

NAG Library Function Document

nag_pde_bs_1d (d03ncc)

1 Purpose

nag_pde_bs_1d (d03ncc) solves the Black–Scholes equation for financial option pricing using a finite difference scheme.

2 Specification

```
#include <nag.h>
#include <nagd03.h>

void nag_pde_bs_1d (Nag_OptionType kopt, double x, Nag_MeshType mesh,
  Integer ns, double s[], Integer nt, double t[], const Nag_Boolean tdpar[],
  const double r[], const double q[], const double sigma[], double alpha,
  Integer ntkeep, double f[], double theta[], double delta[], double gamma[],
  double lambda[], double rho[], NagError *fail)
```

3 Description

nag_pde_bs_1d (d03ncc) solves the Black–Scholes equation (see Hull (1989) and Wilmott *et al.* (1995))

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

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

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 function also returns values of various Greeks.

nag_pde_bs_1d (d03ncc) 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** – Nag_OptionType *Input*

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

kopt = Nag_EuropeanCall
A European call option.

kopt = Nag_AmericanCall
An American call option.

kopt = Nag_EuropeanPut
A European put option.

- kopt** = Nag_AmericanPut
An American put option.
- Constraint:* **kopt** = Nag_EuropeanCall, Nag_AmericanCall, Nag_EuropeanPut or Nag_AmericanPut.
- 2: **x** – double *Input*
On entry: the exercise price X .
- 3: **mesh** – Nag_MeshType *Input*
On entry: indicates the type of finite difference mesh to be used:
mesh = Nag_UniformMesh
Uniform mesh.
mesh = Nag_CustomMesh
Custom mesh supplied by you.
Constraint: **mesh** = Nag_UniformMesh or Nag_CustomMesh.
- 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]** – double *Input/Output*
On entry: if **mesh** = Nag_CustomMesh, **s**[$i - 1$] must contain the i th stock price in the mesh, for $i = 1, 2, \dots, \mathbf{ns}$. These values should be in increasing order, with **s**[0] = S_{\min} and **s**[$\mathbf{ns} - 1$] = S_{\max} .
If **mesh** = Nag_UniformMesh, **s**[0] must be set to S_{\min} and **s**[$\mathbf{ns} - 1$] to S_{\max} , but **s**[1], **s**[2], \dots , **s**[$\mathbf{ns} - 2$] need not be initialized, as they will be set internally by the function in order to define a uniform mesh.
On exit: if **mesh** = Nag_UniformMesh, the elements of **s** define a uniform mesh over $[S_{\min}, S_{\max}]$.
If **mesh** = Nag_CustomMesh, the elements of **s** are unchanged.
Constraints:
if **mesh** = Nag_CustomMesh, **s**[0] ≥ 0.0 and **s**[$i - 1$] $<$ **s**[i], for $i = 1, 2, \dots, \mathbf{ns} - 1$;
if **mesh** = Nag_UniformMesh, $0.0 \leq \mathbf{s}[0] < \mathbf{s}[\mathbf{ns} - 1]$.
- 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]** – double *Input/Output*
On entry: if **mesh** = Nag_CustomMesh then **t**[$j - 1$] must contain the j th time in the mesh, for $j = 1, 2, \dots, \mathbf{nt}$. These values should be in increasing order, with **t**[0] = t_{\min} and **t**[$\mathbf{nt} - 1$] = t_{\max} .
If **mesh** = Nag_UniformMesh then **t**[0] must be set to t_{\min} and **t**[$\mathbf{nt} - 1$] to t_{\max} , but **t**[1], **t**[2], \dots , **t**[$\mathbf{nt} - 2$] need not be initialized, as they will be set internally by the function in order to define a uniform mesh.
On exit: if **mesh** = Nag_UniformMesh, the elements of **t** define a uniform mesh over $[t_{\min}, t_{\max}]$.
If **mesh** = Nag_CustomMesh, the elements of **t** are unchanged.
Constraints:
if **mesh** = Nag_CustomMesh, **t**[0] ≥ 0.0 and **t**[$j - 1$] $<$ **t**[j], for $j = 1, 2, \dots, \mathbf{nt} - 1$;
if **mesh** = Nag_UniformMesh, $0.0 \leq \mathbf{t}[0] < \mathbf{t}[\mathbf{nt} - 1]$.

- 8: **tdpar**[3] – const Nag_Boolean *Input*
On entry: specifies whether or not various arguments are time-dependent. More precisely, r is time-dependent if **tdpar**[0] = Nag_TRUE and constant otherwise. Similarly **tdpar**[1] specifies whether q is time-dependent, and **tdpar**[2] specifies whether σ is time-dependent.
- 9: **r**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **r** must be at least
nt when **tdpar**[0] = Nag_TRUE;
 1 otherwise.
On entry: if **tdpar**[0] = Nag_TRUE then **r**[$j - 1$] must contain the value of the risk-free interest rate $r(t)$ at the j th time in the mesh, for $j = 1, 2, \dots, \mathbf{nt}$.
 If **tdpar**[0] = Nag_FALSE then **r**[0] must contain the constant value of the risk-free interest rate r . The remaining elements need not be set.
- 10: **q**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **q** must be at least
nt when **tdpar**[1] = Nag_TRUE;
 1 otherwise.
On entry: if **tdpar**[1] = Nag_TRUE then **q**[$j - 1$] must contain the value of the continuous dividend $q(t)$ at the j th time in the mesh, for $j = 1, 2, \dots, \mathbf{nt}$.
 If **tdpar**[1] = Nag_FALSE then **q**[0] must contain the constant value of the continuous dividend q . The remaining elements need not be set.
- 11: **sigma**[*dim*] – const double *Input*
Note: the dimension, *dim*, of the array **sigma** must be at least
nt when **tdpar**[2] = Nag_TRUE;
 1 otherwise.
On entry: if **tdpar**[2] = Nag_TRUE then **sigma**[$j - 1$] must contain the value of the volatility $\sigma(t)$ at the j th time in the mesh, for $j = 1, 2, \dots, \mathbf{nt}$.
 If **tdpar**[2] = Nag_FALSE then **sigma**[0] must contain the constant value of the volatility σ . The remaining elements need not be set.
- 12: **alpha** – double *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: $0.0 \leq \mathbf{alpha} \leq 1.0$.
- 13: **ntkeep** – Integer *Input*
On entry: the number of solutions to be stored in the time direction. The function calculates the solution backwards from **t**[**nt** – 1] to **t**[0] at all times in the mesh. These time solutions and the

corresponding Greeks will be stored at times $\mathbf{t}[i - 1]$, for $i = 1, 2, \dots, \mathbf{ntkeep}$, in the arrays **f**, **theta**, **delta**, **gamma**, **lambda** and **rho**. Other time solutions will be discarded. To store all time solutions set $\mathbf{ntkeep} = \mathbf{nt}$.

Constraint: $1 \leq \mathbf{ntkeep} \leq \mathbf{nt}$.

14: **f** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

On exit: **f** $[\mathbf{ns} \times (j - 1) + i - 1]$, for $i = 1, 2, \dots, \mathbf{ns}$ and $j = 1, 2, \dots, \mathbf{ntkeep}$, contains the value f of the option at the i th mesh point $\mathbf{s}[i - 1]$ at time $\mathbf{t}[j - 1]$.

15: **theta** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

16: **delta** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

17: **gamma** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

18: **lambda** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

19: **rho** $[\mathbf{ns} \times \mathbf{ntkeep}]$ – double *Output*

On exit: the values of various Greeks at the i th mesh point $\mathbf{s}[i - 1]$ at time $\mathbf{t}[j - 1]$, as follows:

$$\mathbf{theta}[\mathbf{ns} \times (j - 1) + i - 1] = \frac{\partial f}{\partial t}, \quad \mathbf{delta}[\mathbf{ns} \times (j - 1) + i - 1] = \frac{\partial f}{\partial S}, \quad \mathbf{gamma}[\mathbf{ns} \times (j - 1) + i - 1] = \frac{\partial^2 f}{\partial S^2},$$

$$\mathbf{lambda}[\mathbf{ns} \times (j - 1) + i - 1] = \frac{\partial f}{\partial \sigma}, \quad \mathbf{rho}[\mathbf{ns} \times (j - 1) + i - 1] = \frac{\partial f}{\partial r}.$$

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

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **ns** = $\langle value \rangle$.

Constraint: **ns** ≥ 2 .

On entry, **nt** = $\langle value \rangle$.

Constraint: **nt** ≥ 2 .

On entry, **ntkeep** = $\langle value \rangle$.

Constraint: **ntkeep** ≥ 1 .

NE_INT_2

On entry, **ntkeep** = $\langle value \rangle$ and **nt** = $\langle value \rangle$.

Constraint: **ntkeep** $\leq \mathbf{nt}$.

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.

NE_NOT_STRICTLY_INCREASING

On entry, $\mathbf{s}[i] \leq \mathbf{s}[i - 1]$ in custom mesh: $i = \langle value \rangle$.

On entry, $\mathbf{t}[j] \leq \mathbf{t}[j - 1]$ in custom mesh: $j = \langle value \rangle$.

NE_REAL

On entry, **alpha** = $\langle value \rangle$.

Constraint: **alpha** \leq 1.0.

On entry, **alpha** = $\langle value \rangle$.

Constraint: **alpha** \geq 0.0.

On entry, **s**[0] < 0.0: **s**[0] = $\langle value \rangle$.

On entry, **t**[0] < 0.0: **t**[0] = $\langle value \rangle$.

NE_REAL_2

On entry, **s**[**ns** - 1] = $\langle value \rangle$ and **s**[0] = $\langle value \rangle$.

Constraint: **s**[**ns** - 1] > **s**[0].

On entry, **t**[**nt** - 1] = $\langle value \rangle$ and **t**[0] = $\langle value \rangle$.

Constraint: **t**[**nt** - 1] > **t**[0].

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

nag_pde_bs_1d (d03ncc) 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.

9 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.

9.1 Program Text

```

/* nag_pde_bs_1d (d03ncc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd03.h>

#define F(I, J)      f[ns*((J) -1)+(I) -1]
#define THETA(I, J) theta[ns*((J) -1)+(I) -1]
#define DELTA(I, J) delta[ns*((J) -1)+(I) -1]
#define GAMMA(I, J) gamma[ns*((J) -1)+(I) -1]
#define LAMBDA(I, J) lambda[ns*((J) -1)+(I) -1]
#define RHO(I, J)   rho[ns*((J) -1)+(I) -1]

int main(void)
{
    double      alpha, x;
    Integer     i, igreek, j, ns, nt, ntkeep, exit_status;
    double      *delta, *f, *gamma, *lambda, q[3], r[3], *rho, *s;
    double      sigma[3], *t, *theta, smin, smax, tmin, tmax;
    Nag_Boolean gprnt[5] = { Nag_TRUE, Nag_TRUE, Nag_TRUE, Nag_TRUE, Nag_TRUE };
    Nag_Boolean tdp[3];
    const char  *gname[5] = { "Theta", "Delta", "Gamma", "Lambda", "Rho" };
    NagError    fail;

    INIT_FAIL(fail);

    printf("nag_pde_bs_1d (d03ncc) Example Program Results\n\n");

    /* Skip heading in data file */

    scanf("%s[\n] ");
    exit_status = 0;

    /* Read problem parameters */

    scanf("%lf", &x);
    scanf("%ld%ld", &ns, &nt);
    scanf("%lf%lf", &smin, &smax);
    scanf("%lf%lf", &tmin, &tmax);
    scanf("%lf", &alpha);
    scanf("%ld", &ntkeep);

    /* Allocate memory */

    if (!(s = NAG_ALLOC(ns, double)) ||
        !(t = NAG_ALLOC(nt, double)) ||
        !(f = NAG_ALLOC(ns*ntkeep, double)) ||
        !(theta = NAG_ALLOC(ns*ntkeep, double)) ||
        !(delta = NAG_ALLOC(ns*ntkeep, double)) ||
        !(gamma = NAG_ALLOC(ns*ntkeep, double)) ||
        !(lambda = NAG_ALLOC(ns*ntkeep, double)) ||
        !(rho = NAG_ALLOC(ns*ntkeep, double)))
    {
        printf("Allocation failure\n");
        exit_status = 1;
        goto END;
    }

    /* Set up input parameters for nag_pde_bs_1d (d03ncc) */

    s[0] = smin;

```

```

s[ns-1] = smax;
t[0] = tmin;
t[nt-1] = tmax;
tdpar[0] = Nag_FALSE;
tdpar[1] = Nag_FALSE;
tdpar[2] = Nag_FALSE;
q[0] = 0.0;
r[0] = 0.10;
sigma[0] = 0.4;

/* Call Black-Scholes solver */

/* nag_pde_bs_ld (d03ncc).
 * Finite difference solution of the Black-Scholes equations
 */
nag_pde_bs_ld(Nag_AmericanPut, x, Nag_UniformMesh, ns, s,
              nt, t, tdpar, r, q, sigma, alpha, ntkeep, f,
              theta, delta, gamma, lambda, rho, &fail);

if (fail.code != NE_NOERROR)
{
    printf("Error from nag_pde_bs_ld (d03ncc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Output option values */

printf("\nOption Values\n");
printf("-----\n");
printf("%14s | %s\n", "Stock Price", "Time to Maturity (months)");
printf("%14s | ", "");
for (i = 0; i < ntkeep; i++) printf(" %13.4e", 12.0*(t[nt-1]-t[i]));
printf("\n");
for (i = 0; i < 74; i++) printf("-");
printf("\n");
for (i = 1; i <= ns; i++)
{
    printf(" %13.4e | ", s[i-1]);
    for (j = 1; j <= ntkeep; j++) printf(" %13.4e", F(i, j));
    printf("\n");
}

for (igreek = 0; igreek < 5; igreek++)
{
    if (!gprnt[igreek]) continue;

    printf("\n%s\n", gname[igreek]);
    for (i = 0; i < (Integer) strlen(gname[igreek]); i++) printf("-");
    printf("\n%14s | %s\n", "Stock Price",
           "Time to Maturity (months)");
    printf("%14s | ", "");
    for (i = 0; i < ntkeep; i++)
        printf(" %13.4e", 12.0*(t[nt-1]-t[i]));
    printf("\n");
    for (i = 0; i < 74; i++) printf("-");
    printf("\n");

    for (i = 1; i <= ns; i++)
    {
        printf(" %13.4e | ", s[i-1]);
        switch (igreek)
        {
            case 0:
                for (j = 1; j <= ntkeep; j++)
                    printf(" %13.4e", THETA(i, j));
                break;
            case 1:
                for (j = 1; j <= ntkeep; j++)
                    printf(" %13.4e", DELTA(i, j));
                break;
        }
    }
}

```

```

        case 2:
            for (j = 1; j <= ntkeep; j++)
                printf(" %13.4e", GAMMA(i, j));
            break;
        case 3:
            for (j = 1; j <= ntkeep; j++)
                printf(" %13.4e", LAMBDA(i, j));
            break;
        case 4:
            for (j = 1; j <= ntkeep; j++)
                printf(" %13.4e", RHO(i, j));
            break;
        default:
            break;
    }
    printf("\n");
}
}

END:
if (s) NAG_FREE(s);
if (t) NAG_FREE(t);
if (f) NAG_FREE(f);
if (theta) NAG_FREE(theta);
if (delta) NAG_FREE(delta);
if (gamma) NAG_FREE(gamma);
if (lambda) NAG_FREE(lambda);
if (rho) NAG_FREE(rho);

return exit_status;
}

```

9.2 Program Data

```

nag_pde_bs_1d (d03ncc) Example Program Data
50.
21 11
0.0 100.
0.0 0.4166667
1.0
4

```

9.3 Program Results

```

nag_pde_bs_1d (d03ncc) 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

