E-02  2.9089E-02
  6.0000E+01    |   1.6442E-02  1.6688E-02  1.6860E-02  1.6900E-02
  6.5000E+01    |   1.0032E-02  9.8331E-03  9.5193E-03  9.0515E-03
  7.0000E+01    |   5.8907E-03  5.5669E-03  5.1595E-03  4.6562E-03
```

```
   7.5000E+01   |   3.3809E-03  3.0827E-03  2.7396E-03  2.3529E-03
   8.0000E+01   |   1.9091E-03  1.6834E-03  1.4388E-03  1.1808E-03
   8.5000E+01   |   1.0551E-03  9.0291E-04  7.4543E-04  5.8760E-04
   9.0000E+01   |   5.5449E-04  4.6239E-04  3.7065E-04  2.8244E-04
   9.5000E+01   |   2.5001E-04  2.0330E-04  1.5859E-04  1.1731E-04
   1.0000E+02   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00

 Lambda
 ------
   Stock Price  |   Time to Maturity (months)
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
  ------------------------------------------------------------------
   0.0000E+00   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   5.0000E+00   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   1.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   1.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   2.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   2.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   3.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   3.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   4.0000E+01   |   6.3243E+00  5.1893E+00  3.8089E+00  2.1118E+00
   4.5000E+01   |   1.0721E+01  9.9718E+00  9.2140E+00  8.4953E+00
   5.0000E+01   |   1.2381E+01  1.1807E+01  1.1228E+01  1.0636E+01
   5.5000E+01   |   1.1483E+01  1.0837E+01  1.0142E+01  9.3795E+00
   6.0000E+01   |   9.3227E+00  8.5840E+00  7.7870E+00  6.9211E+00
   6.5000E+01   |   6.9621E+00  6.2206E+00  5.4412E+00  4.6264E+00
   7.0000E+01   |   4.9268E+00  4.2651E+00  3.5937E+00  2.9227E+00
   7.5000E+01   |   3.3602E+00  2.8204E+00  2.2920E+00  1.7866E+00
   8.0000E+01   |   2.2221E+00  1.8126E+00  1.4248E+00  1.0683E+00
   8.5000E+01   |   1.4122E+00  1.1240E+00  8.5856E-01  6.2248E-01
   9.0000E+01   |   8.2686E-01  6.4587E-01  4.8252E-01  3.4083E-01
   9.5000E+01   |   3.7891E-01  2.9252E-01  2.1553E-01  1.4976E-01
   1.0000E+02   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00

 Rho
 ---
   Stock Price  |   Time to Maturity (months)
                |   5.0000E+00  4.5000E+00  4.0000E+00  3.5000E+00
  ------------------------------------------------------------------
   0.0000E+00   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   5.0000E+00   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   1.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   1.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   2.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   2.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   3.0000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   3.5000E+01   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
   4.0000E+01   |  -7.1918E+00 -6.0114E+00 -4.5204E+00 -2.5855E+00
   4.5000E+01   |  -8.4541E+00 -7.6378E+00 -6.8479E+00 -6.1657E+00
   5.0000E+01   |  -7.5988E+00 -6.9323E+00 -6.2879E+00 -5.6707E+00
   5.5000E+01   |  -5.8905E+00 -5.2837E+00 -4.6809E+00 -4.0772E+00
   6.0000E+01   |  -4.1854E+00 -3.6547E+00 -3.1306E+00 -2.6135E+00
   6.5000E+01   |  -2.8221E+00 -2.3904E+00 -1.9743E+00 -1.5775E+00
   7.0000E+01   |  -1.8437E+00 -1.5137E+00 -1.2055E+00 -9.2283E-01
   7.5000E+01   |  -1.1812E+00 -9.4071E-01 -7.2326E-01 -5.3162E-01
   8.0000E+01   |  -7.4513E-01 -5.7680E-01 -4.2921E-01 -3.0383E-01
   8.5000E+01   |  -4.5907E-01 -3.4659E-01 -2.5060E-01 -1.7161E-01
   9.0000E+01   |  -2.6550E-01 -1.9656E-01 -1.3892E-01 -9.2652E-02
   9.5000E+01   |  -1.2280E-01 -8.9807E-02 -6.2569E-02 -4.1033E-02
   1.0000E+02   |   0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
```

Example Program
Option Values and Derivatives at 5 Months to Maturity



Option Values and Derivatives at 3.5 Months to Maturity