## 15.3 A quadratic approximation to American prices due to Barone–Adesi and Whaley.

We now discuss an approximation to the option price of an American option on a commodity, described in Barone-Adesi and Whaley (1987) (BAW).[1] The commodity is assumed to have a continuous payout $b$. The starting point for the approximation is the (Black-Scholes) stochastic differential equation valid for the value of any derivative with price $V$.

$$\frac{1}{2}\sigma^2 S^2 V_S S + bSV_S - rV + V_t = 0 \tag{15.1}$$

Here $V$ is the (unknown) formula that determines the price of the contingent claim. For an European option the value of $V$ has a known solution, the adjusted Black Scholes formula. For American options, which may be exercised early, there is no known analytical solution.

To do their approximation, BAW decomposes the American price into the European price and the early exercise premium

$$C(S,T) = c(S,T) + \varepsilon_C(S,T)$$

Here $\varepsilon_C$ is the early exercise premium. The insight used by BAW is that $\varepsilon_C$ must *also* satisfy the same partial differential equation. To come up with an approximation BAW transformed equation (15.1) into one where the terms involving $V_t$ are neglible, removed these, and ended up with a standard linear homeogenous second order equation, which has a known solution.

The functional form of the approximation is shown in formula 15.2.

$$C(S,T) = \begin{cases} c(S,T) + A_2\left(\frac{S}{S^*}\right)^{q_2} & \text{if} \quad S < S^* \\ S - X & \text{if} \quad S \geq S^* \end{cases}$$

where

$$q_2 = \frac{1}{2}\left(-(N-1) + \sqrt{(N-1)^2 + \frac{4M}{K}}\right)$$

$$A_2 = \frac{S^*}{q_2}\left(1 - e^{(b-r)(T-t)} N\left(d_1(S^*)\right)\right)$$

$$M = \frac{2r}{\sigma^2},\ N = \frac{2b}{\sigma^2},\ K(T) = 1 - e^{-r(T-t)}$$

and $S^*$ solves

$$S^* - X = c\left(S^*, T\right) + \frac{S^*}{q_2}\left(1 - e^{(b-r)(T-t)} N\left(d_1(S^*)\right)\right)$$

**Notation:** $S$ Stock price. $X$: Exercise price.

**Formula 15.2:** The functional form of the Barone Adesi Whaley approximation to the value of an American call

In implementing this formula, the only problem is finding the critical value $S^*$. This is the classical problem of finding a root of the equation

$$g(S^*) = S^* - X - c(S^*) - \frac{S^*}{q_2}\left(1 - e^{(b-r)(T-t)} N\left(d_1(S^*)\right)\right) = 0$$

[1] The approximation is also discussed in Hull (2011).

This is solved using Newton's algorithm for finding the root. We start by finding a first "seed" value $S_0$. The next estimate of $S_i$ is found by:

$$S_{i+1} = S_i - \frac{g()}{g'}$$

At each step we need to evaluate $g()$ and its derivative $g'()$.

$$g(S) = S - X - c(S) - \frac{1}{q_2} S \left(1 - e^{(b-r)(T-t)} N(d_1)\right)$$

$$g'(S) = (1 - \frac{1}{q_2}) \left(1 - e^{(b-r)(T-t)} N(d_1)\right) + \frac{1}{q_2}(e^{(b-r)(T-t)} n(d_1)) \frac{1}{\sigma\sqrt{T-t}}$$

where $c(S)$ is the Black Scholes value for commodities. Code 15.3 shows the implementation of this formula for the price of a call option.

**Example**

Consider the following set of parameters, used as an example in the Barone-Adesi and Whaley (1987) paper: $S = 100$, $X = 100$, $\sigma = 0.20$, $r = 0.08$, $b = -0.04$.

1. Price a call option with time to maturity of 3 months.

C++ program:

```
double S = 100; double X = 100;   double sigma = 0.20;
double r = 0.08; double b = −0.04; double time = 0.25;
cout << " Call price using Barone-Adesi Whaley approximation = "
      << option_price_american_call_approximated_baw(S,X,r,b,sigma,time) << endl;
```

Output from C++ program:

```
Call price using Barone-Adesi Whaley approximation = 5.74339
```

**Exercise 15.1.**

The Barone-Adesi – Whaley insight can also be used to value a put option, by approximating the value of the early exercise premium. For a put option the approximation is

$$P(S) = \begin{array}{ll} p(S,T) + A_1 \left(\frac{S}{S^{**}}\right)^{q_1} & \text{if} \quad S > S^{**} \\ X - S & \text{if} \quad S \leq S^{**} \end{array}$$

$$A_1 = -\frac{S^{**}}{q1}(1 - e^{(b-r)(T-t)} N(-d_1(S^{**}))$$

One again solves iteratively for $S^{**}$, for example by Newton's procedure, where now one would use

$$g(S) = X - S - p(S) + \frac{S}{q_1}\left(1 - e^{(b-r)(T-t)} N(-d_1)\right)$$

$$g'(S) = (\frac{1}{q_1} - 1)\left(1 - e^{(b-r)(T-t)} N(-d_1)\right) + \frac{1}{q_1} e^{(b-r)(T-t)} \frac{1}{\sigma\sqrt{T-t}} n(-d_1)$$

1. Implement the calculation of the price of an American put option using the BAW approach.

```
#include <cmath>
#include <algorithm>
using namespace std;
#include "normdist.h"          // normal distribution
#include "fin_recipes.h"      // define other option pricing formulas

const double ACCURACY=1.0e-6;

double option_price_american_call_approximated_baw( const double& S,
                                                     const double& X,
                                                     const double& r,
                                                     const double& b,
                                                     const double& sigma,
                                                     const double& time) {
    double sigma_sqr = sigma*sigma;
    double time_sqrt = sqrt(time);
    double nn = 2.0*b/sigma_sqr;
    double m = 2.0*r/sigma_sqr;
    double K = 1.0-exp(-r*time);
    double q2 = (-(nn-1)+sqrt(pow((nn-1),2.0)+(4*m/K)))*0.5;

    double q2_inf = 0.5 * ( -(nn-1) + sqrt(pow((nn-1),2.0)+4.0*m)); // seed value from paper
    double S_star_inf = X / (1.0 - 1.0/q2_inf);
    double h2 = -(b*time+2.0*sigma*time_sqrt)*(X/(S_star_inf-X));
    double S_seed = X + (S_star_inf-X)*(1.0-exp(h2));

    int no_iterations=0; // iterate on S to find S_star, using Newton steps
    double Si=S_seed;
    double g=1;
    double gprime=1.0;
    while ((fabs(g) > ACCURACY)
           && (fabs(gprime)>ACCURACY) // to avoid exploding Newton's
           && ( no_iterations++<500)
           && (Si>0.0)) {
        double c  = option_price_european_call_payout(Si,X,r,b,sigma,time);
        double d1 = (log(Si/X)+(b+0.5*sigma_sqr)*time)/(sigma*time_sqrt);
        g=(1.0-1.0/q2)*Si-X-c+(1.0/q2)*Si*exp((b-r)*time)*N(d1);
        gprime=( 1.0-1.0/q2)*(1.0-exp((b-r)*time)*N(d1))
            +(1.0/q2)*exp((b-r)*time)*n(d1)*(1.0/(sigma*time_sqrt));
        Si=Si-(g/gprime);
    };
    double S_star = 0;
    if (fabs(g)>ACCURACY) { S_star = S_seed; } // did not converge
    else { S_star = Si; };
    double C=0;
    double c  = option_price_european_call_payout(S,X,r,b,sigma,time);
    if (S>=S_star) {
        C=S-X;
    }
    else {
        double d1 = (log(S_star/X)+(b+0.5*sigma_sqr)*time)/(sigma*time_sqrt);
        double A2 = (1.0-exp((b-r)*time)*N(d1))* (S_star/q2);
        C=c+A2*pow((S/S_star),q2);
    };
    return max(C,c); // know value will never be less than BS value
};
```

C++ **Code 15.3:** Barone Adesi quadratic approximation to the price of a call option