Assignment 2

April 2018

Long Thanh NGUYEN

long.nguyen2017@qcf.jvn.edu.vn

Question 1

(European Put-Call parity revisited) We consider portfolio A: long a call and short a put on the same underlying S, strike-price K and maturity T. Portfolio B consists of a long prepaid forward contract on S for the same maturity T, as well as borrowing the present value of the strike-price K to be repaid at T.

a. What is the initial cost, intermediate cash-flow and final payoff of A? </P> **b.** What is the initial cost, intermediate cash-flow and final payoff of B? (Let's denote $F_{0,T}$(S) for the price of a prepaid forward contract on S for delivery at T, P(0,T) is the price of a zero-coupon bond paying $1 at T.)

Answer:

Time	**t=0** **Initial Cost**	**t=k** **Intermidiate Cost**	**t=T** **Final Payoff**	**Total Profit**
Portfolio A	-C(K,T) + P(K,T)	Remained Unchanged	Max(S_T-K,0) - Max(K-S_T,0)	S_T-K
Portfolio B	$-F_{0,T}(S) + Ke^{-rT}$	Receive Interest Payment	+S_T-K	S_T-K

c. Using the no-arbitrage principale to get a generic form of the Put-Call parity.

Answer: Acording to no-arbitrage principale:

$$C(K,T) + P(K,T) = S_0 e^{-qT} - Ke^{-rT}$$

or

$$C(K,T) + Ke^{-rT} = P(K,T) + S_0 e^{-qT} (*)$$

Proof:

If the (*) is not equal, assume that:

$C(K,T) + Ke^{-rT} > P(K,T) + S_0e^{-qT}$ then $C(K,T) + Ke^{-rT} - P(K,T) - S_0e^{-qT} > 0$

(**)

We got:
- C(K,T): short a call
- Ke^{-rT}: borrow money
- $-P(K,T)$: long a put
- $-S_0e^{-qT}$: buy stocks

 At the t=T, the final profit should be: $-Max(S_T - K, 0) - K + Max(K - S_T, 0) + (S_T e^{-qT})e^{qT}$
</p>

if $K > S_T$: final profit: 0 - K + K - S_T + S_T = 0

if $K < S_T$: final profit: $-S_T$ + K - K + 0 + S_T = 0
In short, (**) is not true. </P> Furthermore, $C(K,T) + Ke^{-rT} - P(K,T) - S_0e^{-qT} < 0$ also not true (with similar proofing)</p>

Then

$$C(K,T) + Ke^{-rT} = P(K,T) + S_0e^{-qT}$$

d. Suppose the risk-free interest rate is r, provide the Put-Call parity in three specific cases:

i. S pays no dividend;
ii. S pays n discrete dividends d_i at t_i for i ∈ [1, n];
iii. S pays a continuous dividends at the rate q.

Answer:

Time	Put-Call parity
S pays no dividend	C(K,T) + Ke^{-rT} = P(K,T) + S_0
S pays n discrete dividends d_i at t_i for i ∈ [1, n]	C(K,T) + Ke^{-rT} = P(K,T) + S_0 + $\sum_{i=1}^{n}(d_i e^{-rt_i})$
S pays a continuous dividends at the rate q	C(K,T) + Ke^{-rT} = P(K,T) + S_0e^{-qT}

In [1]:
```
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Question 2

Let’s first recall that the Put-Call parity for a stock paying dividends </p>

C(K,T) + Ke^{-rT} = P(K,T) + $S_0 e^{-qT}$, q : dividend yield over (0, T).

The **Implied Dividend** yield is the value of q such that the Put-Call parity holds true. </p>

$$\text{IDIV(K,T)} = -\frac{1}{T} log \frac{C(K,T) - P(K,T) + Ke^{-rT}}{S}$$

a. Get prices of AAPL option on 01/06/2017 from the data file.

In [24]:
```
data = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vSRm
PANey5HKtRoWSPMbvhHrUQfsn92Osf1VMqwQxx3PO5r7KD9_VC2qb5Nz9KC37_VoXUMhLLoP
LaN/pub?gid=1995458478&single=true&output=csv')
data.head()
```

Out[24]:

	Symbol	ExpirationDate	AskPrice	AskSize	BidPrice	BidSize	LastPrice	PutCall	Strik
0	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.04	put	122.
1	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.09	put	109.
2	AAPL	06/02/17	0.23	NaN	0.19	NaN	0.23	put	152.
3	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.06	put	118.
4	AAPL	06/02/17	4.55	NaN	4.10	NaN	4.50	put	157.

In [25]:
```
data.dtypes
```

Out[25]:
```
Symbol                 object
ExpirationDate         object
AskPrice              float64
AskSize               float64
BidPrice              float64
BidSize               float64
LastPrice             float64
PutCall                object
StrikePrice           float64
Volume                  int64
ImpliedVolatility     float64
Delta                 float64
Gamma                 float64
Vega                  float64
Rho                   float64
OpenInterest            int64
UnderlyingPrice       float64
DataDate               object
dtype: object
```

In [26]:
```
## the whole dataset is about APPL
data.AskPrice
```

```
Out[26]: 0         0.02
         1         0.02
         2         0.23
         3         0.02
         4         4.55
         5         0.02
         6         2.03
         7        12.25
         8        22.20
         9         7.10
         10       27.25
         11        9.75
         12       14.75
         13       37.10
         14       42.10
         15       47.20
         16        0.02
         17        0.02
         18        0.14
         19        0.31
         20        0.05
         21       19.75
         22        0.22
         23        2.57
         24        7.25
         25       12.25
         26        0.12
         27        0.02
         28        0.04
         29        0.07
                  ...
         1830      0.01
         1831      0.01
         1832      0.02
         1833      0.04
         1834      0.02
         1835      0.04
         1836      0.03
         1837      0.01
         1838      0.01
         1839      0.03
         1840      0.04
         1841      0.06
         1842      0.04
         1843      0.04
         1844      0.05
         1845      0.06
         1846      0.07
         1847      0.04
         1848      0.03
         1849      0.04
         1850      0.06
         1851      0.05
         1852      0.06
         1853      0.10
         1854      0.07
         1855      0.04
```

```
1856     0.16
1857     0.14
1858     0.16
1859     0.15
Name: AskPrice, Length: 1860, dtype: float64
```

b. Get risk-free interst rate for the same date from [this link](https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=billrates).

In [27]:
```
r1y = 0.0116
```

c. Compute the implied dividend for different maturity T, use the Actual/360 day convention - [see this page](https://wiki.treasurers.org/wiki/Day_count_conventions).

In [28]:
```
data['Timetomaturitydays'] = pd.to_datetime(data['ExpirationDate']) - pd
.to_datetime(data['DataDate'])
data['Timetomaturitydays'] = [d.days for d in data['Timetomaturitydays'
]]
data['TMMActual360'] = data['Timetomaturitydays']/360
data['AVGPrice']=(data['AskPrice']+data['BidPrice'])/2
data.head()
```

Out[28]:

	Symbol	ExpirationDate	AskPrice	AskSize	BidPrice	BidSize	LastPrice	PutCall	Strik
0	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.04	put	122.
1	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.09	put	109.
2	AAPL	06/02/17	0.23	NaN	0.19	NaN	0.23	put	152.
3	AAPL	06/02/17	0.02	NaN	0.00	NaN	0.06	put	118.
4	AAPL	06/02/17	4.55	NaN	4.10	NaN	4.50	put	157.

5 rows × 21 columns

In [29]:
```
## Merge dataset based on Put Call
IDIVdf=pd.merge(data[data.PutCall=='call'],data[data.PutCall=='put'], ho
w='inner',on=["StrikePrice","TMMActual360","UnderlyingPrice"])
IDIVdf.head(10)
```

Out[29]:

	Symbol_x	ExpirationDate_x	AskPrice_x	AskSize_x	BidPrice_x	BidSize_x	LastPrice_
0	AAPL	06/02/17	32.45	NaN	31.80	NaN	32.50
1	AAPL	06/02/17	38.45	NaN	37.80	NaN	37.31
2	AAPL	06/02/17	12.45	NaN	11.75	NaN	13.04
3	AAPL	06/02/17	33.45	NaN	32.80	NaN	33.39
4	AAPL	06/02/17	13.40	NaN	12.90	NaN	12.91
5	AAPL	06/02/17	17.45	NaN	16.80	NaN	19.83
6	AAPL	06/02/17	42.45	NaN	41.80	NaN	35.93
7	AAPL	06/02/17	23.40	NaN	22.90	NaN	22.52
8	AAPL	06/02/17	4.40	NaN	3.95	NaN	3.95
9	AAPL	06/02/17	31.45	NaN	30.80	NaN	33.51

10 rows × 39 columns

In [30]:
```
IDIVdf.dtypes
```

Out[30]:
```
Symbol_x                   object
ExpirationDate_x           object
AskPrice_x                float64
AskSize_x                 float64
BidPrice_x                float64
BidSize_x                 float64
LastPrice_x               float64
PutCall_x                  object
StrikePrice               float64
Volume_x                    int64
ImpliedVolatility_x       float64
Delta_x                   float64
Gamma_x                   float64
Vega_x                    float64
Rho_x                     float64
OpenInterest_x              int64
UnderlyingPrice           float64
DataDate_x                 object
Timetomaturitydays_x        int64
TMMActual360              float64
AVGPrice_x                float64
Symbol_y                   object
ExpirationDate_y           object
AskPrice_y                float64
AskSize_y                 float64
BidPrice_y                float64
BidSize_y                 float64
LastPrice_y               float64
PutCall_y                  object
Volume_y                    int64
ImpliedVolatility_y       float64
Delta_y                   float64
Gamma_y                   float64
Vega_y                    float64
Rho_y                     float64
OpenInterest_y              int64
DataDate_y                 object
Timetomaturitydays_y        int64
AVGPrice_y                float64
dtype: object
```

In [31]:
```
## set up IDIV parameter
IDIVdf['CallPrice']=(IDIVdf.AskPrice_x+IDIVdf.BidPrice_x)/2
IDIVdf['PutPrice']=(IDIVdf.AskPrice_y+IDIVdf.BidPrice_y)/2
IDIVdf['ert']=np.exp(-r1y*IDIVdf["TMMActual360"])
IDIVdf.head()
```

Out[31]:

	Symbol_x	ExpirationDate_x	AskPrice_x	AskSize_x	BidPrice_x	BidSize_x	LastPrice_:
0	AAPL	06/02/17	32.45	NaN	31.80	NaN	32.50
1	AAPL	06/02/17	38.45	NaN	37.80	NaN	37.31
2	AAPL	06/02/17	12.45	NaN	11.75	NaN	13.04
3	AAPL	06/02/17	33.45	NaN	32.80	NaN	33.39
4	AAPL	06/02/17	13.40	NaN	12.90	NaN	12.91

5 rows × 42 columns

In [37]:
```
## IDIV computation
IDIVdf["ID"]=-(1/IDIVdf["TMMActual360"])*np.log((IDIVdf.CallPrice-IDIVdf
.PutPrice+IDIVdf.StrikePrice*IDIVdf.ert)/IDIVdf.UnderlyingPrice)
IDIVdf.head(10)
```

Out[37]:

	Symbol_x	ExpirationDate_x	AskPrice_x	AskSize_x	BidPrice_x	BidSize_x	LastPrice_:
0	AAPL	06/02/17	32.45	NaN	31.80	NaN	32.50
1	AAPL	06/02/17	38.45	NaN	37.80	NaN	37.31
2	AAPL	06/02/17	12.45	NaN	11.75	NaN	13.04
3	AAPL	06/02/17	33.45	NaN	32.80	NaN	33.39
4	AAPL	06/02/17	13.40	NaN	12.90	NaN	12.91
5	AAPL	06/02/17	17.45	NaN	16.80	NaN	19.83
6	AAPL	06/02/17	42.45	NaN	41.80	NaN	35.93
7	AAPL	06/02/17	23.40	NaN	22.90	NaN	22.52
8	AAPL	06/02/17	4.40	NaN	3.95	NaN	3.95
9	AAPL	06/02/17	31.45	NaN	30.80	NaN	33.51

10 rows × 43 columns

In [40]:
```
IDIVdf['ID']
```

```
Out[40]: 0      0.161961
         1      0.161506
         2      0.210504
         3      0.161885
         4      0.092870
         5      0.151341
         6      0.161203
         7      0.092112
         8      0.058292
         9      0.162037
         10     0.210049
         11     0.034257
         12     0.210656
         13     0.105571
         14     0.161809
         15     0.151493
         16     0.102049
         17     0.162112
         18     0.034484
         19     0.151568
         20     0.209746
         21     0.209898
         22     0.210201
         23     0.151190
         24     0.209822
         25     0.211186
         26     0.210731
         27     0.102352
         28     0.454772
         29     0.034560
                  ...
         900    0.029124
         901    0.025013
         902    0.030424
         903    0.024634
         904    0.031940
         905    0.152781
         906    0.028660
         907    0.043595
         908    0.029503
         909    0.021505
         910    0.021395
         911    0.017091
         912    0.018558
         913    0.020006
         914    0.018172
         915    0.015202
         916    0.018755
         917    0.020530
         918    0.021694
         919    0.018724
         920    0.017896
         921    0.018444
         922    0.019355
         923    0.019737
         924    0.019573
         925    0.022533
```

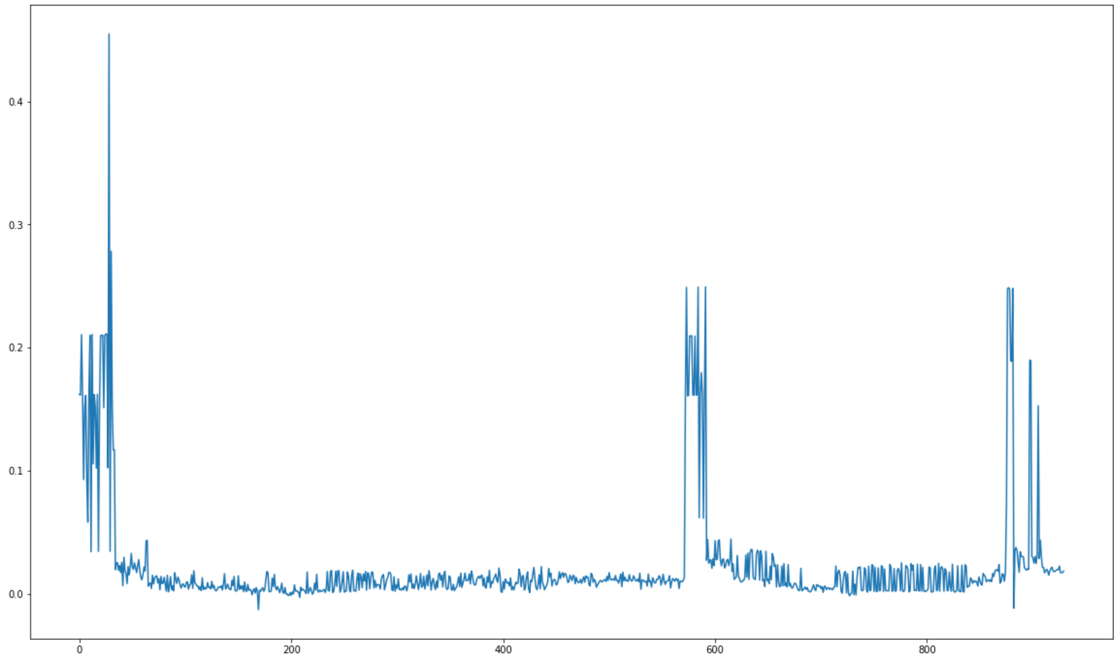
```
926    0.017245
927    0.017204
928    0.017371
929    0.018433
Name: ID, Length: 930, dtype: float64
```

In [42]:
```
IDIVdf['ID'].describe()
```

Out[42]:
```
count    930.000000
mean       0.022303
std        0.044704
min       -0.012896
25%        0.004804
50%        0.010768
75%        0.018361
max        0.454772
Name: ID, dtype: float64
```

d. Compare the IDIV with the historical dividends from [this page] (https://finance.yahoo.com/quote/AAPL).

In [65]:
```
fig, ax = plt.subplots(1,1,figsize=(20,12))
plt.plot(IDIVdf['ID'])
plt.show()
```

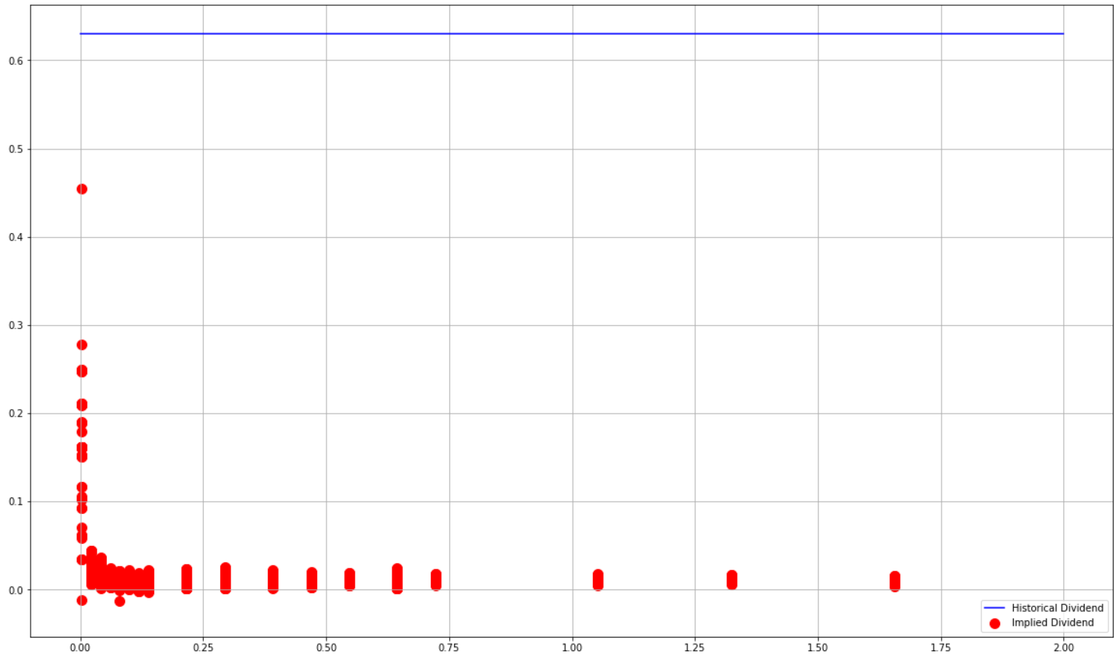


In [46]:
```
historicaldiv=0.63
```

In [64]:
```
fig, ax = plt.subplots(1,1,figsize=(20,12))

plt.scatter(IDIVdf['TMMActual360'],IDIVdf['ID'],marker='o',s=100,color=
'r',label='Implied Dividend')
plt.plot([0,2],[historicaldiv,historicaldiv],color='b',label='Historical
 Dividend')

plt.legend()
plt.grid(True)
plt.show()
```



Question 3

(Model risk in binomial tree framework) We re-consider the binomial tree model, in which the price at t = 1 has the following dynamics: </p>

$$\begin{cases} S_u &= uS_0 \text{ with probability p} \\ S_d &= dS_0 \text{ with probability 1-p} \end{cases}$$

Let's suppose the risk-free rate is constant, thus from B 0 = 1 we have B 1 = 1 + r for an investment in the bank account.

a. Verify that in order to exclude the opportunity arbitrage, one must have

d < 1 + r < u.

At t = 0 we sell a derivative with payoff g(S_1) at the price g_0. We would like to hedge our position in setting a self-financing strategy. The strategy consists in holding Δ units of S and investing the rest in the bank account B.

Answer:

To solve this problem, we assume there would be two cases:

+ d > 1 + r
+ 1 + r >u

Time t	**d > 1 + r**	**1 + r >u**
t = 0	- Borrow S_0 @ r - Long stock stocks @ S_0	- Short stocks for S_0 - Deposit amout S_0 @ r
t = 1	- Sell Stock get dS_0 - Pay back $S_0(1+r)$	- Widraw deposit as $S_0(1+r)$ - Buy stock stocks @ uS_0
Conclusion	$dS_0 > S_0(1+r)$ → It agaist the arbitrage rule as we can make profit without initial investment	$S_0(1+r) > uS_0$ → It agaist the arbitrage rule as we can make profit without initial investment

In short, in order to exclude the opportunity arbitrage, one must have d < 1 + r < u

b. Show that Δ and the price of this derivative are given by:

$$\Delta = \frac{g(S_u)-g(S_d)}{S_u-S_d}$$

$$g_0 = \frac{qg_u+(1-q)g_d}{1+r} \text{ where } q = \frac{(1+r)-d}{u-d}$$

Now instead of supposing S 1 takes only two values, we relax this assumption: without knowing the exact dynamics of S 1 , we only know with probability 1:

$$S_1 \in [Sd, Su].$$

We suppose further that the payoff function is convex, i.e., for y_d ≤ y ≤ y_u,

$$g(y_d) + \frac{g(y_u)-g(y_d)}{y_u-y_d}(y-y_d) \geq g(y_d)$$

Answer:

We got:

Time t	Stocks go up	Stocks go down
t = 0	$\Delta S_0 - g_0 + B_0$	$\Delta S_0 - g_0 + B_0$
t = 1	$\Delta S_0 u - g_u + B_0(1+r)$	$\Delta S_0 d - g_d + B_0(1+r)$

In order to hedge our position, we got: </p>

$\Delta S_0 u - g_u + (1+r)$ = $\Delta S_0 d - g_d + (1+r)$ (as $B_0 = 1$)

$\Leftrightarrow$ ΔS_0 (u-d) = $g_u + g_d$
$\Leftrightarrow$ $\Delta = \frac{g_u + g_d}{S_0(u-d)}$ = $\frac{g(S_u) - g(S_d)}{S_u - S_d}$

Next, as the portfolio is riskless, we can discounted back the asset at time t=1 back to t=0 and both need to be equal as below: </p>

$\frac{\Delta S_0 u - g_u + B_0(1+r)}{1+r} = \Delta S_0 - g_0 + B_0$

$\Leftrightarrow g_0 = \Delta S_0 + 1 - \frac{\Delta S_0 u - g_u + (1+r)}{1+r}$ (as B_0=1)

$\Leftrightarrow g_0 = \Delta S_0 - \frac{\Delta S_0 u - g_u}{1+r}$

$\Leftrightarrow g_0 = \frac{(1+r)\Delta S_0 - \Delta S_0 u + g_u}{r+1}$

$\Leftrightarrow g_0 = \frac{(1+r-u)\Delta S_0 + g_u}{r+1}$

$\Leftrightarrow g_0 = \frac{1}{r+1}((1+r-u)\Delta S_0 + g_u)$

we got: $\Delta = \frac{g_u + g_d}{S_0(u-d)}$ = $\frac{g(S_u) - g(S_d)}{S_u - S_d}$

then:

$\Leftrightarrow g_0 = \frac{1}{r+1}((1+r-u)\frac{g_u + g_d}{S_0(u-d)}S_0 + g_u)$

$\Leftrightarrow g_0 = \frac{1}{r+1}((1+r-u)\frac{g_u + g_d}{(u-d)} + g_u)$

$\Leftrightarrow g_0 = \frac{1}{r+1}(\frac{(1+r-u)(g_u + g_d) + g_u(u-d)}{(u-d)})$

$\Leftrightarrow g_0 = \frac{1}{r+1}(\frac{g_u + g_u r - g_u u + g_d + g_d r - g_d u + g_u u - g_u d}{(u-d)})$

$\Leftrightarrow g_0 = \frac{1}{r+1}(\frac{g_u(1+r-d) + g_d(1+r-u)}{(u-d)})$

$\Leftrightarrow g_0 = \frac{1}{r+1}(\frac{g_u(1+r-d)}{(u-d)} + \frac{g_d(1+r-u)}{(u-d)})$

Let $q = \frac{(1+r-d)}{(u-d)}$ then $1 - q = \frac{(1+r-u)}{(u-d)}$ </p> Finally we got: $g_0 = \frac{qg_u + (1-q)g_d}{1+r}$ where $q = \frac{(1+r)-d}{u-d}$

c. We keep using the same hedging strategy as in b., what is the PnL of the hedging strategy in the new model?

Answer:

We got PnL as follow:

Time t	Portfolio
t = 0	ΔS_0 - g_0 + B_0
	$= \frac{g(S_u)-g(S_d)}{S_u-S_d} S_0$ - $\frac{qg_u+(1-q)g_d}{1+r}$ + B_0
t = 1	ΔS_1 - g_y + $B_0(1+r)$
PnL @ t=1	$= \frac{g(S_u)-g(S_d)}{S_u-S_d} S_1$ - $g(y_d)$ - $\frac{g(y_u)-g(y_d)}{y_u-y_d}(y - y_d)$ + $B_0(1+r)$

d. Show that with the same price g_0 as in b., we have a positive PnL with probability 1.

Answer:

e. (Optional) Generalize this problem in multi-period setting: give 0 = $t_0 < t_1 < \cdots < t_N = T, t_n = nT/N = nh$, with probability 1

$$S_{(n+1)h} \in [dS_{nh}, uS_{nh}], d < 1 + r = e^{\rho h} < u.$$

(Hint: Δ should be the first derivative of the price with respect to the underlying.)