NAG Toolbox for MATLAB

d03nc

1 Purpose

d03nc solves the Black–Scholes equation for financial option pricing using a finite-difference scheme.

2 Syntax

```
[s, t, f, theta, delta, gamma, lambda, rho, ifail] = d03nc(kopt, x, mesh, s, t, tdpar, r, q, sigma, ntkeep, 'ns', ns, 'nt', nt, 'alpha', alpha)
```

3 Description

d03nc solves the Black–Scholes equation Hull 1989, Wilmott *et al.* 1995

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

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

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 parameters r, q and σ may each be either constant or functions of time. The function also returns values of various Greeks.

d03nc 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 parameter **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 Parameters

5.1 Compulsory Input Parameters

1: **kopt – int32 scalar**

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** $= 1$ or 4.

2: **x – double scalar**

The exercise price X.

3: **mesh – string**

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: **s(ns) – double array**

If **mesh** $=$ 'C' then $\mathbf{s}(i)$ must contain the ith stock price in the mesh, for $i = 1, 2, \ldots, \mathbf{ns}$. These values should be in increasing order, with $\mathbf{s}(1) = S_{\min}$ and $\mathbf{s}(\mathbf{ns}) = S_{\max}$.

If **mesh** $=$ 'U' then $\mathbf{s}(1)$ must be set to $S_{\min}$ and $\mathbf{s}(\mathbf{ns})$ to $S_{\max}$, but $\mathbf{s}(2), \mathbf{s}(3), \ldots, \mathbf{s}(\mathbf{ns}-1)$ need not be initialized, as they will be set internally by the function in order to define a uniform mesh.

Constraints:

if **mesh** $=$ 'C', $\mathbf{s}(1) \geq 0.0$ and $\mathbf{s}(i) < \mathbf{s}(i+1)$, for $i = 1, 2, \ldots, \mathbf{ns}-1$;
if **mesh** $=$ 'U', $0.0 \leq \mathbf{s}(1) < \mathbf{s}(\mathbf{ns})$.

5: **t(nt) – double array**

If **mesh** $=$ 'C' then $\mathbf{t}(j)$ must contain the jth time in the mesh, for $j = 1, 2, \ldots, \mathbf{nt}$. These values should be in increasing order, with $\mathbf{t}(1) = t_{\min}$ and $\mathbf{t}(\mathbf{nt}) = t_{\max}$.

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

Constraints:

if **mesh** $=$ 'C', $\mathbf{t}(1) \geq 0.0$ and $\mathbf{t}(j) < \mathbf{t}(j+1)$, for $j = 1, 2, \ldots, \mathbf{nt}-1$;
if **mesh** $=$ 'U', $0.0 \leq \mathbf{t}(1) < \mathbf{t}(\mathbf{nt})$.

6: **tdpar(3) – logical array**

Specifies whether or not various parameters are time-dependent. More precisely, r is time-dependent if $\mathbf{tdpar}(1) = \mathbf{true}$ and constant otherwise. Similarly $\mathbf{tdpar}(2)$ specifies whether q is time-dependent, and $\mathbf{tdpar}(3)$ specifies whether σ is time-dependent.

7: **r**$(*)$ **– double array**

Note: the dimension of the array **r** must be at least **nt** if **tdpar**$(1) =$ **true**, and at least 1 otherwise.

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 $j = 1, 2, \ldots,$ **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.

8: **q**$(*)$ **– double array**

Note: the dimension of the array **q** must be at least **nt** if **tdpar**$(2) =$ **true**, and at least 1 otherwise.

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 $j = 1, 2, \ldots,$ **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.

9: **sigma**$(*)$ **– double array**

Note: the dimension of the array **sigma** must be at least **nt** if **tdpar**$(3) =$ **true**, and at least 1 otherwise.

If **tdpar**$(3) =$ **true** then **sigma**(j) must contain the value of the volatility $\sigma(t)$ at the jth time in the mesh, for $j = 1, 2, \ldots,$ **nt**.

If **tdpar**$(3) =$ **false** then **sigma**(1) must contain the constant value of the volatility σ. The remaining elements need not be set.

10: **ntkeep – int32 scalar**

the number of solutions to be stored in the time direction. The function 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**.

5.2 Optional Input Parameters

1: **ns – int32 scalar**

Default: The dimension of the array **s**.

the number of stock prices to be used in the finite-difference mesh.

Constraint: **ns** ≥ 2.

2: **nt – int32 scalar**

Default: The dimension of the array **t**.

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

Constraint: **nt** ≥ 2.

3: **alpha – double scalar**

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** $= 0.55$.

Default: 0.55

Constraint: $0.0 \leq$ **alpha** ≤ 1.0.

5.3 Input Parameters Omitted from the MATLAB Interface

ldf, work, iwork

5.4 Output Parameters

1: **s(ns) – double array**

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.

2: **t(nt) – double array**

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.

3: **f(ldf,ntkeep) – double array**

$\mathbf{f}(i,j)$, for $i = 1,2,\ldots,$ **ns** and $j = 1,2,\ldots,$ **ntkeep**, contains the value f of the option at the ith mesh point $\mathbf{s}(i)$ at time $\mathbf{t}(j)$.

4: **theta(ldf,ntkeep) – double array**
5: **delta(ldf,ntkeep) – double array**
6: **gamma(ldf,ntkeep) – double array**
7: **lambda(ldf,ntkeep) – double array**
8: **rho(ldf,ntkeep) – double array**

The values of various Greeks at the ith mesh point $\mathbf{s}(i)$ at time $\mathbf{t}(j)$, as follows:

$$\mathbf{theta}(i,j) = \frac{\partial f}{\partial t}, \quad \mathbf{delta}(i,j) = \frac{\partial f}{\partial S}, \quad \mathbf{gamma}(i,j) = \frac{\partial^2 f}{\partial S^2},$$
$$\mathbf{lambda}(i,j) = \frac{\partial f}{\partial \sigma}, \quad \mathbf{rho}(i,j) = \frac{\partial f}{\partial r}.$$

9: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

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:

$$\mathbf{s}(1) < \mathbf{s}(\mathbf{ns}),$$
$$\mathbf{t}(1) < \mathbf{t}(\mathbf{nt})$$

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

ifail $= 3$

mesh $=$ 'C' and the constraints:

$$\mathbf{s}(i) < \mathbf{s}(i+1), \text{ for } i = 1, 2, \ldots, \mathbf{ns} - 1,$$
$$\mathbf{t}(i) < \mathbf{t}(i+1), \text{ for } i = 1, 2, \ldots, \mathbf{nt} - 1$$

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

7 Accuracy

The accuracy of the solution f and the various derivatives returned by the function 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 Further Comments

8.1 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 function is therefore proportional to **ns** $\times$ **nt**.

8.2 Algorithmic Details

d03nc 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 parameters σ and r respectively, the entire solution process is repeated with perturbed values of these parameters.

9 Example

```
kopt = int32(4);
x = 50;
mesh = 'U';
s = [0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     100];
t = [0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0.4166667];
tdpar = [false; false; false];
r = [0.1];
q = [0];
sigma = [0.4];
ntkeep = int32(4);
[sOut, tOut, f, theta, delta, gamma, lambda, rho, ifail] = ...
    d03nc(kopt, x, mesh, s, t, tdpar, r, q, sigma, ntkeep,'alpha',1)
```

```
sOut =
     0
     5
    10
    15
    20
    25
    30
    35
    40
    45
    50
    55
    60
    65
    70
    75
```

```
    80
    85
    90
    95
   100
tOut =
         0
    0.0417
    0.0833
    0.1250
    0.1667
    0.2083
    0.2500
    0.2917
    0.3333
    0.3750
    0.4167
f =
   50.0000   50.0000   50.0000   50.0000
   45.0000   45.0000   45.0000   45.0000
   40.0000   40.0000   40.0000   40.0000
   35.0000   35.0000   35.0000   35.0000
   30.0000   30.0000   30.0000   30.0000
   25.0000   25.0000   25.0000   25.0000
   20.0000   20.0000   20.0000   20.0000
   15.0000   15.0000   15.0000   15.0000
   10.1543   10.0958   10.0464   10.0117
    6.5848    6.4424    6.2916    6.1306
    4.0672    3.8785    3.6729    3.4463
    2.4264    2.2423    2.0454    1.8336
    1.4174    1.2662    1.1096    0.9481
    0.8195    0.7072    0.5953    0.4851
    0.4724    0.3941    0.3190    0.2484
    0.2726    0.2202    0.1717    0.1282
    0.1573    0.1233    0.0929    0.0667
    0.0897    0.0685    0.0501    0.0347
    0.0484    0.0363    0.0259    0.0175
    0.0211    0.0156    0.0110    0.0073
         0         0         0         0
theta =
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
   -1.4043   -1.1857   -0.8328   -0.2806
   -3.4185   -3.6183   -3.8646   -4.1880
   -4.5285   -4.9339   -5.4387   -6.0796
   -4.4165   -4.7277   -5.0821   -5.4821
   -3.6294   -3.7585   -3.8748   -3.9632
   -2.6946   -2.6860   -2.6441   -2.5561
   -1.8790   -1.8018   -1.6941   -1.5505
   -1.2578   -1.1621   -1.0461   -0.9097
   -0.8154   -0.7282   -0.6301   -0.5231
   -0.5084   -0.4411   -0.3689   -0.2943
   -0.2928   -0.2484   -0.2024   -0.1566
   -0.1324   -0.1108   -0.0888   -0.0674
         0         0         0         0
delta =
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -1.0000   -1.0000   -1.0000   -1.0000
   -0.9846   -0.9904   -0.9954   -0.9988
```

```
   -0.8415   -0.8558   -0.8708   -0.8869
   -0.6087   -0.6217   -0.6373   -0.6565
   -0.4158   -0.4200   -0.4246   -0.4297
   -0.2650   -0.2612   -0.2563   -0.2498
   -0.1607   -0.1535   -0.1450   -0.1348
   -0.0945   -0.0872   -0.0791   -0.0700
   -0.0547   -0.0487   -0.0424   -0.0357
   -0.0315   -0.0271   -0.0226   -0.0182
   -0.0183   -0.0152   -0.0122   -0.0093
   -0.0109   -0.0087   -0.0067   -0.0049
   -0.0069   -0.0053   -0.0039   -0.0027
   -0.0048   -0.0036   -0.0026   -0.0017
   -0.0042   -0.0031   -0.0022   -0.0015
gamma =
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
    0.0062    0.0038    0.0019    0.0005
    0.0510    0.0500    0.0480    0.0443
    0.0421    0.0436    0.0454    0.0479
    0.0351    0.0371    0.0396    0.0429
    0.0253    0.0264    0.0277    0.0291
    0.0164    0.0167    0.0169    0.0169
    0.0100    0.0098    0.0095    0.0091
    0.0059    0.0056    0.0052    0.0047
    0.0034    0.0031    0.0027    0.0024
    0.0019    0.0017    0.0014    0.0012
    0.0011    0.0009    0.0007    0.0006
    0.0006    0.0005    0.0004    0.0003
    0.0003    0.0002    0.0002    0.0001
         0         0         0         0
lambda =
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
    6.3243    5.1893    3.8089    2.1118
   10.7215    9.9718    9.2140    8.4953
   12.3808   11.8073   11.2277   10.6365
   11.4834   10.8366   10.1417    9.3795
    9.3227    8.5840    7.7870    6.9211
    6.9621    6.2206    5.4412    4.6264
    4.9268    4.2651    3.5937    2.9227
    3.3602    2.8204    2.2920    1.7866
    2.2221    1.8126    1.4248    1.0683
    1.4122    1.1240    0.8586    0.6225
    0.8269    0.6459    0.4825    0.3408
    0.3789    0.2925    0.2155    0.1498
         0         0         0         0
rho =
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
   -7.1918   -6.0114   -4.5204   -2.5855
   -8.4541   -7.6378   -6.8479   -6.1657
   -7.5988   -6.9323   -6.2879   -5.6707
   -5.8905   -5.2837   -4.6809   -4.0772
```

```
   -4.1854   -3.6547   -3.1306   -2.6135
   -2.8221   -2.3904   -1.9743   -1.5775
   -1.8437   -1.5137   -1.2055   -0.9228
   -1.1812   -0.9407   -0.7233   -0.5316
   -0.7451   -0.5768   -0.4292   -0.3038
   -0.4591   -0.3466   -0.2506   -0.1716
   -0.2655   -0.1966   -0.1389   -0.0927
   -0.1228   -0.0898   -0.0626   -0.0410
         0         0         0         0
ifail =
           0
```