NAG Toolbox for Matlab

s30ja

1 Purpose

s30ja computes the European option price using the Merton jump-diffusion model.

2 Syntax

[p, ifail] = s30ja(calput, x, s, t, sigma, r, lambda, jvol, 'm', m, 'n', n)

3 Description

s30ja uses Merton’s jump-diffusion model (Merton (1976)) to compute the price of a European option. This assumes that the asset price is described by a Brownian motion with drift, as in the Black–Scholes–Merton case, together with a compound Poisson process to model the jumps. The corresponding stochastic differential equation is,

\[
dS_S = (\alpha - \lambda k)dt + \delta dW_t + dq_t.
\]

Here \( \alpha \) is the instantaneous expected return on the asset price, \( S \); \( \delta^2 \) is the instantaneous variance of the return when the Poisson event does not occur; \( dW_t \) is a standard Brownian motion; \( dq_t \) is the independent Poisson process.

This leads to the following price for a European option (see Haug (2007))

\[
P_{\text{call}} = \sum_{j=0}^{\infty} e^{-\lambda T} \left( \frac{\lambda T}{j!} \right) C_j(S, X, T, r, \sigma_j'),
\]

where \( T \) is the time to expiry; \( X \) is the strike price; \( r \) is the annual risk-free interest rate; \( C_j(S, X, T, r, \sigma_j') \) is the Black–Scholes–Merton option pricing formula for a European call (see s30aa).

\[
\sigma_j' = \sqrt{z^2 + \delta^2 \left( \frac{j}{T} \right)},
\]

\[
z^2 = \sigma^2 - \lambda \delta^2,
\]

\[
\delta^2 = \frac{\sigma^2}{\lambda},
\]

where \( \sigma \) is the total volatility including jumps; \( \lambda \) is the expected number of jumps given as an average per year; \( \gamma \) is the proportion of the total volatility due to jumps.

The value of a put is obtained by substituting the Black–Scholes–Merton put price for \( C_j(S, X, T, r, \sigma_j') \).

4 References


Merton R C (1976) Option pricing when underlying stock returns are discontinuous Journal of Financial Economics 3 125–144
5 Parameters

5.1 Compulsory Input Parameters

1: calput – string
   Determines whether the option is a call or a put.
   calput = 'C'
   A call. The holder has a right to buy.
   calput = 'P'
   A put. The holder has a right to sell.
   Constraint: calput = 'C' or 'P'.

2: x(m) – double array
   m, the dimension of the array, must satisfy the constraint m ≥ 1.
   x(i) must contain X_i, the i'th strike price, for i = 1, 2, ..., m.
   Constraint: x(i) ≥ z and x(i) ≤ 1 / z, where z = X02AMF(), the safe range parameter, for i = 1, 2, ..., m.

3: s – double scalar
   S, the price of the underlying asset.
   Constraint: s ≥ z and s ≤ 1 / z, where z = X02AMF(), the safe range parameter.

4: t(n) – double array
   n, the dimension of the array, must satisfy the constraint n ≥ 1.
   t(i) must contain T_i, the i'th time, in years, to expiry, for i = 1, 2, ..., n.
   Constraint: t(i) ≥ z, where z = X02AMF(), the safe range parameter, for i = 1, 2, ..., n.

5: sigma – double scalar
   σ, the annual total volatility, including jumps.
   Constraint: sigma > 0.0.

6: r – double scalar
   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.

7: lambda – double scalar
   λ, the number of expected jumps per year.
   Constraint: lambda > 0.0.

8: jvol – double scalar
   The proportion of the total volatility associated with jumps.
   Constraint: 0.0 ≤ jvol < 1.0.

5.2 Optional Input Parameters

1: m – int32 scalar
   Default: The dimension of the array x.
the number of strike prices to be used.

Constraint: \( m \geq 1 \).

2: \( n \) – int32 scalar

Default: The dimension of the array \( t \).

the number of times to expiry to be used.

Constraint: \( n \geq 1 \).

5.3 Input Parameters Omitted from the MATLAB Interface

ldp

5.4 Output Parameters

1: \( p(ldp,n) \) – double array

The \( m \times n \) array \( p \) contains the computed option prices.

2: \( ifail \) – int32 scalar

\( ifail = 0 \) 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, \( \text{calput} \neq 'C' \) or 'P'.

\( ifail = 2 \)

On entry, \( m \leq 0 \).

\( ifail = 3 \)

On entry, \( n \leq 0 \).

\( ifail = 4 \)

On entry, \( x(i) < z \) or \( x(i) > 1/z \), where \( z = X02AMF() \), the safe range parameter.

\( ifail = 5 \)

On entry, \( s < z \) or \( s > 1/z \), where \( z = X02AMF() \), the safe range parameter.

\( ifail = 6 \)

On entry, \( t(i) < z \), where \( z = X02AMF() \), the safe range parameter.

\( ifail = 7 \)

On entry, \( \text{sigma} \leq 0.0 \).

\( ifail = 8 \)

On entry, \( r < 0.0 \).

\( ifail = 9 \)

On entry, \( \text{lambda} \leq 0.0 \).
ifail = 10
    On entry, jvol < 0.0 or jvol ≥ 1.0.

ifail = 12
    On entry, ldp < m.

7  Accuracy

The accuracy of the output is dependent on the accuracy of the cumulative Normal distribution function, \( \Phi \), occurring in \( C_j \). This is evaluated using a rational Chebyshev expansion, chosen so that the maximum relative error in the expansion is of the order of the machine precision (see s15ab and s15ad). An accuracy close to machine precision can generally be expected.

8  Further Comments

None.

9  Example

```matlab
put = 'C';
lambda = 3;
s = 45;
sigma = 0.25;
r = 0.1;
jvol = 0.4;
x = [55.0];
t = [0.25];
[p, ifail] = s30ja(put, x, s, t, sigma, r, lambda, jvol);

fprintf('Merton Jump-Diffusion Model
European Call:
');
fprintf('Spot = %9.4f
', s);
fprintf('Volatility = %9.4f
', sigma);
fprintf('Rate = %9.4f
', r);
fprintf('Jumps = %9.4f
', lambda);
fprintf('Jump Vol = %9.4f

', jvol);

disp('Strike Expiry Option Price');
for i=1:1
    for j=1:1
        fprintf('%9.4f %9.4f %9.4f
', x(i), t(j), p(i,j));
    end
end
```

Merton Jump-Diffusion Model
European Call:

<table>
<thead>
<tr>
<th>Strike Expiry Option Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.0000 0.2500 0.2417</td>
</tr>
</tbody>
</table>

__________________________________________________________________________________________________________________________ ... ______________________________________________________________________