TP3_MMDFA

February 24, 2021

[14]:
```
# TP 2 - Options américaines (reprendre le TP2 au début)

import numpy as np

n = 100 # nombre d'étapes
T = 1.0 # temps final
deltat = T/n # pas de temps
S0 = 80 # prix initial
sigma = 0.1 # volatilité
up = np.exp(sigma*np.sqrt(deltat)) # up
down = 1/up # down

# taux d'intérêt et facteur d'actualisation
r = 0.01
R = np.exp(r*deltat)

# probabilité risque neutre
p = (R-down)/(up-down)

print("p =",p)
```

```
p = 0.5025001875049342
```

[15]:
```
# matrice des prix de l'actif (TP1)
def CRR(n,down,up,S0):
    S = np.zeros((n+1,n+1))
    S[0,0] = S0
    for i in range(n):
        S[i+1,0] = S[i,0]*down
        for j in range(i+1):
            S[i+1,j+1] = S[i,j]*up
    return S
```

[16]:
```
S = CRR(n,down,up,S0)
S
```

[16]:
```
array([[ 80.        ,   0.        ,   0.        , …,   0.        ,
          0.        ,   0.        ],
```

```
       [ 79.2039867 ,  80.80401337,   0.        , …,   0.        ,
         0.        ,   0.        ],
       [ 78.41589386,  80.        ,  81.6161072 , …,   0.        ,
         0.        ,   0.        ],
       …,
       [ 30.02488791,  30.63143088,  31.25022683, …, 213.15649935,
         0.        ,   0.        ],
       [ 29.72613528,  30.32664305,  30.93928188, …, 211.03555675,
        215.29875779,   0.        ],
       [ 29.43035529,  30.02488791,  30.63143088, …, 208.93571787,
        213.15649935, 217.46254628]])
```

[17]:
```
# paramètres de l'option
K = S0 # strike (ici, option à la "monnaie")

def payoffcall(S,K):
    phicall = max(S-K,0) # option d'achat
    return phicall

def payoffput(S,K):
    phiput = max(K-S,0) # option de vente
    return phiput
```

[18]:
```
# évaluation du prix d'un call européen par récurrence rétrograde
C = np.zeros((n+1,n+1))
for j in range(n+1):
    C[n,j] = payoffcall(S[n,j],K)
for i in range(n-1,-1,-1):
    for j in range(i+1):
        C[i,j] = (p*C[i+1,j+1]+(1-p)*C[i+1,j])/R
```

[19]:
```
print("La prime du contrat call européen vaut C0 =",C[0,0])
```

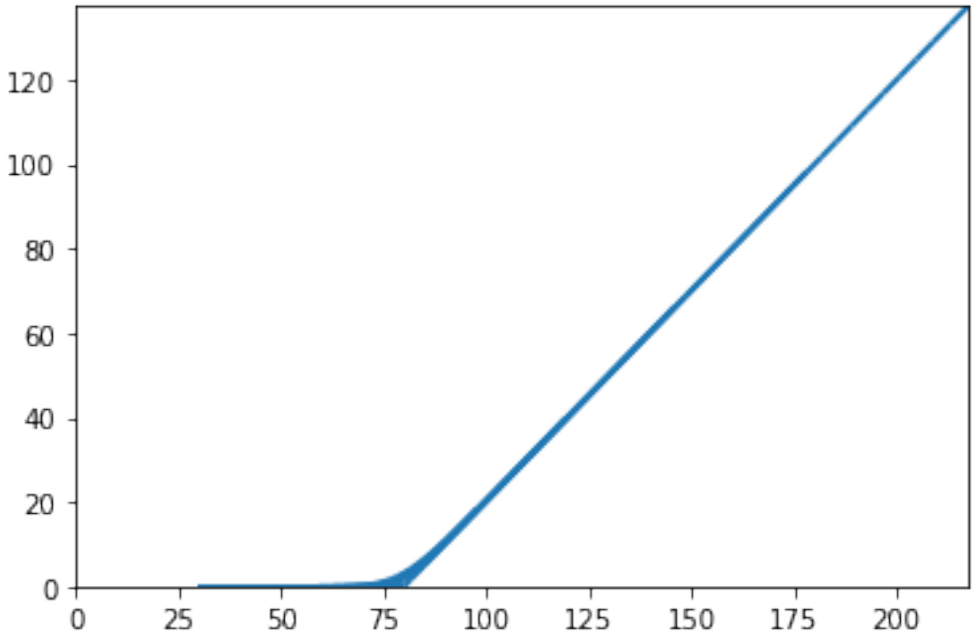
```
La prime du contrat call européen vaut C0 = 3.5802241626860263
```

[20]:
```
# librairies graphiques
import matplotlib.pyplot as plt
import matplotlib.collections as mc

# liste des couples de points
lines = []
for i in range(0,n+1,20):
    for j in range(i):
        lines.append([(S[i,j],C[i,j]),(S[i,j+1],C[i,j+1])])

# plot
lc = mc.LineCollection(lines, cmap=plt.cm.rainbow, linewidths=2)
fig,ax = plt.subplots()
```

```
ax.set_xlim(0,S.max())
ax.set_ylim(0,C.max())
ax.add_collection(lc)
plt.show()
```



[21]:
```
# évaluation du prix d'un call américain par récurrence rétrograde
CA = np.zeros((n+1,n+1))
for j in range(n+1):
    CA[n,j] = payoffcall(S[n,j],K)
for i in range(n-1,-1,-1):
    for j in range(i+1):
        CA[i,j] = max(payoffcall(S[i,j],K),(p*CA[i+1,j+1]+(1-p)*CA[i+1,j])/R)
```

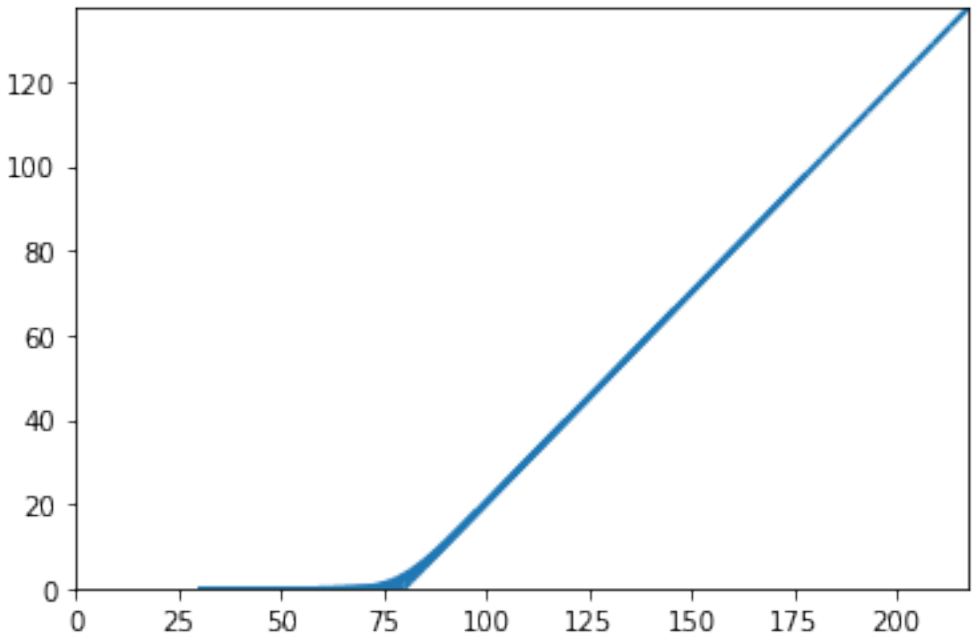
[22]:
```
print("La prime du contrat call américain vaut CA0 =",CA[0,0])
```

```
La prime du contrat call américain vaut CA0 = 3.5802241626860263
```

[23]:
```
# on représente l'option américaine en fonction du prix de l'actif
lines = []
for i in range(0,n+1,20):
    for j in range(i):
        lines.append([(S[i,j],CA[i,j]),(S[i,j+1],CA[i,j+1])])

# plot
lc = mc.LineCollection(lines, cmap=plt.cm.rainbow, linewidths=2)
fig,ax = plt.subplots()
```

```
ax.set_xlim(0,S.max())
ax.set_ylim(0,C.max())
ax.add_collection(lc)
plt.show()
```



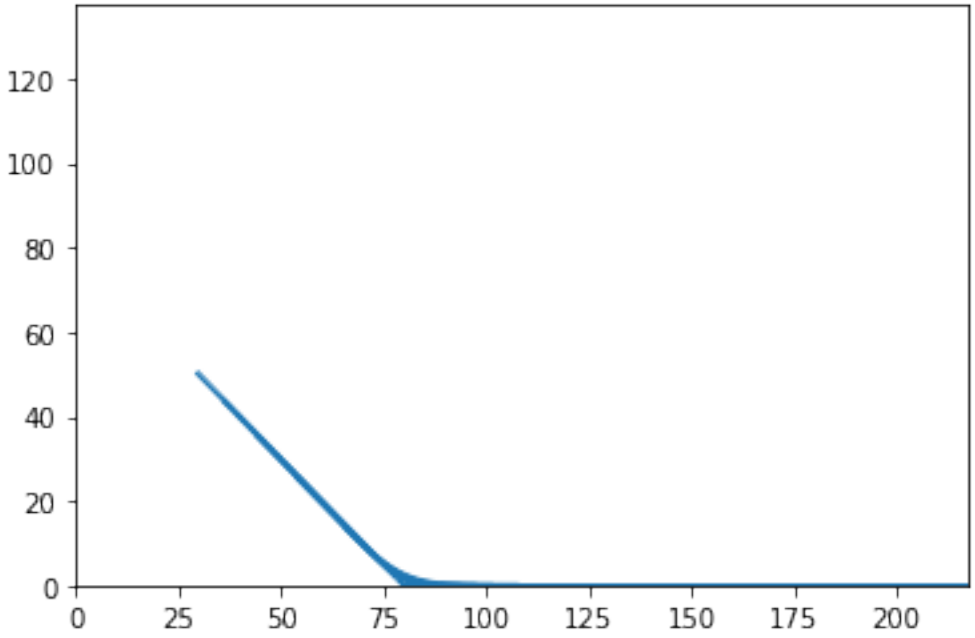
[24]:
```
# put européen
P = np.zeros((n+1,n+1))
for j in range(n+1):
    P[n,j] = payoffput(S[n,j],K)
for i in range(n-1,-1,-1):
    for j in range(i+1):
        P[i,j] = (p*P[i+1,j+1]+(1-p)*P[i+1,j])/R

# put américain
PA = np.zeros((n+1,n+1))
for j in range(n+1):
    PA[n,j] = payoffput(S[n,j],K)
for i in range(n-1,-1,-1):
    for j in range(i+1):
        PA[i,j] = max(payoffput(S[i,j],K),(p*PA[i+1,j+1]+(1-p)*PA[i+1,j])/R)
```

[25]:
```
# on représente le put européen en fonction du prix de l'actif
lines = []
for i in range(0,n+1,20):
    for j in range(i):
```

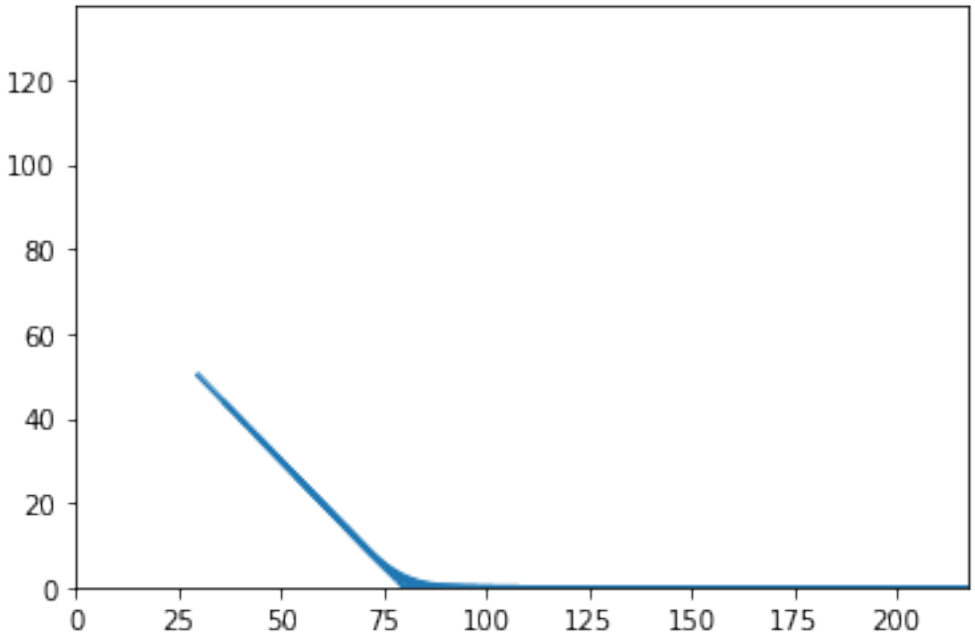
```
        lines.append([(S[i,j],P[i,j]),(S[i,j+1],P[i,j+1])])

# plot
lc = mc.LineCollection(lines, cmap=plt.cm.rainbow, linewidths=2)
fig,ax = plt.subplots()
ax.set_xlim(0,S.max())
ax.set_ylim(0,C.max())
ax.add_collection(lc)
plt.show()
```



[26]:
```
# on représente le put américain en fonction du prix de l'actif
lines = []
for i in range(0,n+1,20):
    for j in range(i):
        lines.append([(S[i,j],PA[i,j]),(S[i,j+1],PA[i,j+1])])

# plot
lc = mc.LineCollection(lines, cmap=plt.cm.rainbow, linewidths=2)
fig,ax = plt.subplots()
ax.set_xlim(0,S.max())
ax.set_ylim(0,C.max())
ax.add_collection(lc)
plt.show()
```



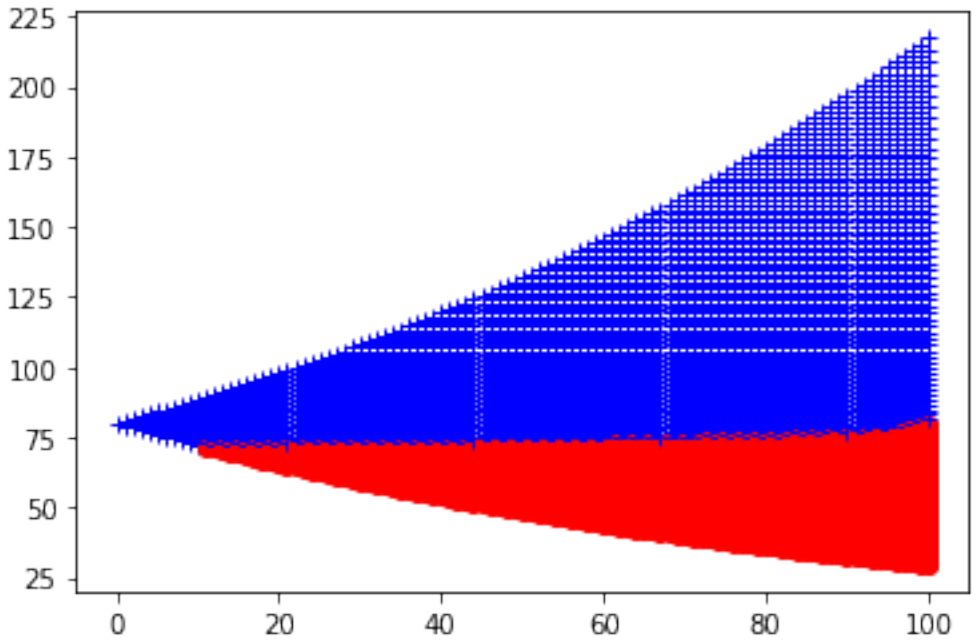
[27]:
```
print('Les primes sont: call européen:',C[0,0],' - call américain:',CA[0,0])
print('Les primes sont: put européen:',P[0,0],' - put américain:',PA[0,0])
```

```
Les primes sont: call européen: 3.5802241626860263  - call américain:
3.5802241626860263
Les primes sont: put européen: 2.7842108626190885  - put américain:
2.852872130458123
```

[28]:
```
# frontière d'exercice du put américain
EPA = np.ones((n+1,n+1)) # 0 si on attend, 1 si on exerce
for j in range(n+1):
    if S[n,j]<=K:
        EPA[n,j]=1
    else:
        EPA[n,j]=-1
for i in range(n-1,-1,-1):
    for j in range(i+1):
        if (p*PA[i+1,j+1]+(1-p)*P[i+1,j])>=payoffput(S[i,j],K):
            EPA[i,j]=-1
        else:
            EPA[i,j]=1
```

[29]:
```
for i in range(n+1):
    for j in range(i+1):
        if EPA[i,j]==1:
            plt.plot(i,S[i,j],'ro')
```

```
        else:
            plt.plot(i,S[i,j],'b+')
```

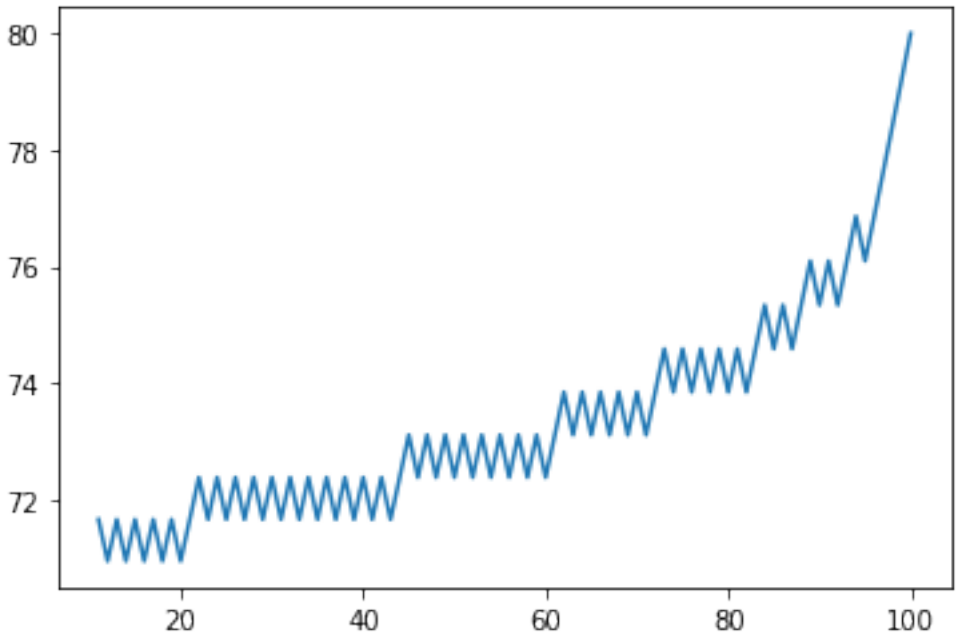


[30]:
```
# recherche de la frontière
F = np.zeros(n+1) # prix correspondant à l'indice maximal de la zone rouge
for i in range(n+1):
    prixrouge = S[i,[j for j in range(n+1) if EPA[i,j]==1]] # prix de la zone␣
 ↪rouge
    if prixrouge.size>0:
        F[i] = np.amax(prixrouge) # on récupère le prix max des rouges
F
```

[30]:
```
array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        , 71.66673082, 70.95363494, 71.66673082, 70.95363494,
       71.66673082, 70.95363494, 71.66673082, 70.95363494, 71.66673082,
       70.95363494, 71.66673082, 72.38699344, 71.66673082, 72.38699344,
       71.66673082, 72.38699344, 71.66673082, 72.38699344, 71.66673082,
       72.38699344, 71.66673082, 72.38699344, 71.66673082, 72.38699344,
       71.66673082, 72.38699344, 71.66673082, 72.38699344, 71.66673082,
       72.38699344, 71.66673082, 72.38699344, 71.66673082, 72.38699344,
       73.11449482, 72.38699344, 73.11449482, 72.38699344, 73.11449482,
       72.38699344, 73.11449482, 72.38699344, 73.11449482, 72.38699344,
       73.11449482, 72.38699344, 73.11449482, 72.38699344, 73.11449482,
       72.38699344, 73.11449482, 73.84930771, 73.11449482, 73.84930771,
```

```
       73.11449482, 73.84930771, 73.11449482, 73.84930771, 73.11449482,
       73.84930771, 73.11449482, 73.84930771, 74.59150559, 73.84930771,
       74.59150559, 73.84930771, 74.59150559, 73.84930771, 74.59150559,
       73.84930771, 74.59150559, 73.84930771, 74.59150559, 75.34116269,
       74.59150559, 75.34116269, 74.59150559, 75.34116269, 76.09835396,
       75.34116269, 76.09835396, 75.34116269, 76.09835396, 76.86315513,
       76.09835396, 76.86315513, 77.63564268, 78.41589386, 79.2039867 ,
       80.        ])
```

[31]:
```
# on dessine la frontière rouge
indice = np.amin(np.nonzero(F)) # indice temporel du début de la zone rouge
plt.plot(range(indice,n+1),F[indice:n+1])
plt.show()
```



[ ]:

[ ]: