PROJET : Pricing d'options asiatiques, réduction de variance et Estimation de la volatilité à partir de données réelles

Ousmane KANE *et Jordan NZEUCHE KOUAM †

April 2020

*Ingénieur Sup'Galilée MACS2, Institut Galilée - Université Sorbonne Paris Nord, email ousmane.98.ok@gmail.com

†Ingénieur Sup'Galilée MACS2, Institut Galilée - Université Sorbonne Paris Nord, email andradejordan1304@gmail.com

Contents

1 Introduction

Dans le cadre de ce projet, nous nous proposons d'étudier le Pricing d'options asiatiques et la reduction de variance ainsi que l'estimation de la volatilité à partir de données réelles. Nous nous sommes servi du langage python pour faire les simulations et justifier nos résulats.
Nous nous mettons sur le modèle de Black-Sholes :

$S_t = S_0 \exp((r - \frac{\sigma^2}{2})t + \sigma W_t)$

Et nous voulons simuler le prix de l'option asiatique $\mathbb{E}[e^{-rT}(\frac{1}{T}\int_0^T S_t\, dt - K)_+]$ par la méthode de Monte-Carlo avec différents schémas, montrer l'importance de la réduction de variance pour le calcul du prix et déterminer la volatilité à partir de données réelles.

2 Les Schémas

Si nous voulons simuler le prix de l'option asiatique par les méthodes de Monte-Carlo, nous devons approximer l'intégrale $\int_0^T S_t\,dt$, où S_t peut être exactement simulé. Nous allons discrétiser l'intervale [0,T] en n parties égales, tels que $\Delta = T/n$ qui sera notre pas et nous définisons alors l'espace temps $t_k = \Delta k$. Notons N le nombre de réalisation de la variable aléatoire que l'on effectue pour approcher la moyenne de Monte-Carlo.

2.1 Méthode de Riemann

C'est la méthode la plus simple pour approcher l'intégrale :

$$\frac{1}{T}\int_0^T S_t\,dt = \frac{1}{n}\sum_{k=0}^{n-1} S_{t_k}$$

Où les S_{t_k} sont calculées à partir des W_{t_k} qui sont faciles à calculer car les accroissements sont des gaussiennes indépendantes centrées tels $W_{t_{k+1}} - W_{t_k}$ suit une loi $N(0, t_{k+1} - t_k)$.

2.2 Méthode de Trapèze

Nous pouvons aussi approcher l'intégrale par la méthode des trapèzes :

$$\frac{1}{T}\int_0^T S_t\,dt = \frac{1}{n}\sum_{k=0}^{n-1}\frac{S_{t_{k+1}} + S_{t_k}}{2} = \frac{1}{n}(\frac{S_0 + S_{t_n}}{2} + \sum_{k=1}^{n-1} S_{t_k})$$

2.3 Code

le code est le suivant pour déterminer le prix d'options asiatiques à l'aide de la méthode de Riemann et la méthodes des trapèzes :

2.3.1 Programme

```
import numpy as np    # math operations
import numpy.random as npr # random
# Pathdependent Asian option BS

def Asian_call_MC_BS(r,S0,sigma,T,K,N,n):

    delta=float(T/n)
    G=npr.normal(0,1,size=(N,n))
    #Log returns
    LR=(r-0.5*sigma**2)*delta+np.sqrt(delta)*sigma*G
    # concatenate with log(S0)
```

```
LR=np.concatenate((np.log(S0)*np.ones((N,1)),LR),axis=1)
# cumsum horizontally (axis=1)
LR=np.cumsum(LR,axis=1)
# take the expo Spath matrix
Spaths=np.exp(LR)

#initial and end p_values
fa = Spaths[:,0]
fb = Spaths[:,n-1]

## Riemann approximation
#remove final time component
Spaths=Spaths[:,0:len(Spaths[0,:])-1]
#print(Spaths)
#take the average over each row
Sbar=np.mean(Spaths,axis=1)
#print(Sbar)
payoff=np.exp(-r*T)*np.maximum(Sbar-K,0) #call function
Asian_MC_price_R=np.mean(payoff)
# 95% C.I
sigma=np.std(payoff) # standard deviation estimator : ecart type de monte_ca
error=1.96*sigma/np.sqrt(N)
CI_up_R=Asian_MC_price_R + error
CI_down_R=Asian_MC_price_R -error

 ## Trapeze approximation
spathsTRAP = Spaths[:,1:len(Spaths[0,:])]
Sbar1 =np.cumsum(spathsTRAP,axis=1)[:,n-2]
SbarTRAP = (Sbar1 + (fa+fb)*0.5)/n
payoff1=np.exp(-r*T)*np.maximum(SbarTRAP-K,0) #call function
Asian_MC_price_T=np.mean(payoff1)
#95% CI SbarTRAP
sigma1=np.std(payoff1) # standard deviation estimator : ecart type de monte_
error_T=1.96*sigma1/np.sqrt(N)
CI_up_T=Asian_MC_price_T + error_T
CI_down_T=Asian_MC_price_T -error_T

return Asian_MC_price_R,CI_up_R,CI_down_R,error,Asian_MC_price_T,CI_up_T,CI_
```

2.3.2 Résulats

```
*******MC_Price with Riemann*****
8.781989765507591 8.96490565918269 8.599073871832491 0.1829158936751
*******MC_Price with Trapeze*****
8.826361534014884 9.010553363562 8.642169704467769 0.1841918295471158
```

Figure 1: Asian Price with Rieman and Trapeze

2.4 Bonus

le but de cette partie est de calculer le prix de l'option asiatique par Monte-carlo avec la méthode de **Simpson** on a :

$$\frac{1}{T}\int_0^T S_t\,dt = \frac{1}{n}\sum_{k=0}^{n-1}\frac{S_{t_k}+4S_{\frac{t_k+t_{k+1}}{2}}+S_{t_{k+1}}}{6} = \frac{1}{n}\left(\frac{S_0+S_{t_n}}{6}+\frac{1}{3}\sum_{k=1}^{n-1}S_{t_k}+\frac{2}{3}\sum_{k=0}^{n-1}S_{\frac{t_k+t_{k+1}}{2}}\right)$$

2.4.1 Programme

```
import numpy as np    # math operations
import numpy.random as npr # random

def Asian_call_MC_BS(r,S0,sigma,T,K,N,n):

    delta=float(T/(2*n))
    G=npr.normal(0,1,size=(N,2*n))
    #Log returns
    LR=(r-0.5*sigma**2)*delta+np.sqrt(delta)*sigma*G
    # concatenate with log(S0)
    LR=np.concatenate((np.log(S0)*np.ones((N,1)),LR),axis=1)
    # cumsum horizontally (axis=1)
    LR=np.cumsum(LR,axis=1)
    # take the expo Spath matrix
    Spaths=np.exp(LR)
    #print(Spaths)
    print(len(Spaths[0,:]))

    spathsSIMP = Spaths[:,1:2*n:2]    #point millieu
    spathsSIMP0 = Spaths[:,0:2*n+1:2] # base matrix

    Spaths = spathsSIMP0 #change new spaths

    ## Trapeze approximation
    fa = Spaths[:,0]
```

```
fb = Spaths[:,n]
spathsTRAP = Spaths[:,1:len(Spaths[0,:])-1]
Sbar1 =np.cumsum(spathsTRAP,axis=1)[:,n-2]

SbarTRAP = (Sbar1 + (fa+fb)*0.5)/n
payoff1=np.exp(-r*T)*np.maximum(SbarTRAP-K,0) #call function
price=np.mean(payoff1)

#simpson approximation
Sbar2 =np.cumsum(spathsSIMP,axis=1)[:,n-1] #somm point milieu
u = float(1/3)
v = float(1/6)
w = float(2/3)
SbarSIMP = (Sbar2*w + Sbar1*u + (fa+fb)*v)/n
payoff2=np.exp(-r*T)*np.maximum(SbarSIMP-K,0) #call function
price_SIMP=np.mean(payoff2)

#riemann approximation
Spaths=Spaths[:,0:len(Spaths[0,:])-1]
Sbar=np.mean(Spaths,axis=1)
payoff=np.exp(-r*T)*np.maximum(Sbar-K,0) #call function
Asian_MC_price=np.mean(payoff)

return Asian_MC_price,price,price_SIMP,CI_up,CI_down,errorTRAP
```

2.4.2 Résulats

```
*MC_Price_Riemann* *MC_Price_Trapeze* *MC_Price_Simpson* :
8.770270504196818 8.8149547138289 8.815834496319658
```

Figure 2: Asian Price with Rieman, Trapeze and Simpson

3 Variables de Contrôle

Dans cette partie, afin d'améliorer l'efficacité de la simulation de Monte Carlo, nous allons procéder à une réduction de variance par des variables de contrôle qui sont $Y = \exp(\frac{1}{T}\int_0^T \log(S_u)\,du)$ et $Z = e^{-rT}(e^{\frac{1}{T}\int_0^T \log(S_u)\,du} - K)_+$ ensuite nous allons montrer la plus efficace entre ces deux dernières.

3.1 Variables de contrôle Y

3.1.1 Montrer que $\mathbb{E}(Y) = S_0 e^{r\frac{T}{2} - \sigma^2 \frac{T}{4}}$

on a : $\mathbb{E}(Y) = \mathbb{E}(e^{\frac{1}{T}\int_0^T \log(S_u)\,du})$ or
$\log(S_t) = \log(S_0) + (r - \frac{\sigma^2}{2})t + \sigma W_t \sim N(\log(S_0) + (r - \frac{\sigma^2}{2})t, \sigma^2 t)$
car $W_t \sim \sqrt{t}G$ ou $G \sim N(0,1)$ mais on ne peux pas conclure que $\int_0^T \log(S_t)\,dt$ est une gaussienne quand $\log(S_u)$ est gaussienne dons nous allons passer plutôt par la formule d'IPP. on a :
$\int_0^T W_u\,du = TW_T - \int_0^T u\,dW_u = \int_0^T (T-u)\,dW_u$ où $(T-u)$ déterministe donc
$\int_0^T (T-u)\,dW_u \sim N(0, \int_0^T (T-u)^2\,du) \sim N(0, \frac{T^3}{3})$ donc $\frac{1}{T}\int_0^T W_u\,du \sim N(0, \frac{T}{3})$
Donc on avait $\frac{1}{T}\int_0^T \log(S_u)\,du = \log(S_0) + (r - \frac{\sigma^2}{2})\frac{T}{2} + \frac{\sigma}{T}\int_0^T W_u\,du)$
$= \log(S_0) + (r - \frac{\sigma^2}{2})\frac{T}{2} + \sigma\frac{T}{3}G$ où $G \sim N(0,1)$
donc $Y = S_0 e^{(r - \frac{\sigma^2}{2})\frac{T}{2}} * e^{\sigma\frac{T}{3}G}$ et avec la transformé de Laplace on aura
$\mathbb{E}(Y) = S_0 e^{(r - \frac{\sigma^2}{2})\frac{T}{2} + \sigma^2\frac{T}{6}} = S_0 e^{r\frac{T}{2} - \frac{\sigma^2 T}{12}}$

3.1.2 Programme

```
import numpy as np    # math operations
import numpy.random as npr # random
import matplotlib.pyplot as plt # plot

# Pathdependent Asian option BS

def Asian_call_MC_BS(r,S0,sigma,T,K,N,n):

    delta=float(T/n)
    G=npr.normal(0,1,size=(N,n))
    #Log returns
    LR=(r-0.5*sigma**2)*delta+np.sqrt(delta)*sigma*G
    # concatenate with log(S0)
    LR=np.concatenate((np.log(S0)*np.ones((N,1)),LR),axis=1)
    # cumsum horizontally (axis=1)
    LR=np.cumsum(LR,axis=1)
    # take the expo Spath matrix
    Spaths=np.exp(LR)
    ## Riemann approximation
    #remove final time component
    LR1=LR[:,0:len(Spaths[0,:])-1] #log
    Spaths=Spaths[:,0:len(Spaths[0,:])-1]
    #take the average over each row
    Sbar1=Sbar=np.mean(LR1,axis=1)
    Sbar=np.mean(Spaths,axis=1)
    prix_vec=np.exp(-r*T)*np.maximum(Sbar-K,0)-np.exp(Sbar1)
    Asian_MC_price=np.mean(prix_vec) + S0*np.exp((r*T/2) - (sigma**2)*T/12)
```

```
    # 95% C.I
    sigma=np.std(prix_vec,ddof=1) # standard deviation estimator
    error=1.96*sigma/np.sqrt(N)
    CI_up=Asian_MC_price + error
    CI_down=Asian_MC_price -error

    return Asian_MC_price,CI_up,CI_down,error

[Asian_MC_price,CI_up,CI_down,error]=Asian_call_MC_BS(0.05,100,0.2,1,95
,10000,100)
print('*******MC_Price*****')
print(Asian_MC_price,CI_up,CI_down,error)
```

3.1.3 Résultats

```
*******MC_Price*****
8.870779215106126 8.950155582500427 8.791402847711824
0.07937636739430111
```

Figure 3: Asian Price avec la variable Y

3.2 Variable de controle Z

3.2.1 Montrer que $\mathbb{E}(Z) = e^{-rT}[-KN(-d) + S_0 e^{(r-\frac{\sigma^2}{6})\frac{T}{2}} N(-d+\sigma\sqrt{\frac{T}{3}})]$

on a:
$\mathbb{E}(Z) = \mathbb{E}(e^{-rT}(e^{\frac{1}{T}\int_0^T \log(S_u)\,du} - K)_+$
$\Rightarrow \mathbb{E}(Z) = e^{-rT}(\mathbb{E}(e^{\frac{1}{T}\int_0^T \log(S_u)\,du}\mathbf{1}_{e^{\frac{1}{T}\int_0^T \log(S_u)\,du}\geqslant K}) - K\mathbb{E}(\mathbf{1}_{e^{\frac{1}{T}\int_0^T \log(S_u)\,du}\geqslant K}))$
$or \left\{e^{\frac{1}{T}\int_0^T \log(S_u)\,du} \geqslant K\right\} = \left\{e^{\frac{1}{T}\int_0^T \log(S_0 e^{(r-\frac{\sigma^2}{)}u+\sigma W_u)\,du} \geqslant K\right\}$
$\Rightarrow \left\{e^{\frac{1}{T}\int_0^T \log(S_u)\,du} \geqslant K\right\} = \left\{\frac{1}{T}\int_0^T \log(S_0 e^{(r-\frac{\sigma^2}{)}u+\sigma W_u)\,du} \geqslant \log(K)\right\}$
Comme $\int_0^T W_u\,du \sim N(0,\frac{T}{3}) = \sqrt{\frac{T}{3}}G$ où $G \sim N(0,1)$
on a: $\left\{e^{\frac{1}{T}\int_0^T \log(S_u)\,du} \geqslant K\right\} = \left\{G \geqslant \frac{1}{\sigma}\sqrt{\frac{T}{3}}[\log(\frac{K}{S_0}) - (r - \frac{sigma^2}{2})\frac{T}{2}]\right\}$

$\Rightarrow \left\{e^{\frac{1}{T}\int_0^T \log(S_u)\,du} \geqslant K\right\} = \{G \geqslant d\} = \{G \leqslant -d\}$ où $d = \frac{1}{\sigma}\sqrt{\frac{T}{3}}\left[\log(\frac{K}{S_0}) - (r - \frac{sigma^2}{2})\frac{T}{2}\right]$
donc $\mathbb{E}(Z) = e^{-rT}(\mathbb{E}(e^{\frac{1}{T}\int_0^T \log(S_u)\,du}\mathbf{1}_{G\geqslant d}) - K\mathbb{E}(\mathbf{1}_{G\leqslant -d}))$
$\Rightarrow \mathbb{E}(Z) = e^{-rT}(S_0 e^{(r-\frac{\sigma^2}{2})\frac{T}{2}}\mathbb{E}(e^{\frac{\sigma}{T}\int_0^T \log(W_u)\,du}\mathbf{1}_{G\geqslant d}) - K\mathbb{E}(\mathbf{1}_{G\leqslant -d})$
$\Rightarrow \mathbb{E}(Z) = e^{-rT}(\frac{1}{\sqrt{2\pi}}S_0 e^{(r-\frac{\sigma^2}{2})\frac{T}{2}}\int_d^\infty e^{\frac{-x^2}{2}+\sigma\frac{\sqrt{T}}{3}x}\,dx - K\mathbb{E}(\mathbf{1}_{G\leqslant -d})$
$\Rightarrow \mathbb{E}(Z) = e^{-rT}(\frac{1}{\sqrt{2\pi}}S_0 e^{(r-\frac{\sigma^2}{2})\frac{T}{2}}\int_d^\infty e^{-\frac{1}{2}(x-\sigma\sqrt{\frac{T}{3}})^2+\sigma^2\frac{T}{6}}\,dx - K\mathbb{E}(\mathbf{1}_{G\leqslant -d})$
$\Rightarrow \mathbb{E}(Z) = e^{-rT}(\frac{1}{\sqrt{2\pi}}S_0 e^{(r-\frac{\sigma^2}{2})\frac{T}{2}}e^{\sigma^2\frac{T}{6}}\int_{d-\sigma\sqrt{\frac{T}{3}}}^\infty e^{-\frac{1}{2}x^2}\,dx - K\mathbb{E}(\mathbf{1}_{G\leqslant -d})$

$$\Rightarrow \mathbb{E}(Z) = e^{-rT}(\frac{1}{\sqrt{2\pi}} S_0 e^{(r-\frac{\sigma^2}{2})\frac{T}{6}} e^{\sigma^2 \frac{T}{2}} \int_{d-\sigma\sqrt{\frac{T}{3}}}^{\infty} e^{-\frac{1}{2}x^2}\, dx - K\mathbb{E}(\mathbf{1}_{G\leqslant -d})$$

$$\Rightarrow \mathbb{E}(Z) = e^{-rT}(S_0 e^{(r-\frac{\sigma^2}{6}\frac{T}{2})}\mathbb{P}(G \geqslant d - \sigma\sqrt{\frac{T}{3}}) - K\mathbb{P}(G \leqslant -d))$$

$$\Rightarrow \mathbb{E}(Z) = e^{-rT}(S_0 e^{(r-\frac{\sigma^2}{6})\frac{T}{2}} N(-d + \sigma\sqrt{\frac{T}{3}}) - KN(-d))$$

3.2.2 Programme

```
import numpy as np   # math operations
import numpy.random as npr # random
import matplotlib.pyplot as plt # plot
from scipy.stats import norm

def esperanceZ(r,S0,sigma,T,K):

    d=(np.log(K/S0)-(r-sigma**2/2)*T/2)*(1/sigma)*(np.sqrt(3/T))
    E_Z=np.exp(-r*T)*((S0*np.exp((r-sigma**2/6)*T/2)*
    norm.cdf(-d+sigma*np.sqrt(T/3),0,1)-K*norm.cdf(-d,0,1)));

    return E_Z

def Asian_call_MC_BS(r,S0,sigma,T,K,N,n):

    delta=float(T/n)
    G=npr.normal(0,1,size=(N,n))
    #Log returns
    LR=(r-0.5*sigma**2)*delta+np.sqrt(delta)*sigma*G
    # concatenate with log(S0)
    LR=np.concatenate((np.log(S0)*np.ones((N,1)),LR),axis=1)
    # cumsum horizontally (axis=1)
    LR=np.cumsum(LR,axis=1)
    # take the expo Spath matrix
    Spaths=np.exp(LR)

    ## Riemann approximation
    #remove final time component
    LR1=LR[:,0:len(Spaths[0,:])-1]
    Spaths=Spaths[:,0:len(Spaths[0,:])-1]
    #take the average over each row
    Sbar1=Sbar=np.mean(LR1,axis=1)
    Sbar=np.mean(Spaths,axis=1)
    prix_vec=np.exp(-r*T)*np.maximum(Sbar-K,0)-np.exp(-r*T)*
    np.maximum(np.exp(Sbar1)-K,0)
    Asian_MC_price=np.mean(prix_vec) + esperanceZ(r,S0,sigma,T,K)

    # 95% C.I
```

```
    sigma=np.std(prix_vec,ddof=1) # standard deviation estimator
    error=1.96*sigma/np.sqrt(N)
    CI_up=Asian_MC_price + error
    CI_down=Asian_MC_price -error

    return Asian_MC_price,CI_up,CI_down,error

[Asian_MC_price,CI_up,CI_down,error]=Asian_call_MC_BS(0.05,100,0.2,1,95
,10000,100)
print('*******MC_Price*****')

print(Asian_MC_price,CI_up,CI_down,error)
```

3.2.3 Résultats

```
*******MC_Price*****
8.816631477987606 8.823173373447435 8.810089582527777
0.0065418954598297566
```

Figure 4: Asian Price avec la variable Z

3.2.4 Interprétation

On voit que l'erreur a diminué avec la variable Y mais elle a diminué encore plus avec la variable Z. Ces deux variables nous ont donc permis de réduire la variance. L'intervalle de confiance se "rétrécit" et on se rapproche encore plus du vrai prix qui est 8.82.

4 Calcul de la Volatilité de l'actif SP500

nous allons télécharger la valeur de l'actif a la fermeture entre la date 15/09/01 et 16/11/01 et nous allons calculer la valeur de la volatilité de cet actif

4.1 Formule

$$\sigma = \sqrt{\frac{\sum_{k=0}^{n-1}(S_{t_{k+1}} - S_{t_k})^2}{\frac{T}{n}\sum_{k=0}^{n-1} S_{t_k}^2}}$$

4.2 code

```
import numpy as np    # math operations

import numpy.random as npr # random
import datetime as dt
```

```
import matplotlib.pyplot as plt # plot
from pandas_datareader import DataReader
import pandas as pd

tickers = ['SPY']
portfolio_returns = pd.DataFrame()
start, end = dt.datetime(2015,9,1), dt.datetime(2016,11,1)
portfolio_returns = DataReader(tickers,'yahoo',start,end)
portfolio_returns = portfolio_returns.loc[:,'Close']

stkcarre = portfolio_returns * portfolio_returns

stkcarre = stkcarre[:-1]
T = 14/12 #car le temps est en annee
a = stkcarre.mean()*T

deltaStkcarre = portfolio_returns.diff()**2
deltaStkcarre=deltaStkcarre[1:]
b = deltaStkcarre.sum()

vol = np.sqrt(b/a)
print('*Volatilite* :')
print(vol)
```

4.3 Resultats

```
*Volatilite* :
Symbols
SPY    0.142234
dtype: float64
```

Figure 5: Volatilite de l'actif SP500

5 CONCLUSION

Avec le modèle de Black Scholes, nous avons pu pricer l'option asiatique sans réduction de variation avec trois méthodes numériques d'approximation d'intégrale. Cependant on voyait que les prix obtenus n'étaient pas trop proches du vrai prix. Nous avons donc utilisé des variables de controle très efficaces qui nous ont permis de réduire l'intervalle de confiance. Enfin nous avons approcher la volatilité de l'actif SP500 entre la date 01/09/2015 et le 01/11/2016. Ce projet fut très intéressant. Nous avons pu mettre en pratique les notions de calcul stochastique que l'on avait étudiées.