Untitled

December 23, 2019

[2]:
```
import numpy as np
```

[3]:
```
y = np.array([6, 5.80, 5.62, 5.46, 5.33])
T = np.array([1, 2, 3, 4, 5])
cb = np.array([6, 8, 106, 7, 9])
ch1 = np.array([4, 104, 0, 0, 0])
ch2 = np.array([10, 10, 10, 110, 0])
```

[4]:
```
pb = cb*np.exp(-y*T/100)
ph1 = ch1*np.exp(-y*T/100)
ph2 = ch2*np.exp(-y*T/100)
```

[5]:
```
db = np.sum(pb*T)/np.sum(pb)
dh1 = np.sum(ph1*T)/np.sum(ph1)
dh2 = np.sum(ph2*T)/np.sum(ph2)
```

[6]:
```
qdh2 = -(db*np.sum(pb))/(dh2*np.sum(ph2))
```

[ ]:
```

```

[7]:
```
pold = np.sum( pb + qdh2*ph2 )
pnew = np.sum( (cb + qdh2*ch2) * np.exp(-(y+2)*T/100) )

(pnew-pold)/pold
```

[8]:
```
(pnew-pold)/pold * 10000
```

[9]:
```
pold, pnew
```

[ ]:
```

```

[10]:
```
cob = np.sum(pb*T*T)/np.sum(pb)
coh1 = np.sum(ph1*T*T)/np.sum(ph1)
coh2 = np.sum(ph2*T*T)/np.sum(ph2)
```

[11]:
```

```

```
qc = np.matmul( np.linalg.inv( np.array([[-dh1*np.sum(ph1), -dh2*np.sum(ph2)],␣
↪[coh1*np.sum(ph1), coh2*np.sum(ph2)]]) ),  np.array([[db*np.
↪sum(pb)],[-cob*np.sum(pb)]]))
```

[ ]:

[12]:
```
pold = np.sum( pb + qc[0]*ph1 + qc[1]*ph2 )
pnew = np.sum( (cb + qc[0]*ch1 + qc[1]*ch2) * np.exp(-(y+2)*T/100) )

(pnew-pold)/pold
```

[13]:
```
(pnew-pold)/pold * 10000
```

[14]:
```
pold, pnew
```

[ ]:

[11]:
```
from scipy.stats import norm
import math
```

[25]:
```
1 - norm.cdf(math.log(3) - (3/2))
```

[26]:
```
(1 - norm.cdf(1))**2
```

[27]:
```
(1 - norm.cdf(1))*(1 - norm.cdf(1/math.sqrt(2)))
```

[ ]:

[29]:
```
math.sqrt((math.exp(.4) - 1) * math.exp(-.4) / 40)
```

[30]:
```
norm.cdf( -0.05 / math.sqrt((math.exp(.4) - 1) * math.exp(-.4) / 40) )
```

[32]:
```
.05 - .5*.05*.05
```

[ ]:

[ ]:

[33]:
```
k = 0.86
theta = 0.08
r0 = 0.06
sigma = 0.01
```

[34]:
```
sk2 = (sigma/k)**2
```

[42]:
```
def A(t):
    return t*(theta - sk2/2) + (1/k)*(-theta + 3*sk2/4) + (math.exp(-k*t)/
 ↪k)*(theta - sk2) + (math.exp(-2*k*t)/(2*k))*(sk2/2)

def B(t):
    return (1/k) - (math.exp(-k*t)/k)

def P(t):
    return math.exp(-A(t) -B(t)*r0)

def d1(t0, t1):
    return (1/2) * math.sqrt( sk2 * ((math.exp(-k*t0) - math.exp(-k*t1))**2) *␣
 ↪((math.exp(2*k*t0)-1)/(2*k)))

def d2(t0, t1):
    return -d1(t0, t1)
```

[43]:
```
T1 = 1/2
T0 = 1/4
```

[44]:
```
pcall = P(T1) * ( norm.cdf(d1(T0, T1)) - norm.cdf(d2(T0, T1)) )
```

[45]:
```
pcall
```

[ ]:
```

```

[46]:
```
T0 = 1
T1 = T0 + 1/4
```

[47]:
```
val = (sk2/k) * ( 1 - math.exp(-k*T0) + math.exp(-k*T1) - 1.5*math.
 ↪exp(-k*(T1-T0)) + 0.5*math.exp(-k*(T1+T0)) + 0.5*math.exp(-2*k*(T1-T0)) - 0.
 ↪5*math.exp(-2*k*T1) )
```

[49]:
```
ans = (1/(T1-T0)) * (P(T0)/P(T1)) * (math.exp(val) - 1)
```

[50]:
```
ans
```

[51]:
```
ans * 10000
```

[ ]:
```

```