Stochastic Calculus Problem Set I: Question 1

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
# As specified in question setup and part (e)
u = 1.005
d = 1.002
r = 0.003
p_1 = 0.4
p_2 = 0.6
N = 100
L = 1000
S_0 = 1

# Assert that model is arbitrage-free
assert d < 1+r, 'd >= 1+r. This model is not arbitrage-free.'
assert 1+r < u, '1+r >= u. This model is not arbitrage-free.'
```

Part (a)

For the binomial asset pricing model, $$ S_n = (\prod_{i=1}^{N}{y_i}) S_0 $$

where $y_i = u$ if the result of the ith coin toss is heads or $y_i = d$ if the result of the ith coin toss is tails.

Let $$ R_n = \log(\frac{S_n}{S_0}) $$

Then,

$$ R_n = \sum_{i=1}^{N}{\log y_i} $$

$$ \implies R_n = (\log u - \log d)\sum_{i=1}^{N}{Y_n} + n\log d$$

where $Y_n \sim B(n, p)$.

Thus,

$$ E(R_n) = (\log u - \log d)(np) + n\log(d) = n(p\log u + (1-p)\log d) $$

and

$$ Var(R_n) = (\log u - \log d)^2 np(1-p) $$

Part (b)

In [3]:

```
def risk_neutral_probabilities(u, d, r):
    '''
    Computes risk neutral probabilities from binomial
    model parameters (u and d) and risk-free rate (r).
    '''
    p_tilde = ((1+r) - d) / (u - d)
    q_tilde = (u - (1+r)) / (u - d)
    return p_tilde, q_tilde


p_tilde, q_tilde = risk_neutral_probabilities(u, d, r)
```

Part (c)

In [4]:

```
def stock_value(path):
    '''
    Computes the value of a stock given some path
    (a NumPy array of 0s and 1s).
    Returns the value of the stock at time N.
    '''
    return S_0 * np.prod((u - d)*path + d)


path = np.random.randint(low=0, high=2, size=N)
```

In [5]:

```
def one_step_back(omega_np1, X_np1, derivative_value, stock_value):
    heads_case = np.append(omega_np1[:-1], 1)
    tails_case = np.append(omega_np1[:-1], 0)
    numerator = derivative_value(heads_case) - derivative_value(tails_case)
    denominator = stock_value(heads_case) - stock_value(tails_case)
    delta_n = numerator / denominator
    X_n = 1/(1+r) * (X_np1 - delta_n*(stock_value(omega_np1) - (1+r)*stock_value(omega_np1[:-1])))
    return delta_n, X_n
```

Part (d)

In [6]:

```
def exotic_derivative(path):
    return max(S_0 * np.cumprod((u-d)*path + d))
```

In [7]:

```
def european_call(paths, strike_price):
    # If there is only one path, coerce to 2D
    if len(paths.shape) == 1:
        paths = np.atleast_2d(paths)
    S_N = (S_0 * np.cumprod((u-d)*paths + d, axis=-1))[:, -1]
    out = S_N.copy()
    out[S_N - strike_price <= 0] = 0
    return out
```

In [8]:

```
def european_put(paths, strike_price):
    # If there is only one path, coerce to 2D
    if len(paths.shape) == 1:
        paths = np.atleast_2d(paths)
    S_N = (S_0 * np.cumprod((u-d)*paths + d, axis=-1))[:, -1]
    out = S_N.copy()
    out[strike_price - S_N <= 0] = 0
    return out
```

Part (e)

In [9]:

```
probabilities = [p_tilde, p_1, p_2]
strike_prices = [S_0 * np.exp(N*(p*np.log(u) + (1-p)*np.log(d)))
                 for p in probabilities]
```

In [10]:

```
# Bernoulli(p) = B(1, p)
paths_1 = np.random.binomial(n=1, p=p_1, size=[L, N])
E1_V_N_call = [np.mean(european_call(paths_1, K)) for K in strike_prices]
E1_V_N_put = [np.mean(european_put(paths_1, K)) for K in strike_prices]
```

```
paths_2 = np.random.binomial(n=1, p=p_2, size=[L, N])
E2_V_N_call = [np.mean(european_call(paths_2, K)) for K in strike_prices]
E2_V_N_put = [np.mean(european_put(paths_2, K)) for K in strike_prices]
```

Part (f)

By the martingale property, $V_0 = \tilde{E}(\frac{V_N}{(1+r)^N})$. But we compute $\tilde{E}$ using a Monte Carlo approach, so we need only average.

In [15]:

```
paths_tilde = np.random.binomial(n=1, p=p_tilde, size=[L, N])
Etilde_V_0_call = [np.mean(european_call(paths_tilde, K) / (1+r)**N) for K in strike_prices]
Etilde_V_0_put = [np.mean(european_put(paths_tilde, K) / (1+r)**N) for K in strike_prices]
```

Part (g)

In [45]:

```
path = np.random.binomial(n=1, p=1/2, size=N)

delta_nm1, X_nm1  = \
    one_step_back(path, stock_value(path),
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm2, X_nm2 = \
    one_step_back(path[:-1], X_nm1,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm3, X_nm3 = \
    one_step_back(path[:-1], X_nm2,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)

X_nm3
```

Out[45]:

```
1.4137740635846199
```

In [46]:

```
path = np.random.binomial(n=1, p=1/2, size=N)

delta_nm1, X_nm1  = \
    one_step_back(path, stock_value(path),
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm2, X_nm2 = \
    one_step_back(path[:-1], X_nm1,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm3, X_nm3 = \
    one_step_back(path[:-1], X_nm2,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)

X_nm3
```

Out[46]:

```
1.409570706721379
```

In [47]:

```
path = np.random.binomial(n=1, p=1/2, size=N)

delta_nm1, X_nm1  = \
    one_step_back(path, stock_value(path),
```

```python
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm2, X_nm2 = \
    one_step_back(path[:-1], X_nm1,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)
delta_nm3, X_nm3 = \
    one_step_back(path[:-1], X_nm2,
                  lambda x: european_call(x, strike_prices[0]).item(),
                  stock_value)

X_nm3
```

Out[47]:

```
1.4053630329699724
```

Part (h)

In [48]:

```python
results = np.vstack([
    E1_V_N_call,
    E1_V_N_put,
    E2_V_N_call,
    E2_V_N_put,
    Etilde_V_0_call,
    Etilde_V_0_put
])

names = [
    'E1_V_N_call',
    'E1_V_N_put',
    'E2_V_N_call',
    'E2_V_N_put',
    'Etilde_V_0_call',
    'Etilde_V_0_put'
]

probabilities = ['p_tilde', 'p_1', 'p_2']

pd.DataFrame(data=results,
             index=names,
             columns=probabilities)
```

Out[48]:

	p_tilde	p_1	p_2
E1_V_N_call	1.265483	0.705913	0.000000
E1_V_N_put	0.111365	0.664053	1.376848
E2_V_N_call	1.461309	1.461309	0.734896
E2_V_N_put	0.000000	0.000000	0.710341
Etilde_V_0_call	0.490610	0.064725	0.000000
Etilde_V_0_put	0.509126	0.927871	0.999736