NAG Toolbox for MATLAB

Implied Volatility Exercise

Solution to Question 1

The following MATLAB program solves this problem:

```
function option_price
  S = 100.0; % Price of underlying asset
  K = 90.0;  % Strike price
  T = 1.5;   % Time to maturity in years
  r = 0.03;  % Risk-free interest rate
  q = 0.015; % Dividend rate
  sigma = 0.09;  % Volatility of underlying asset - N.B. 0.09 means 9%
  [price] = bs(S, K, T, r, q, sigma);
  fprintf('Option price is %g\n', price);
end

% A function to compute the European option price given by the
% Black-Scholes-Merton formula, for a "Call" option.
function [price] = bs(S, K, T, r, q, sigma)
    calput = 'Call';
    [p, ifail] = nag_specfun_opt_bsm_price(calput,K,S,T,sigma,r,q);
    price = p;
end
```

Solution to Question 2

The following MATLAB program solves this problem:

```
function implied_volatility

    S = 100;     % Stock price
    K = 90;      % Strike price
    T = 1.5;     % Time to maturity, in years
    r = 0.03;    % Risk-free interest rate
    q = 0.015;   % Dividend rate

  % The target price for which we wish to determine the implied volatility
    Ctarget = 12.35008695;

  % Store all the parameters in a 'user' array so that
  % they can be passed easily to the evaluation routine

    user(1) = S;
    user(2) = K;
    user(3) = T;
    user(4) = r;
    user(5) = q;
```

```
    user(6) = Ctarget;

    fprintf('Calculation of Black-Scholes-Merton implied volatility ');
    fprintf('for a European "call" option, with:\n');
    fprintf('  Stock price: %g\n', S);
    fprintf('  Strike price: %g\n', K);
    fprintf('  Time to maturity: %g\n', T);
    fprintf('  Risk-free interest rate: %g\n', r);
    fprintf('  Dividen rate: %g\n\n', q);

  % An initial guess at what the volatility might be
    sigma = 0.15;

  % Accuracy parameters of the NAG root finding routine
    eps = 1.0e-6;
    eta = 1.0e-6;
    nfmax = nag_int(1500);

  % Call the NAG root finder to calculate the implied volatility
    [impvol, user, ifail] = nag_roots_contfn_cntin(sigma,eps,eta,@bs, ...
        nfmax,'user',user);
    fprintf('  Implied volatility for target price %7.4f = %9.6f\n', ...
        Ctarget, impvol);

  % Repeat with a different target price
    Ctarget = 25.5;
    user(6) = Ctarget;
    sigma = 0.15;
    [impvol, user, ifail] = nag_roots_contfn_cntin(sigma,eps,eta,@bs, ...
        nfmax,'user',user);
    fprintf('  Implied volatility for target price %7.4f = %9.6f\n', ...
        Ctarget, impvol);
end

% A function to compute the difference between a European option price
% given by the Black-Scholes-Merton formula, and a target price
function [result, user] = bs(sigma, user)
    calput = 'Call';
  % Get the BSM parameters back from the 'user' array
    S = user(1);  % Stock price
    K = user(2);  % Strike price
    T = user(3);  % Time to maturity, in years
    r = user(4);  % Interest rate
    q = user(5);  % Dividend rate
    [p, ifail] = nag_specfun_opt_bsm_price(calput,K,S,T,sigma,r,q);
  % The target price was stored in user(6), and the current price is in p.
  % We are seeking to find p = target price.
    result = p - user(6);
end
```