

```

function [] = TrinomialAmerican(CallPut, AssetP, Strike, RiskFree, Div,
Time, Vol, nSteps)

% Computes the Boyle (1986) Trinomial Tree for American Call/Put Option
Values based
% on the following inputs:
% CallPut          =      Call = 1, Put = 0
% AssetP           =      Underlying Asset Price
% Strike           =      Strike Price of Option
% RiskFree         =      Risk Free rate of interest
% Div              =      Dividend Yield of Underlying
% Time             =      Time to Maturity
% Vol              =      Volatility of the Underlying
% nSteps           =      Number of Time Steps for Trinomial Tree to
take
% Please note that the use of this code is not restricted in anyway.
% However, referencing the author of the code would be appreciated.
% To run this program, simply use the function defined in the 1st line.
% http://www.global-derivatives.com
% info@global-derivatives.com
% Kevin Cheng (Nov 2003)

dt = Time / nSteps;           % Allocates the time steps
cc = RiskFree - Div;         % Specifies the cost of carry (r - D)

if CallPut
    b = 1;
end
if ~CallPut
    b = -1;
end

RR = exp(RiskFree * dt);
Up = exp(Vol * sqrt(2 * dt)); % The magnitude of an up movement
Down = 1 / Up;               % The magnitude of a down movement

%%% Specifies the probability of up, down and mid moves for trinomial tree
P_up = ((exp(cc * dt / 2) - exp(-Vol * sqrt(dt / 2))) / (exp(Vol * sqrt(dt
/ 2)) - exp(-Vol * sqrt(dt / 2)))) ^ 2;
P_down = ((exp(Vol * sqrt(dt / 2)) - exp(cc * dt / 2)) / (exp(Vol *
sqrt(dt / 2)) - exp(-Vol * sqrt(dt / 2)))) ^ 2;
P_mid = 1 - P_up - P_down;
Df = exp(-RiskFree * dt);

% Sets up the asset movements on the trinomial tree
for i = 0:(2 * nSteps)
    State = i + 1;
    Value(State) = max(0, b * (AssetP * Up ^ max(i - nSteps, 0) * Down ^

```

```

        max(nSteps * 2 - nSteps - i, 0) - Strike));

end

% Works backwards recursively to determine the price of the option
for TT = (nSteps - 1):-1:0
    for i = 0:(TT * 2)
        State = i + 1;
        Value(State) = max((P_up * Value(State + 2) + P_mid * Value(State
            + 1) + P_down * Value(State)) * Df, b * (AssetP *
            Up ^ max(i - TT, 0) * Down ^ max(TT * 2 - TT - i, 0)
            - Strike));
    end
end

Trinomial = Value(1)

```