# MFE 230Q
# Assignment 4 Solutions

## 1 Exact Price

We have

$$\frac{dS}{S} = r\,dt + \sigma\,dW^Q$$

The payout of a digital call option on strike $K$ and maturity $T$ is given by

$$\Phi(S_T) = \begin{cases} 1 & S_T \geq K\,, \\ 0 & S_T < K\,. \end{cases}$$

As usual one finds that $\log(\frac{S_T}{S_0}) = (r - \frac{\sigma^2}{2})T + \sigma W_T$. It is convenient to write $\sigma W_T = \sigma\sqrt{T}Z$, where $Z \sim N(0,1)$. Then we have $S_T \geq K$ if and only if $\log(S_T) \geq \log K$, which is equivalent to $\log(\frac{S_0}{K}) + (r - \frac{\sigma^2}{2})T + \sigma\sqrt{T}Z \geq 0$. The time 0 value of the option is then given by

$$C(0) = e^{-rT}\mathbb{E}_0^{\mathbb{Q}}\left[\Phi(S_T)\right] = e^{-rT}\mathbb{E}_0^{\mathbb{Q}}\left[\chi_{\{S_T \geq K\}}\right] = e^{-rT}\mathbb{E}_{\mathbb{Q}}\left[\chi_{\{Z \geq -d_2\}}\right]$$

where $\chi_A(x)$ is the usual set indicator function and $d_2 = \frac{\log(S_0/K)+(r-\sigma^2/2)T}{\sigma\sqrt{T}}$. Let $n(z)$ denote the standard normal density function. We obtain

$$\begin{aligned} C(0) &= e^{-rT}\int_{-\infty}^{\infty} \chi_{\{z \geq -d_2\}} n(z)\,dz \\ &= e^{-rT}\int_{-d_2}^{\infty} n(z)\,dz \\ &= e^{-rT}\int_{-\infty}^{d_2} n(z)\,dz \\ &= e^{-rT}N(d_2) \end{aligned}$$

where $N(x)$ is the standard normal cumulative distribution function.

For $S_0 = 100$, $K = 110$, $\sigma = 0.16$, $r = 0.1$, and $T = 1.0$, we have

$$C(0) \approx \$0.434129\,. \tag{1}$$

## 2 Numerical Approximations

In Table 1, we summarize the numerical results. For the deterministic algorithms, we display the minimum value $N_0$ such that the approximation error is less than \$0.01 for **all** $N \geq N_0$. For the two Monte Carlo methods, we instead display the value of $N_0$ that provides an estimate of the price to within \$0.01 with 95% confidence [1] For each $N_0$, we also include the running time incurred by the executing the routine with this $N_0$. In Appendix A, sample routines in Matlab for each numerical method are included.

Table 1: Numerical results

| | $N_0$ | $t$ (seconds) |
|---|---|---|
| Binomial Tree | 1244 | 0.0213 |
| Finite Difference | 851 | 0.0523 |
| Monte Carlo | 8000 | 0.000534 |
| Monte Carlo (antithetic) | 500 | 0.000262 |

## A Matlab code

### A.1 Exact solution

```
function [ exact ] = digital_call(r,T,s,S0,K)
% Black-Scholes value of a digital call

d2 = (log(S0/K) + (r-0.5*s^2)*T)/(s*sqrt(T));
exact = exp(-r*T)*normcdf(d2);
end
```

### A.2 Binomial Tree

```
function [ price ] = digital_call_binom( N,r,T,s,S0,K )
% Binomial tree pricing of a digital call

tic
dt = T/N;
R = exp(r*dt);
u = exp(s*sqrt(dt));
d = 1.0/u;
q = (R-d)/(u-d);

now = zeros(N+1,1);
for i=1:N+1
    now(i) = (S0*(u^(N-i+1))*(d^(i-1)) > K);
end

for i=1:N
    prev = zeros(N-i+1,1);
    for j=1:length(prev)
        prev(j) = (1/R)*(q*now(j)+(1-q)*now(j+1));
    end
    now=prev;
end
```

[1] These values of $N_0$ are approximate. Since the algorithm uses a random number generator, there is no guarantee that the 95% confidence width will be less than \$0.01 every time the routine is run with the same $N_0$.

```
price = prev(1);
toc
end
```

## A.3 Monte Carlo

```
function [ price,sd,se ] = digital_call_mc(N,r,T,s,S0,K,anti)
% Monte Carlo pricing of a digital call
%   anti='True' for antithetic paths
%   anti='False' for no antithetic paths

tic
Z = randn(N,1);
payouts = (S0.*exp((r-0.5*s^2)*T + s*sqrt(T).*Z) > K);

if strcmp(anti,'False')
    price = exp(-r*T)*mean(payouts);
    sd = exp(-r*T)*std(payouts);
    se = sd/sqrt(N);
else
    neg_payouts = (S0.*exp((r-0.5*s^2)*T - s*sqrt(T).*Z) > K);
    anti_payouts = 0.5*(payouts + neg_payouts);
    price = exp(-r*T)*mean(anti_payouts);
    sd = exp(-r*T)*std(anti_payouts);
    se = sd/sqrt(N);
end
toc
end
```

## A.4 Finite Differences

```
function [ price ] = digital_call_fd( N,r,T,s,S0,K )
% Explicit finite difference algorithm to price a digital call

tic
k = T/N;
h = s*sqrt(3*k);

a = (k/(2*h))*(s^2*(1/h + 1/2) - r);
b = 1 - (k*s^2)/(h^2) - k*r;
c = (k/(2*h))*(s^2*(1/h - 1/2) + r);

now = ((log(S0)-N*h:h:log(S0)+N*h) > log(K));

for i=1:N
    prev = zeros(2*(N-i+1)-1,1);
    for j=1:length(prev)
        prev(j) = a*now(j)+b*now(j+1)+c*now(j+2);
    end
    now = prev;
end

price = prev(1);
toc
end
```