Projet Monte Carlo

Nicolas Caviezel & John Sibony

Le sujet se découpe en 2 parties dans le but d'estimer l'espérance d'une fonction dépendant d'un processus pris en un temps T défini par une équation différentielle stochastique à partir d'un mouvement brownien (appelé processus d'Itô avec un terme de bruit Brownien). Dans le cadre de la matière, nous essayerons non seulement d'estimer cette espérance, via la méthode de Monte Carlo classique (moyenne empirique), et également d'implémenter de nouvelles variables aléatoires par le bais de méthodes du cours ayant une espérance identique à l'initial mais une variance plus faible afin de réduire plus ou moins significativement l'intervalle de confiance de l'espérance réelle demandée.

Dans la première partie, afin de lever le problème de l'EDS nous allons approximer la variable aléatoire du processus par le schéma d'Euler en utilisant une suite récurrence. En découpant l'intervalle [0,T] en pas constant Δ, la suite définie en chaque point $t(k)=k\Delta$ avec $k\in[0,T/\Delta]$ est une approximation du processus pris en en ces points (la suite est définie par l'approximation de la dérivée en prenant en compte la forme différentielle équivalente de l'EDS). Cependant, l'erreur induite (l'écart entre la valeur de la suite et le processus vu en T) peut s'avérer éloigné si le pas constant considéré n'est pas assez petit. Nous noterons donc qu'il s'agira dans cette partie d'estimer l'espérance cherchée à partir d'une estimation même de la variable aléatoire issu du processus d'Itô.
Dans la seconde partie, nous verrons la méthode du schéma Sans Biais reposant sur la même suite récurrente que le schéma d'Euler mais en utilisant cette fois-ci des pas aléatoires sur [0,T] suivant des lois Gamma (comme somme de loi Exponentielle). Suite à de récentes recherches (cf page de Mr.Tan Xiaolu), cette méthode approximative permet en faite de retomber exactement sur l'espérance initiale en prenant l'espérance d'une certaine fonction de cette suite récurrente. L'estimateur de la moyenne empirique de cette fonction sera donc sans biais pour le problème posé.

Nous détaillerons dans un premier temps notre algorithme à l'aide du pseudo code, puis nous analyserons et commenterons nos résultats obtenus (espérance, variance, intervalle de confiance…) à travers les 2 parties à l'aide de graphiques en annexe. Enfin, nous conclurons notre rapport en apportant une analyse sur les 2 méthodes du schéma d'Euler.

I- Pseudo Code

Conventions : - Les fonctions se déclarent de la manière suivante :
typeDuRetour nomDeLaFonction (typeDeL'argument nomDeL'argument)

- Les tableaux unidimensionnels dont les cases sont numérotées de 0 à taille-1 notés : type nomTableau[taille]. On peut rajouter une case à la fin du tableau avec la fonction ajout noté : ajouter(valeur, nomTableau)

- On suppose s'avoir simuler des tableaux unidimensionnels (vecteur) de loi iid Uniforme sur [0,1] ainsi que des lois iid Normale centrée réduite notées :

Simuler Uniforme nomTableau[taille]
Simuler Gaussien nomTableau[taille]

- On connait la fonction de répartition ϕ d'une loi Normale centrée réduite

1) Partie I : Méthode par le schéma d'Euler

- **Déclaration des fonctions**

```
entier K <— 1, T<— 1 n <— 10, ntotal <— 1000000
reel Δ <— T/n, σ <— 0.5                    // déclaration des valeurs du sujet avec ntotal le
                                                nombre des échantillons iid //

reel fonctionMu (reel x)                                  // définition de la fonction Mu //
début
    return 0.1*(sqrt(exp(x)-1)-1/8)
fin

reel fonctionXtilde (reel W[n+1])          // définition de la fonction Xtilde question n°3 //
début
    return (σ*W[n])
fin

reel fonctionPricing (reel x)                             // définition de la fonction g //
début
    return max(exp(x)-K,0)
fin

reel[n+1] brownien ()                        // simuler un vecteur brownien de taille n+1 //
début
    reel W[n+1]                              // n+1 est le nombre de points t_k sur [0,T] //
    W[0]  <— 0
    Simuler Gaussien Z[n]
    pour i allant de 0 à n-1
        W[i+1] <— W[i] + sqrt(Δ)*Z[i]        // méthode forward avec le pas constant Δ //
    return W
fin

reel recurrence(reel W[n+1])              // définition de la récurrence du schéma d'Euler //
début
    reel X[n+1]
    X[0] <— 0
    pour i allant de 0 à n-1
        X[i+1] <— X[i] + fonctionMu(X[i])*Δ + σ(W[i+1]-W[i])
    return X[n]                                // seule la dernière valeur nous intéresse //
fin

vide resultats (reel I, reel S, entier k)          // afficher les résultats de la question n°k //
début
    reel IC [2]                       // intervalle de confiance à 95% sous forme de tableau //
    reel ER                                 // erreur relative maximal à 95% en pourcentage//
```

```
    IC[0] <— I - 1.96*sqrt(S)/sqrt(ntotal)
    IC[1] <— I + 1.96*sqrt(S)/sqrt(ntotal)
    ER <— 100*1.96sqrt(S)/I*sqrt(ntotal)
    Ecrire : « Pour la question »
    Afficher k
    Ecrire : « L'estimation de la moyenne, de la variance, l'intervalle de confiance à 95% et
    l'erreur relative maximal à 95% sont respectivement »
    Afficher : I, S, IC, ER
fin
```

- **Question n°1**

```
reel I <— 0                                                    // estimateur de l'espérance //
reel S <— 0                                                    // estimateur de la variance //
pour i allant de 0 à ntotal-1
    W <— brownien()                  // sur chaque nouvelle boucle les échantillons sont iid //
    I <— I + fonctionPricing(recurrence(W))                    // somme des échantillons iid //
    S <— S + fonctionPricing(recurrence(W))^2
I <— I/ntotal                                                  // définition de l'espérance empirique //
S <— S/ntotal - I^2                          // définition développée de la variance empirique //
resultats(I,S,1)
```

- **Question n°2**

```
I <— 0
S <— 0                                  // on réinitialise les variables pour la question suivante //
pour i allant de 0 à ntotal-1
    W <— brownien()
    I <— I + fonctionPricing(recurrence(W)) + fonctionPricing(recurrence(-W))
    S <— S + (fonctionPricing(recurrence(W)) + fonctionPricing(recurrence(-W)))^2
    // sur une même boucle, les échantillons de la variable W et de la variable antithétique
                                    -W  sont issus du même brownien, ils sont donc corrélés //
I <— I/2*ntotal
S <— S/4*ntotal - I^2
resultats(I,S,2)
```

- **Question n°3**

```
m <— exp((σ^2)/2)*ϕ(- (ln(K)-σ^2)/σ) - K*ϕ(- ln(K)/σ)   // valeur explicite cf TP3 avec μ=0 //
b <— 0                  // on veut estimer la variable de contrôle b qui minimise la variance //
b1 <— 0                                        // estimateur de la covariance entre V_Δ et Xtilde //
b2 <— 0                                                    // estimateur de la variance de Xtilde //
I1 <— 0                                                      // estimateur de la moyenne V_Δ //
I2 <— 0                                                       // estimateur de la moyenne m //
pour i allant de 0 à ntotal-1                              // on calcule I1 et I2 pour trouver b //
    W <— brownien()
    I1 <— I1 + fonctionPricing(recurrence(W))
    I2 <— I2 + fonctionPricing(fonctionXtilde(W)
I1 <— I1/ntotal
I2 <— I2/ntotal
```

```
pour i allant de 0 à ntotal-1                                   // calcul de b=b1/b2 //
    W <— brownien()
    b1 <— (fonctionPricing(recurrence(W)) - I1)(fonctionPricing(fonctionXtilde(W)) - I2)
    b2 <— fonctionPricing(fonctionXtilde(W))^2
b2 <— b2/ntotal - I2^2                          // forme développée de la variance empirique //
b <— b1/b2
I <— 0                                  // on calcule maintenant les estimateurs à l'aide de b //
S <— 0
pour i allant de 0 à ntotal-1
    W <— brownien()
    I <— I + fonctionPricing(recurrence(W)) - b*( fonctionPricing(fonctionXtilde(W)) - m)
    S<—S +(fonctionPricing(recurrence(W))-b*( fonctionPricing(fonctionXtilde(W))- m))^2
I <— I/ntotal
S <— S/ntotal - I^2
resultats(I,S,3)
```

- **<u>Question n°4</u>**

```
reel recurrenceBis(reel W[n+1], reel θ[n])  // on réécrit la récurrence d'Euler en introduisant
                                        notre nouveau vecteur Z de taille n où Z[i] ~ N(θ[i], Δ) //
début
    reel X[n+1]
    X[0] <— 0
    pour i allant de 0 à n-1
        X[i+1] <— X[i] + fonctionMu(X[i])*Δ + σ(W[i+1]-W[i] + θ[i])   // Z[i]=W[i+1]-W[i] + θ[i] //
    return X[n]
fin

reel quotientDensite (reel W[n+1], reel θ[n], boolean a)
                        // on définit le quotient des 2 densités prises avec le vecteur initial des n
                          composantes W[i+1]-W[i] si a==false ou avec le vecteur Z si a==true //
début
    s <— 0
    pour i allant de 0 à n-1
        si (a==true) {
        s <— s - ((W[i+1]-W[i] + θ[i])*θ[i]/Δ) + (θ[i]^2)/2Δ
                    }
        sinon {
        s <— s - ((W[i+1]-W[i])*θ[i]/Δ) + (θ[i]^2)/2Δ
              }
    return exp(s)
fin

reel iemeDerive (int i, reel W[n+1], reel θ[n])    // calcul de la ième dérivée pour l'algorithme
                                                        du gradient stochastique de θ[i] //
début
    return ((fonctionPricing(recurrence(W)))^2) * (-1/Δ)(W[i+1]-W[i] - θ[i])*
    quotientDensite(W, θ, false)              // pour minimiser la variance on utilise le vecteur
                                                        initial des accroissements brownien //
fin
```

```
reel tetaRecurrence (int i, reel θ[n], reel γ, reel W[n+1]) // formule de récurrence de l'algo //
début
    return θ[i] - γ*iemeDerive (i, W[n+1], θ)      // le vecteur θ tend vers l'espérance de Z qui
                                                          minimise la variance //
fin

I <— 0
S <— 0
reel θ[n]
pour i allant de 0 à n-1                          // on initialise les premières valeurs de θ à 0 //
    θ[i] <— 0
pour i allant de 0 à ntotal-1      // on utilise la forme récursive des estimateurs empiriques //
    W <— brownien()
    I <— (i/i+1)*I + (1/i+1) * fonctionPricing(recurrenceBis(W, θ))*quotientDensite(W,
          θ, true)                                  // on utilise maintenant la nouvelle variable Z //
    S <— (i/i+1)*S + (1/i+1) * (fonctionPricing(recurrenceBis(W, θ))*
          quotientDensite(W, θ, true))^2
    pour j allant de 0 à n-1            // on change la valeur des n θ[j] vérifiant la récurrence //
        θ[j] <— tetaRecurrence (j, θ, 1/100*(i+1), W)                    // γI=1/100*I //
S <— S-I^2
resultats(I,S,3)
Ecrire : « Les paramètres qui minimise la nouvelle variance sont : »
afficher θ
```

2) Partie II : Méthode par un schéma Sans Biais

- **Déclaration des fonctions**

```
β <— 0.1        // on fait en sorte qu'en moyenne, les τ(k) soient bien inférieurs à T=1 pour
                                                    obtenir plusieurs pas sur [0,T] //

reel[ ] grilleAleatoire ()
début
    reel t[1]
    t[0] <— 0                                          // on initialise la première valeur de t //
    while t[taille(t) -1]<T {      // tant que la dernière valeur est inférieur à T on peut rajouter
                                                                un pas //
    Simuler Uniforme U[1]   // on simule une loi exponentielle avec la méthode d'inversion //
    ajouter( min(t[taille(t) -1] + (-1/β)*log(U[0]) , T) , t)
                          // la valeur actuelle est la somme de la valeur précédente et d'une loi
                                  exponentielle, si cette valeur actuelle ne dépasse pas T //
                           }
                                  // la boucle se termine donc bien grâce a la fonction min //
return t
fin

reel[n+1] brownienBis (reel t[ ])          // on redéfinit le brownien avec les nouveaux pas //
début
    reel W[taille(t)]                    // la taille du brownien dépendant du nombre de pas //
    W[0]  <— 0
```

```
    Simuler Gaussien Z[n]
    pour i allant de 0 à taille(t)-2
        W[i+1] <— W[i] + sqrt(t[i+1]-t[i])*Z[i]              // méthode forward avec les pas de t //
    return W
fin

reel X[ ] recurrenceTer (reel W[ ], reel t[ ])   // définition de la récurrence du schéma d'Euler
                                                                                    sans biais //
début
    reel X[taille(t)]
    X[0] <— 0
    pour i allant de 0 à taille(t)-2
        X[i+1] <— X[i] + fonctionMu(X[i])*(t[i+1]-t[i]) + σ(W[i+1]-W[i])          // pas aléatoires //
    return X                                        // on stocke toutes les valeurs pour la fonction ψ //
fin

reel fonctionPhi (reel W[ ], reel t[ ])                                  // définition de la fonction ψ //
début
    int NT <— taille(t)-2                                  // numéro de l'avant dernière case de t //
    p <— 1
    X <— recurrenceTer (W, t)
    si (NT>0) {                                        // on s'assure que la boucle ait bien un sens //
    pour i allant de 0 à NT-1
        p <— p*( (fonctionMu(X[i+1])) - (fonctionMu(X[i])))*(W[i+2]-W[i+1])/σ*β*(t[i+2]-t[i+1])
                                // la fonction est bien défini : quand i=NT-1, i+2=NT+1 qui est le
                                                                        numéro de la dernière case de t //
                }
     return p*exp(βT)*( (fonctionPricing(X[NT+1]) - fonctionPricing(X[NT]))
                // l'indicatrice ne sert à rien ici car si NT=0, fonctionPricing(X[NT])=0 car K=1 //
fin
```

- **Question n°1**

```
reel I <— 0
reel S <— 0
pour i avant de 0 à ntotal-1
    t <— grilleAleatoire ()
    W <— brownienBis(t)              // sur chaque nouvelle boucle les échantillons sont iid //
    I <— I + fonctionPhi (W, t)
   S <— S + fonctionPhi (W, t) ^2
I <— I/ntotal
S <— S/ntotal - I^2
resultats(I,S,1)
```

- **Question n°2 : Allocation Proportionnelle**

```
reel factorielle(int k)                                  // fonction factorielle pour la loi de Poisson //
début
    si (k==0) {
```

```
    return 1
                }
    pour i allant de 1 à k
        i <— i*k
    return i
fin

reel probaPoisson(int k)                    // probabilité que l'avant dernier pas N_T =k //
début
    return (exp(-β*T)*(β*T)^k)/factorielle(k)
fin

reel[ ] tableauStrates ()
début
    reel t[1]       // déclaration d'un tableau de taille 1
    t[0] <— probaPoisson(0)*ntotal      // la première case vaut la probabilité que N_T =0 //
    k <— probaPoisson(1)*ntotal
    i <— 1                                          // on est à la case 1 (2ème case) //
    tant que k>0  // le nombre de simulation sur une strate est un entier strictement positif //
        ajouter (k, t)                      // on ajoute une case de valeur k au tableau t //
        k <— probaPoisson(i+1)*ntotal       // on regarde la valeur de la case suivante //
    return t
fin

reel t[ ] triCroissant (reel t[ ])
début
    pour i allant de 0 à taille(t)-2        // i va de la première case à l'avant dernière //
        pour j allant de i+1 à taille(t)-1
            si (t[j]<t[i]) { // la case i prend la valeur minimal parmi toutes les cases suivantes //
            a <— t[j]                                   // on switch les 2 valeurs //
            t[i] <— t[j]
            t[j] <— a
                        }
    return t
fin

reel t[ ] statistiqueOrdre (int k)          // vecteur de statistique d'ordre de taille k //
début
    reel t[k+2]  // tableau de taille k+2 : les extrémités valent 0 et 1 en plus des k variables //
    t[0] <— 0
    t [k+1] <— 1
    si (k==1) {             // l'algorithme de tri marche quand il y a au minimum 2 valeurs //
    t[1] <— Simuler Uniforme u[1]
                }
    si (k>=2) {
    Simuler Uniforme u[k]                       // on simule k loi Uniformes //
    triCroissante (u)
    pour i allant de 1 à k+1        // on complète les valeurs de t en dehors des extrémités //
        t[i] <— u[i]            // ces valeurs forment le vecteur de loi de statistique d'ordre //
                }
    return t
fin
```

```
reel t[ ]
t <— tableauStrates                    // tableau avec le nombre de simulation sur chaque strate //
reel a <— 0                            // somme des lois iid de statistique d'ordre sur chaque strate //
reel b <— 0          // carré de la somme des lois iid de statistique d'ordre sur chaque strate //
reel I <— 0                                                    // estimateur de l'espérance totale//
reel S <— 0                                                    // estimateur de la variance totale //
    pour i allant de 0 à taille(t)-1
        pour j allant de 0 à t[i]-1
            u <— statistiqueOrdre (i)
            W <— brownienBis(u)      // sur chaque nouvelle boucle les échantillons sont iid //
            a <— a + fonctionPhi (W, u)
            b <— b + fonctionPhi (W, u)^2

        S <— S + ( b/t[i] - (a/t[i])^2 )*probaPoisson(i)     // somme sur i des : variances de la
                                        i-ème strate multiplié par la probabilité de la i-ème strate //
        I <— I + a*probaPoisson(i)/t[i]
        a <— 0  // on réinitialise a et b pour les prochaines statistiques d'ordre de taille i+1 //
        b <— 0
    S <— S/ntotal
resultats(I,S,2.1)
```

II- Analyse des résultats

Nous pouvons remarquer dans un premier temps que les estimateurs de l'espérance coïncident pour chaque question dans les 2 parties au centième près (voir **Annexe**). Nos résultats semblent donc correctes (estimateurs consistant) et les variables bien posées. Nous constatons également que l'estimateur de l'espérance de la première partie concorde avec l'estimateur de l'espérance exacte du problème de la seconde partie et donc l'approximation de la méthode d'Euler semble correcte pour estimer le processus prit sur les différents points de [0,T]. Ainsi, contre intuitivement n=10 est ici un pas suffisamment petit pour avoir une bonne approximation du processus.

1) Partie I

Dans la question n°1, nous avons implémenté l'algorithme pour estimer l'espérance en fonction de la variable d'approximation, noté g(rec(W)) où W est un vecteur brownien et « rec » la récurrence sur W permettant de trouver X_T biaisé. Nous avons trouvé que sa variance est de l'ordre de 0,41 au centième près. Nous allons maintenant comparer cette variance à celle des nouvelles variables introduites en se demandant si elle a bien diminuée, dans le but de réduire l'intervalle de confiance asymptotique de l'espérance cherchée.

Dans la question n°2, nous avons introduit une nouvelle variable notée Y=(g(rec(W)) + g(rec(-W)))/2 de variable antithétique -W qui suit bien la même loi que W vu en tant que vecteur gaussien. Donc E(Y)=E(g(rec(W))). D'après le cours, la variance de Y diminue par rapport à la variance de g(W) lorsque cov(g(rec(W)),g(rec(-W))) est négatif (en faisant attention à prendre la même variable W). Ici, la covariance est négative quand la fonction

g o rec est monotone par résultat du cours, ce qui est bien le cas car g est croissante, et la récurrence vu avec le brownien W est monotone en chacune de ses variables (car la fonction μ est croissante) donc g est monotone par composition. Ainsi, la variance de Y diminue est passe à 0,25 au centième près par estimation.

Dans la question n°3, on a posé $Y=g(rec(W)) - b(g(f(W))-m)$ où f est l'application coordonnée qui à un vecteur renvoie la dernière coordonnée multiplié par σ, $m=E(g(f(W)))$, et b est un paramètre de contrôle pour minimiser la variance
Trivialement, on montre que $E(Y)=E(g(rec(W)))$.
On connait explicitement la valeur de m d'après le TP3 1) c) car f(W) suit une loi $N(0,T*\sigma^2)$ par définition du mouvement brownien et T=1. Donc
$m= \exp((\sigma^2)/2)*\phi(- (\ln(K)-\sigma^2)/\sigma) - K*\phi(- \ln(K)/\sigma)$
Pour minimiser la variance par rapport à b, on a démontré que
$b=cov(g(rec(W)), g(f(W)))/Var(g(f(W)))$ était le meilleur contrôle
Ainsi, la variance de Y diminue forcément en dépendant du coefficient de corrélation entre g(rec(W)) et g(f(W)) : plus la corrélation est proche des valeurs extrêmes -1 et 1, plus la variance de Y est petite. Dans notre cas, elle passe à 0,05 au centième près par estimation ce qui prouve que la fonction f choisie est bien posée de façon à rendre le coefficient de corrélation important.

Dans la question n°4, on a posé $Y=g(rec(Z))*f1(Z)/f2_\theta(Z)$ où f1 et $f2_\theta$ sont les densités positives respectivement de l'accroissement du brownien W et et d'un nouveau accroissement brownien Z dont la coordonnée i suit une loi $N(\theta_i,\Delta)$, et les θ_i sont des paramètres de contrôle pour minimiser la variance.
On remarquera que Z et W+θ avec θ le vecteur des espérances sont des vecteurs de même loi. Pour minimiser la variance, d'après le cours on doit avoir
$E(\nabla_\theta(g(rec(W))^2*f1(W)/f2_\theta(W)))=0_{IR^n}$ avec ∇ la matrice jacobienne dérivé en les θ_i
On se retrouve donc avec n équation car il y a n θ_i provenant des n composantes du vecteur Z. On fera attention à appliquer la matrice avec le vecteur W et non pas Z.
Nous avons donc besoin de calculer le quotient des densités $f1(x)/f2_\theta(x)$
Puisque les accroissements sont iid, cette quantité est le produit de densité $N(0,\Delta)$ sur le produit de densité $N(\theta_i,\Delta)$. Un rapide calcul nous mène à :
$f1(x)/f2_\theta(x) = \exp(\sum_{i:1,n} (-x_i*(\theta_i/\Delta)+(\theta_i^2)/2\Delta))$
On voit donc qu'on se retrouve avec n équations qu'on ne peut pas résoudre analytiquement, on utilise donc l'algorithme du gradient stochastique n fois pour estimer le vecteur θ en remarquant que la fonction définie par l'équation est séparante. Nous avons imposé que la suite γ_n vérifie $\gamma_n=1/100*n$ afin qu'elle vérifie les hypothèses de convergence de l'algorithme. La constante 1/100 a été choisie de façon à éviter des problèmes lors de la compilation.
Contrairement aux précédentes méthodes, la méthode de fonction d'importance n'implique pas obligatoirement une baisse de la variance, on aura seulement la plus petite variance de Y existante sans liaison apparente avec la variance initiale. Cependant, à l'aide de nos résultats, la variance de Y diminue bien en passant à 0,14 au centième près par estimation.

Ainsi, pour cette première partie, la méthode de contrôle par variation de la constante à la question n°3 est incontestablement la plus efficace pour réduire l'intervalle de confiance de l'espérance demandée.

2) Partie II

Dans la question n°1, nous avons estimé l'espérance du problème initial posé grâce à la fonction ψ qui permet de retomber sur l'espérance exacte en fonction de la simulation d'un vecteur brownien ayant une variance aléatoire de loi gamma. Nous avons pu estimer la variance de ψ qui vaut 0,45 au centième près. On constatera que bien qu'il s'agisse d'une variable permettant de retrouver l'espérance exacte (non biaisée), sa variance reste supérieure aux variables de la partie I. Nous allons donc de nouveau tenter de diminuer sa variance.

Dans la question n°2, nous avons utilisé la méthode de stratification reposant sur la formule de Bayes. L'énoncé nous conduit à choisir comme loi conditionnelle la loi du vecteur $T=(T_1 \dots T_k)$ sachant $N_T=k$ qui suit une loi de vecteur de statistique d'ordre de taille k, donc en appelant la fonction ψ sur chacun de ces nouveaux vecteurs nous retrouvons la meme espérance de ψ pris avec les pas aléatoire exponentielle. On peut ainsi facilement simuler la loi conditionnelle à l'aide de la simulation d'un vecteur de loi uniforme iid de taille k sur chacune des K strates, et d'un algorithme de tri.
Tout d'abord, nous avons choisi le nombre d'allocation n_k de la statistique d'ordre de façon proportionnelle à la probabilité que $N_T=k$, qui suit la loi (décroissante en k) de poisson de paramètre βT. On a donc fixé premièrement le nombre total de simulation iid (10^5) puis on a déterminé chacune des quantités n_k jusqu'à ce que ce nombre soit égal à 0 (n_k est entier). Ce procédé permet ainsi de restreindre le support IN de la loi de Poisson en un nombre fini K de strate déterminé par le nombre de n_k viable, qui théoriquement aurait du tendre vers l'infini (quand le nombre total de simulation iid tend vers l'infini) pour donner sens à la formule de Bayes. D'après le cours, cette méthode diminue obligatoirement notre variance.
Après calcul, nous trouvons que la variance de cette méthode à allocation proportionnelle vaut 0,35 au centième près. Au vu de ces résultats peu convaincant, nous avons décidé d'utiliser également la méthode de stratification à allocation optimale qui sera forcément meilleure que la précédente, puisqu'elle permet par sa définition de minimiser la variance de la variable de stratification. Pour cette méthode, les quantités n_k vérifient la relation suivante : $n_k = ntotal*(\sigma_k*p_k/\sum_{i:1,K} \sigma_i*p_i)$ avec ntotal le nombre total de simulation fixé, K le nombre de strate déterminer par le fait que n_k doit être un entier plus grand que 1, σ_k l'écart type de la loi conditionnelle, et p_k la probabilité que la loi de poisson valent k. Pour trouver les n_k nous avons donc besoin de connaitre la variance des loi de statistique d'ordre sur chaque strate. Nous les avons calculer par l'estimateur de la variance empirique en simulant pour chaque k, un vecteur de statistique d'ordre de taille k. Nous avons donc pu obtenir le nombre n_k de simulation sur chaque strate, puis de la même façon que pour la méthode à allocation proportionnelle, nous avons pu estimer l'espérance de ψ ainsi que sa variance par la méthode de stratification. Ainsi, nous trouvons une nouvelle variance égale à 0,2 au centième près ce qui est bien mieux que pour l'allocation proportionnelle.

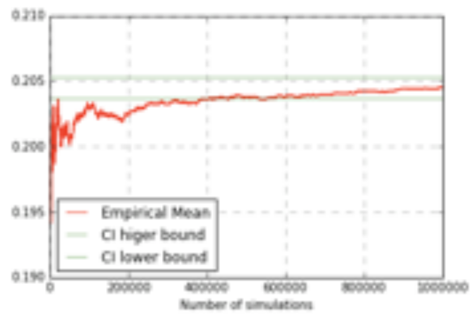
Ainsi, la méthode de contrôle par variation de la constante semble la plus judicieuse pour notre problème. On notera que libre est à l'utilisateur selon ses exigences de diminuer le pas n pour avoir une meilleur approximation (mais toujours biaisée) de l'espérance demandée (quitte à augmenter la complexité de l'algorithme …). D'autre part, selon les préférences de l'utilisateur, le choix de la méthode utilisée varie selon qu'il veuille un estimateur biaisé ou non, et dans ce dernier cas il choisira alors la méthode de

stratification à allocation optimale de la partie II dont la variance empirique de cette nouvelle variable est inférieure à celle de ψ, pour une même espérance non biaisée. Enfin, ces méthodes sont loin d'être exhaustives et il est très certainement possible de diminuer encore un peu plus la variance.

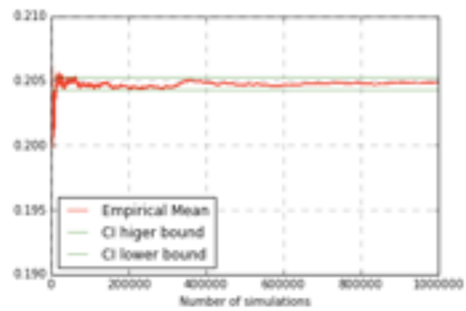
III- Annexe : convergence des estimateurs

1) Partie I

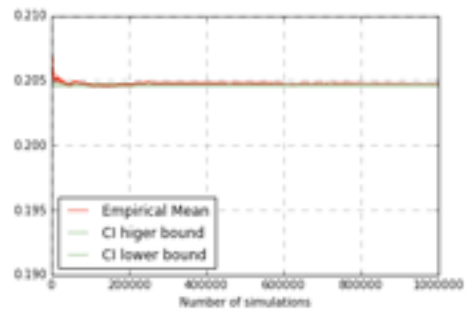
```
In [2]: q1()
Par la méthode de Monte Carlo classique:
Estimateur : 0.204465764808
Variance empirique: 0.414498321047
Intervalle de confiance a 95% : [0.203653348009894658, 0.205278181517450 48]
Erreur relative maximale a 95% : 0.397336302248 %
```



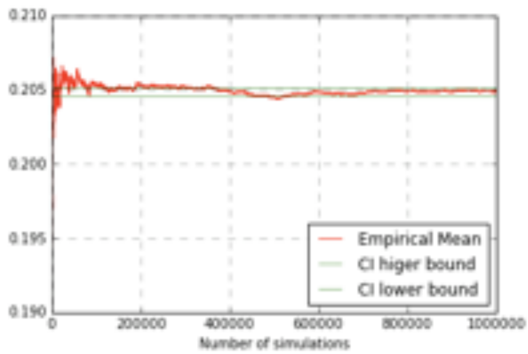

```
In [3]: q2()
Par la méthode de Monte Carlo Antithetique:
Estimateur : 0.204812804432
Variance empirique: 0.25488026417
Intervalle de confiance a 95% : [0.204313278314644 88, 0.20531233055019152]
Erreur relative maximale a 95% : 0.243093988541 %
```




```
In [4]: q3()
b estim 0.041777125602
Par la méthode de Monte Carlo par Variable de Controle:
Estimateur : 0.204718201252
Variance empirique: 0.0474039447884
Intervalle de confiance a 95% : [0.20462528951980247, 0.20481111298337318]
Erreur relative maximale a 95% : 0.0453851837391 %
```

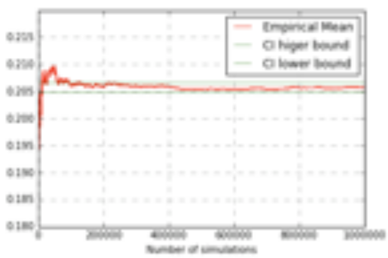



```
In [6]: q4()
Par la méthode de Monte Carlo par fonction d'importance:
Estimateur : [illegible]
Parametre minimiseur de la famille de densité [illegible]
Variance empirique : [illegible]
Intervalle de confiance a 95% : [illegible]
Erreur relative maximale a 95% : [illegible] %
```



2) Partie II

```
In [8]: q1()
Par la méthode de Monte Carlo classique:
Estimateur : 0.205609342318
Variance empirique : 0.448277416708
Intervalle de confiance a 95% : [0.204820718581009985, 0.206577966054593 5]
Erreur relative maximale a 95% : 0.427139780998 %
```




```
In [30]: q2_1()
nombres de strates:  5
Par la méthode de Monte Carlo par stratification proportionnelle:
Estimateur : 0.205819009492
Variance total : 0.354378144004
Intervalle de confiance a 95% : [0.20512442833001912,
0.20651359065402874]
Erreur relative maximale a 95% : 0.337471822509 %
```

```
In [9]: q2_2()
Par la méthode de Monte Carlo par stratification avec Allocation Optimale:
Estimateur : [illegible]
Variance : [illegible]
Les bornes de l intervalle de confiance sont : [illegible]
et [illegible]
La variance totale est [illegible]
La probabilité de tomber dans chaque strates est : [illegible]
[illegible]
```