Groupon Sep / 17/2024

The French Vanilla Option

A Structured Products by Jayson Barnicle

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        from sklearn.linear_model import
        from sklearn.metrics import
        %matplotlib inline
        from sklearn.ensemble import
```

I. Simulation

To get an idea for what can needed means to our computations.
We'll generate an stock on one year, with a zero and then generate another year.

```
In [2]: np.random.seed(42)
        n_steps = 252
        ...
In [3]: plot_paths(paths[0:0+1,:],0,S0,strike,T,r,b,sigma,mu,mkt);
        ...
        ...
        ...
```

```
    out_put.write(myline)
    out_put.write('\n')
    out_put.close()

print(limits(_iv,_load))
```

```
{'a': 6, 'b': 24, 'H1': 130.0, 'H2': 70.0}
```

B. Sensitivity Test Suite

a. Base Case

In [11]:
```
store_trial(100,100,0.05,0,0.3,1,2000,40000)
```

b. Starting Stock price change

In [10]:
```
# 2% increase
store_trial(102,100,0.05,0,0.3,1,2000,40000)
# 4% increase
store_trial(104,100,0.05,0,0.3,1,2000,40000)
# 8% increase
store_trial(108,100,0.05,0,0.3,1,2000,40000)
```

c. Risk-free rate change

In [9]:
```
# 2% increase
old_r=0.05
store_trial(100,100,old_r*1.02,0,0.3,1,2000,40000)
# 4% increase
store_trial(100,100,old_r*1.04,0,0.3,1,2000,40000)
# 8% increase
store_trial(100,100,old_r*1.08,0,0.3,1,2000,40000)
# 20% increase
store_trial(100,100,old_r*1.2,0,0.3,1,2000,40000)
```

d. Dividend Yield Change

In [8]:
```
comp_r = 0.02
# comparison case
store_trial(100,100,0.05,comp_r,0.3,1,2000,40000)
# 2% increase
store_trial(100,100,0.05,comp_r*1.02,0.3,1,2000,40000)
# 4% increase
store_trial(100,100,0.05,comp_r*1.04,0.3,1,2000,40000)
# 8% increase
store_trial(100,100,0.05,comp_r*1.08,0.3,1,2000,40000)
```

e. Volatility Change

In [7]:
```
old_vol = 0.3
# 2% increase
store_trial(100,100,0.05,0,old_vol*1.02,1,2000,40000)
# 4% increase
store_trial(100,100,0.05,0,old_vol*1.04,1,2000,40000)
# 8% increase
store_trial(100,100,0.05,0,old_vol*1.08,1,2000,40000)
# 20% increase
store_trial(100,100,0.05,0,old_vol*1.2,1,2000,40000)
```

f. Time to Maturity Change

In [6]:
```
old_T =1
# 25% increase
store_trial(100,100,0.05,0,0.3,old_T*1.25,2000,40000)
# 50% increase
store_trial(100,100,0.05,0,0.3,old_T*1.5,2000,40000)
# 100% increase
store_trial(100,100,0.05,0,0.3,old_T*2,2000,40000)
# 50% decrease
store_trial(100,100,0.05,0,0.3,old_T*0.5,2000,40000)
```

h. Barrier-width change

In [5]:
```
def_limits = {'a':400,'b':1600,'H1':130,'H2':70}
d2d = {'a':def_limits['a'],'b':def_limits['b'],'H1':120,'H2':80}
d1d = {'a':def_limits['a'],'b':def_limits['b'],'H1':110,'H2':90}
d4i = {'a':def_limits['a'],'b':def_limits['b'],'H1':140,'H2':60}
d6i = {'a':def_limits['a'],'b':def_limits['b'],'H1':160,'H2':40}

d5i = {'a':def_limits['a'],'b':def_limits['b'],'H1':150,'H2':50}
d0inc = {'a':def_limits['a'],'b':def_limits['b'],'H1':200,'H2':0}

# 20% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d2d)
# 10% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d1d)
# 40% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d4i)
# 60% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d6i)

# 50% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d5i)
# 100% range
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=d0inc)
```

i. Garrison period change

In [4]:
```
def_limits = {'a':400,'b':1600,'H1':130,'H2':70}
L1090 = {'a':200,'b':1800,'H1':130,'H2':70}
L3070 = {'a':600,'b':1400,'H1':130,'H2':70}
L4060 = {'a':800,'b':1200,'H1':130,'H2':70}
L0199 = {'a':20,'b':1980,'H1':130,'H2':70}
L4951 = {'a':980,'b':1020,'H1':130,'H2':70}
# [10,90]
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L1090)
# [30,70]
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L3070)
# [40,60]
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L4060)
# [1,99]
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L0199)
# [49,50]
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L4951)
```

j. Garrison Period Special Case

$t_a = t_b$

- Centered in the product duration
- At the default position for a
- At the default position for b

In [3]:
```
L5050a = {'a':400,'b':400,'H1':130,'H2':70}
L5050b = {'a':1600,'b':1600,'H1':130,'H2':70}
L5050c = {'a':1000,'b':1000,'H1':130,'H2':70}

# [a,a] --> 20% of Time to maturity
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L5050a)
# [b,b] --> 80% of Time to maturity
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L5050b)
# [c,c] --> 50% of Time to maturity
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=L5050c)
```

k. Garrison Width Special Case

$H_1 = H_2$

This is a *near* impossible case. At time t_a, if $H_1 = H_2$ then

- S_a will either be above both, below both or equal to both
  - If S_a is **not** equal to both, then at time t_a it cannot be both below H_1, and above H_2.
  - Therefore the product ceases to exist.

In [2]:
```
knock_up = {'a':400,'b':1600,'H1':130,'H2':130}
knock_down = {'a':400,'b':1600,'H1':70,'H2':70}
knock_knock = {'a':400,'b':1600,'H1':100,'H2':100}

# H1=H2 and they start above S0
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=knock_up)
# H1=H2 and they start below S0
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=knock_down)
# H1=H2 = S0
store_trial(100,100,0.05,0,0.3,1,2000,40000,lim_replace=knock_knock)
```

A full treatment of the results from testing the product sensitivity can be found in the main report.