Project A – Malliavin Calculus & Monte Carlo Method

Tuan Binh NGUYEN & Hai Dang NGO

Part A: Discretization and integration by parts

We consider a risky asset whose dynamic on $[0,T]$ is given by the following equation

$$dX_t^x = rX_t^x dt + \sigma_t X_t^x dB_t \,; X_0^x = x$$

Where $r > 0$ is the risk-free rate, $x > 0$ is the initial condition, $\sigma_t \in L^2([0,T])$ and $(B_t)_{t\in[0,T]}$ the standard Brownian motion defined in the course.

Let $F = \Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)$ where $t_0 = 0 < t_1 < \cdots < t_{n-1} < t_n = T$ and $\Phi \in C^1 \cap Lip(\mathbb{R}^{n+1}, \mathbb{R})$.

We consider

$$P(x) = e^{-rT}\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\right]$$

(a)

From the dynamic

$$dX_t^x = rX_t^x dt + \sigma_t X_t^x dB_t \,; X_0^x = x$$

we have $X_t^x = xe^{rt+\int_0^t \sigma_s dB_s - \frac{1}{2}\int_0^t \sigma_s^2 ds}\ \forall t \in [0,T] \Rightarrow X_t^x \in C^1$ and also:

$$\frac{dX_t^x}{dx} = e^{rt+\int_0^t \sigma_s dB_s - \frac{1}{2}\int_0^t \sigma_s^2 ds} = \frac{X_t^x}{x}$$

Since $\Phi \in C^1 \cap Lip(\mathbb{R}^{n+1}, \mathbb{R})$, there is a sequence $\Phi_n \in C_k^1(\mathbb{R}^2, \mathbb{R})$ that converge to Φ in $L^2 \Rightarrow$ By Lebesgue's differentiation theorem, we have: $P \in C^1$.

Also by Propostion 2 (chainrule) in lecture note:

$$\Delta(x) = \frac{dP(x)}{dx} = e^{-rT}\frac{d}{dx}\,\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\right] = e^{-rT}\mathbb{E}\left[\frac{\mathrm{d}}{\mathrm{dx}}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\right]$$

$$= e^{-rT}\,\mathbb{E}\left[\sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial\Phi}{\partial \mathrm{x_i}}\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\frac{dX_{t_i}^x}{dx}\right]$$

$$= e^{-rT}\,\mathbb{E}\left[\sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial \Phi}{\partial \mathrm{x_i}}\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\frac{X_{t_i}^x}{x}\right] \blacksquare$$

(b)

$$\begin{aligned}
\Delta(x) &= e^{-rT}\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\delta(w)\right] \\
\text{(by def. 4)} \quad &= e^{-rT}\mathbb{E}\left[\langle \mathrm{D_t}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right), w_t\rangle_{L^2([0,T])}\right] \\
&= e^{-rT}\mathbb{E}\left[\int_0^T \mathrm{D_t}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right) w(t)dt\right] \\
\text{(chain rule)} \quad &= e^{-rT}\mathbb{E}\left[\int_0^T \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial}{\partial \mathrm{x_i}}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right) D_t X_{t_i}^x\, w(t)\, dt\right] \\
\text{(prop.9 in lecture note)} \quad &= e^{-rT}\mathbb{E}\left[\int_0^T \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial}{\partial \mathrm{x_i}}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\sigma_t\, X_{t_i}^x w(t)\mathbf{1}_{t\le t_i} dt\right] \\
&= e^{-rT}\mathbb{E}\left[\int_0^{t_i} \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial}{\partial \mathrm{x_i}}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\sigma_t\, X_{t_i}^x w(t)\, dt\right] \\
&= e^{-rT}\mathbb{E}\left[\sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial}{\partial \mathrm{x_i}}\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\int_0^{t_i}\sigma_t\, X_{t_i}^x w(t)\, dt\right]
\end{aligned}$$

Also from a) we have:

$$\Delta(x) = e^{-rT}\,\mathbb{E}\left[\sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\partial \Phi}{\partial \mathrm{x_i}}\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\frac{X_{t_i}^x}{x}\right]$$

So by the definition of conditional expectation:

$$\mathbb{E}\left[\frac{X_{t_i}^x}{x} \mid X_0^x, X_{t_1}^x, \dots, X_T^x\right] = \mathbb{E}\left[\int_0^{t_i}\sigma_t\, X_{t_i}^x w(t)\, dt \;\middle|X_0^x, X_{t_1}^x, \dots, X_T^x\right]$$

$$\Leftrightarrow \mathbb{E}\left[\int_0^{t_i}\sigma_t\, x\, w(t)\, dt \;\middle|X_0^x, X_{t_1}^x, \dots, X_T^x\right] = 1 \;\; \forall\, 0 \le i \le n \;\blacksquare$$

(c)

$\forall 1 < i \le n$ we have:

$$\lambda_i = \frac{\frac{1}{x} - \sum_{j=1}^{i-1}\int_{t_{j-1}}^{t_j}\sigma_t dt\;\lambda_j}{\int_{t_{i-1}}^{t_i}\sigma_t dt}$$

$$\Rightarrow \int_{t_{i-1}}^{t_i} \sigma_t dt\, \lambda_i \;=\; \frac{1}{x} - \sum_{j=1}^{i-1} \int_{t_{j-1}}^{t_j} \sigma_t dt\;\lambda_j$$

$$\Rightarrow \sum_{j=1}^{i} \int_{t_{j-1}}^{t_j} \sigma_t dt\;\lambda_j \;=\; \frac{1}{x}$$

$$\Rightarrow \int_{0}^{t_1} \sigma_t dt\;\lambda_1 + \sum_{j=2}^{i} \int_{t_{j-1}}^{t_j} \sigma_t dt\;\lambda_j \;=\; \frac{1}{x}$$

(from the definition of λ_1)

$$\Rightarrow \frac{\int_0^{t_1} \sigma_t dt}{x \int_0^{t_1} \sigma_t dt} + \sum_{j=2}^{i} \int_{t_{j-1}}^{t_j} \sigma_t dt\;\lambda_j \;=\; \frac{1}{x}$$

$$\Rightarrow \sum_{j=2}^{i} \int_{t_{j-1}}^{t_j} \sigma_t dt\;\lambda_j \;=\; 0 \;\forall\, n \geq i > 1 \qquad (*)$$

(take i = n and change index)

$$\Rightarrow \sum_{i=2}^{n} \int_{t_{i-1}}^{t_i} \sigma_t dt\;\lambda_i \;=\; 0$$

So:

$$\int_0^{t_i} \sigma_t\, x\, w(t)\, dt \;=\; \int_0^{t_i} \sigma_t\, x \sum\nolimits_{\mathrm{i}=1}^{\mathrm{n}} \lambda_{\mathrm{i}} \mathbf{1}_{[t_{i-1},t_i]}(t)\; dt$$

$$= \int_0^{t_i} \sigma_t\, x\, \lambda_1 \mathbf{1}_{[0,t_1]}(t) dt + \int_0^{t_i} \sigma_t\, x \sum_{\mathrm{i}=2}^{\mathrm{n}} \lambda_{\mathrm{i}} \mathbf{1}_{[t_{i-1},t_i]}(t)\; dt$$

$$= \frac{\int_0^{t_1} x\, \sigma_t dt}{x \int_0^{t_1} \sigma_t dt} + \sum_{i=2}^{n} \int_{t_{i-1}}^{t_i} \sigma_t dt\;\lambda_i$$

$$= 1 + 0 = 1$$

We can easily see that $w(t) \in dom(\delta) \subset L^2(\Omega \times [0,T])$ so:

$$\mathbb{E}\left[\int_0^{t_i} \sigma_t\, X_{t_i}^x w(t)\, dt \;\middle|\, X_0^x, X_{t_1}^x, \dots, X_T^x \right] = 1$$

and so (by prop. 5) in lecture note:

$$\begin{aligned}\Pi = \delta(w) &= \int_0^T \sum_{\mathrm{i}=1}^{\mathrm{n}} \lambda_{\mathrm{i}} \mathbf{1}_{[t_{i-1},t_i]}(t)\ dB_t \\ &= \sum_{\mathrm{i}=1}^{\mathrm{n}} \lambda_{\mathrm{i}} \int_0^T \mathbf{1}_{[t_{i-1},t_i]}(t)\, dB_t \\ &= \sum_{\mathrm{i}=1}^{\mathrm{n}} \lambda_{\mathrm{i}} (B_{t_i} - B_{t_{i-1}})\end{aligned}$$

(d)

In Black-Scholes setting we have the constant volatility σ, so $(*)$ becomes:

$$\begin{aligned}&\sigma \sum_{j=2}^{i} \lambda_j \int_{t_{j-1}}^{t_j} dt\ = 0 \\ &\Leftrightarrow \sigma \sum_{j=2}^{i} \lambda_j (t_{ij} - t_{j-1}) = 0\ \forall n \geq j > 1 \\ &\Leftrightarrow \sum_{j=2}^{i} \lambda_j \left(t_{ij} - t_{j-1}\right) = 0\ \forall n \geq j > 1 \\ &\Leftrightarrow \lambda_j = 0\ \forall\ n \geq j > 1\end{aligned}$$

From c) we have:

$$\begin{aligned}\Pi &= \sum_{\mathrm{i}=1}^{\mathrm{n}} \lambda_{\mathrm{i}} (B_{t_i} - B_{t_{i-1}}) \\ &= \lambda_1 B_{t_1} + \sum_{\mathrm{i}=2}^{\mathrm{n}} \lambda_{\mathrm{i}} (B_{t_i} - B_{t_{i-1}}) \\ &= \frac{B_{t_1}}{x\sigma \int_0^{t_1} dt} + 0 \\ &= \frac{B_{t_1}}{x\sigma t_1}\ \blacksquare\end{aligned}$$

(e)

From (6), we have $e^{rT}\Delta(x) = \mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\right]$. By definition of variance, $\forall \Pi \in \mathcal{W}$:

$$Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right) \Pi\right] = \mathbb{E}\left[\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi - e^{rT}\Delta(x)\right)^2\right]$$

$$= \mathbb{E}\left[\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)(\Pi - \Pi_0) + \left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0 - e^{rT}\Delta(x)\right)\right)^2\right]$$

$$= \mathbb{E}\left[\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)(\Pi - \Pi_0)\right)^2\right]$$

$$+ 2\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)(\Pi - \Pi_0)\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0 - e^{rT}\Delta(x)\right)\right]$$

$$+ \mathbb{E}\left[\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0 - e^{rT}\Delta(x)\right)^2\right]$$

$$\geq 0 + Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0\right] +$$

$$2\mathbb{E}\left[\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)(\Pi - \Pi_0)\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0 - e^{rT}\Delta(x)\right) \middle| X_0^x, X_{t_1}^x, \dots, X_T^x\right]\right]$$

$$= Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0\right]$$

$$+ 2\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\left(\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0 - e^{rT}\Delta(x)\right)\underbrace{\mathbb{E}\left[(\Pi - \Pi_0) \middle| X_0^x, X_{t_1}^x, \dots, X_T^x\right]}_{0}\right]$$

$$= Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0\right]$$

So: $Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi_0\right] = \min_{\Pi \in \mathcal{W}} Var\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right]$ ■

(f)

($\Leftarrow$) is obvious since $\Phi \in C^1 \cap Lip(\mathbb{R}, \mathbb{R}) \Rightarrow \Phi$ is bounded & continuous $\Rightarrow \Phi \in C(\mathbb{R}, \mathbb{R})$ with linear growth.

($\Rightarrow$) using sequence approximation:

For an arbitrary function $\Phi \in C(\mathbb{R}, \mathbb{R})$ with linear growth, $\exists\ (\Phi_n)_{n\in\mathbb{N}} \in C^1 \cap Lip(\mathbb{R}, \mathbb{R})$ such that $\lim_{n\to\infty} \Phi_n = \Phi$

$\Rightarrow \forall\ w \in \Omega$:

$$\lim_{n\to\infty} \Phi_n\left(X_0^x(w), X_{t_1}^x(w), \dots, X_T^x(w)\right)\Pi(w) = \Phi\left(X_0^x(w), X_{t_1}^x(w), \dots, X_T^x(w)\right)\Pi(w)$$

or:

$$\lim_{n\to\infty} \Phi_n\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi = \Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi \ \ a.s$$

Since $(\Phi_n)_{n\in\mathbb{N}} \in C^1 \cap Lip(\mathbb{R}, \mathbb{R}) \Rightarrow \Phi_n$ has linear growth, or $\exists\ c > 0$ such that:

$$\left|\Phi_{\mathrm{n}}\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right| \leq c(1 + \ |X_T^x|)\ |\Pi| \qquad (**)$$

And because $X_T^x \in L^2([0,T] \times \Omega)$ and $\Pi \in L^2(\Omega)$, we have the RHS of $(**)$ belongs to $L^2([0,T] \times \Omega)$. By dominated convergence theorem:

$$\lim_{n \to \infty} e^{-rT}\ \mathbb{E}\left[\Phi_{\mathrm{n}}\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right] = \ e^{-rT}\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right] \text{ in } L^2([0,T \times \Omega)$$

Or $g(x) = \ e^{-rT}\mathbb{E}\left[\Phi\left(X_0^x, X_{t_1}^x, \dots, X_T^x\right)\Pi\right]\ \forall \Phi \in C(\mathbb{R}, \mathbb{R})$ with linear growth ■

Part B: Integration by parts and discretization

Let $F = \Phi(\int_0^T X_s^x ds)$ where $t_0 = 0 < t_1 < t_2 < \cdots < t_n = T$ and $\Phi \in C^1 \cap Lip(\mathbb{R}^{n+1}, \mathbb{R})$.

We consider:

$$P(x) = e^{-rT}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\right]$$

(a)

Using the same argument as in part A we have $P(x) \in C^1$ and:

$$\begin{aligned}
\Delta(x) = \frac{dP(x)}{dx} &= e^{-rT}\frac{d}{dx}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\right] \\
\text{(chain rule)} &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\frac{d}{dx}\int_0^T X_s^x ds\right] \\
\text{(prop.6 lecture note)} &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T \frac{dX_s^x}{dx} ds\right] \\
\text{(as calculated in part A.a)} &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T \frac{X_s^x}{x} ds\right] \quad (***)
\end{aligned}$$

Also as in the definition in the question with $\Pi = \delta(w)$ and $w(s) = \frac{2X_s^x}{x\sigma_s \int_0^T X_s^x ds} \in$ $dom(\delta)$ we have:

$$\begin{aligned}
e^{-rT}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\Pi\right] &= e^{-rT}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\delta(\mathrm{w})\right] \\
\text{(def.4 lecture note)} &= e^{-rT}\mathbb{E}\left[\langle D_t\Phi\left(\int_0^T X_s^x ds\right), w_t\rangle\right] \\
\text{(chain rule and prop. 6)} &= e^{-rT}\mathbb{E}\left[\int_0^T \Phi'\left(\int_0^T X_s^x ds\right)\left(\int_0^T D_t X_s^x ds\right) w(t)dt\right] \\
\text{(prop 9 applies into this settings)} &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T \left(\int_0^T \sigma_t X_s^x \mathbf{1}_{t\le s} ds\right) w(t)dt\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T \left(\int_t^T \sigma_t X_s^x ds\right) w(t)dt\right]
\end{aligned}$$

$$
\begin{aligned}
\text{(Fubini's theorem)} \quad &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T X_s^x\left(\int_0^s \sigma_t w(t)dt\right)ds\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T X_s^x\left(\int_0^s \sigma_t \frac{2X_t^x}{x\sigma_t\int_0^T X_t^x dt}dt\right)ds\right] \\
\left(\text{denote}\int_0^s X_t^x dt = u(s)\right) \quad &= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\frac{\int_0^T 2u'(s)u(s)ds}{xu(T)}\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\frac{u(t)^2]_0^T}{xu(T)}\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\frac{\mathrm{u(T)}}{\mathrm{x}}\right] \\
&= e^{-rT}\mathbb{E}\left[\Phi'\left(\int_0^T X_s^x ds\right)\int_0^T \frac{X_t^x}{\mathrm{x}}\,\mathrm{dt}\right] \quad (****)
\end{aligned}
$$

From $(***)$ and $(****)$ we have $\forall\ \Phi \in C^1 \cap Lip(\mathbb{R}^{n+1}, \mathbb{R})$:

$$\Delta(x) = e^{-rT}\mathbb{E}\left[\Phi\left(\int_0^T X_s^x ds\right)\Pi\right] \blacksquare$$

(b)

Fixing $\sigma_t = \sigma$ constant $\forall t \in [0,T]$ as in Black-Scholes world. Since $w(s) = \frac{2X_s^x}{x\sigma\int_0^T X_s^x ds} \in$ $dom(\delta)$, with $X_s^x \in dom(\delta)$ and $\frac{2}{x\sigma\int_0^T X_s^x ds} \in \mathbb{D}$, we have (prop.7):

$$
\begin{aligned}
\Pi(w) \quad &= \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} - \int_0^T D_s\left(\frac{2}{x\sigma\int_0^T X_s^x ds}\right)X_s^x ds \\
&= \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} - \int_0^T D_s\left(\frac{2}{x\sigma\int_0^T X_t^x dt}\right)X_s^x ds \\
\text{(chain rule \& prop.6)} \quad &= \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} + \frac{\int_0^T 2X_s^x\left(\int_0^T D_s X_t^x dt\right)ds}{x\sigma\left(\int_0^T X_t^x dt\right)^2} \\
\text{(prop.9)} \quad &= \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} + \frac{\int_0^T 2X_s^x\left(\int_0^s X_t^x dt\right)ds}{x\left(\int_0^T X_t^x dt\right)^2}
\end{aligned}
$$

(same argument as in B.a) $$= \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} + \frac{1}{x}$$

From the dynamic $dX_t^x = rX_t^x dt + \sigma X_t^x dB_t$ with constant $r, \sigma, X_0^x = x$ we have:

$$\int_0^T X_s^x dB_s = \frac{1}{\sigma}\left(X_T^x - x - r\int_0^T X_s^x ds\right)$$

So:

$$\Pi = \frac{2}{x\sigma}\frac{\int_0^T X_s^x dB_s}{\int_0^T X_s^x ds} + \frac{1}{x} = \frac{2}{x\sigma^2}\frac{(X_T^x - x - r\int_0^T X_s^x ds)}{\int_0^T X_s^x ds} + \frac{1}{x}$$

Denote $A = \int_0^T X_s^x ds$

$$\Rightarrow \Pi = \frac{1}{x}\left[\frac{2}{A\sigma^2}(X_T^x - x) - \frac{2r}{\sigma^2} + 1\right]$$

Part C: Numerical part

The aim is to compute financial quantities associated to Asian options by Monte Carlo methods in a Black-Scholes model where x = 100, r = 0.03, $\sigma = 0.2$, T = 1, K1 = 100, K2 = 110. The confidence interval we will use is at 95%.

In order to use Monte Carlo methods we will use the following approximation:

$$\int_0^T X_s^x ds \approx \frac{T}{M} \sum_{k=0}^{M} X_{\frac{Tk}{M}}^x$$

In the following sections we will denote the price of option with

$$\mathbf{payoff1} = \left(\int_0^T X_s^x ds - K_1 \right)_+$$

is **price1**, its delta **Delta1**, the price of option with

$$\mathbf{payoff2} = \mathbf{1}_{K_1 < \int_0^T X_s^x ds < K_2}$$

is **price2**, its delta **Delta2**.

(a)

Using the approximation formula above with the given parameter, we setup & run the code **`partCa.m`** with the number of simulations N =100,000 which give us the following results: (LCI is length of confidence interval)

M	Price1	Empirical variance	Confidence interval		
			lbound	ubound	LCI
50	5.197568	57.558718	5.150544	5.244591	0.094047
150	5.28842	60.353461	5.240269	5.336571	0.096302
250	5.284098	59.133873	5.236436	5.331761	0.095325

*Table 1: Monte Carlo approximation results for **price1** with different number of time steps M (N = 100,000)*

M	Price2	Empirical variance	Confidence interval		
			lbound	ubound	LCI
50	0.295307	0.199375	0.292539	0.298074	0.005535
150	0.292628	0.19835	0.289868	0.295389	0.005521
250	0.288096	0.196584	0.285348	0.290844	0.005496

*Table 2: Monte Carlo approximation results for **price2** with different number of time steps M (N = 100,000)*

We notice with **payoff1**, when M increase the empirical variance also increase, which results in increase of LCI. The situation is opposite with **payoff2**, when M increase the empirical variance and LCI decrease. Therefore in the following estimations we will use M =50 for option with **payoff1**, and M = 250 for option with **payoff2**.

Results of MC simulations with N = 1000 to N = 51000 increase by 2000 can be found in the Appendix. The following figures show the convergence of each estimators when N increase. We take the Reference Value as the number estimated when running Monte Carlo simulation with N = 1,000,000. As expected, the estimators line Price1 and Price2 get closer to the Ref Value line (flat) when N increases, and also the confidence interval getting tighter.

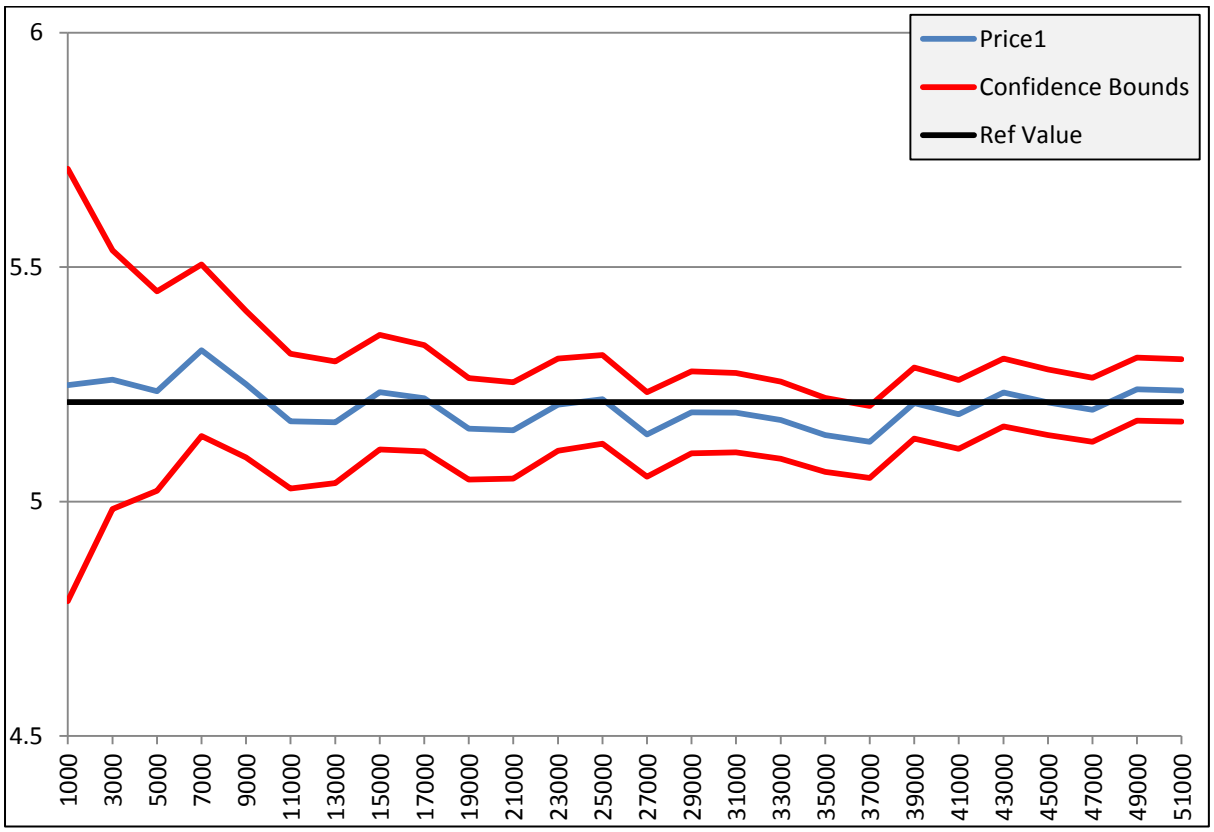

*Figure 1: Convergence of **price1** as the number of simulations N increase, with ref value taken when running N =1,000,000 (M = 50).*

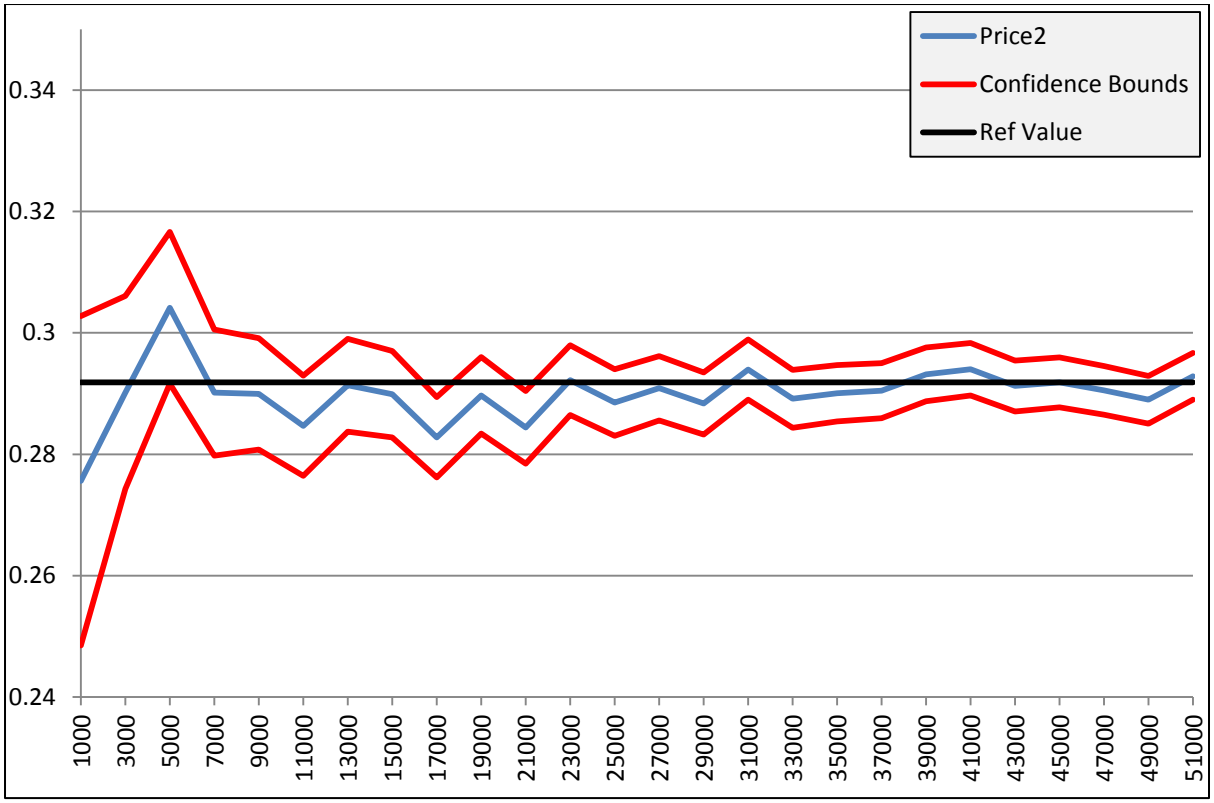

*Figure 2: Convergence of **price2** as the number of simulations N increase, with ref value taken when running N =1,000,000 (M = 250)*

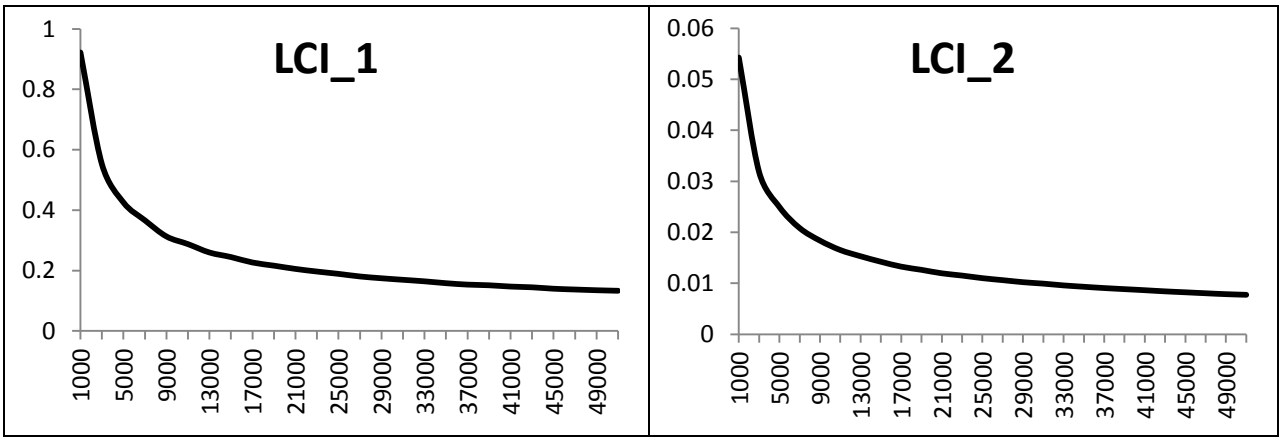

*Figure 3: Convergence of **LCI (length of confidence interval)** as the number of simulations N increase for both payoffs*

As in figure 3 showed, the length of confidence interval (LCI) will converge to 0 when N increases. However, it will decrease slowly as the rate of convergence is proportional to $\frac{1}{\sqrt{n}}$. In other words, in order to decrease LCI by a times, we have to increase the number of simulations by a^2 times.

(b)

Regarding the finite difference method, we will use the following approximations:

$$\Delta(x) = \frac{P(x+\epsilon) - P(x-\epsilon)}{2\epsilon}$$

which is the central difference approximation that introduces the smaller error compare to forward and backward difference. P(x) is approximated using same Monte Carlo method as in (a). Using the same notation rule we denote the delta of two payoff **Delta1** and **Delta2** respectively.

Also in `partCb.m` we use same Gaussian random variable sample for approximation of $P(x+\epsilon)$ and $P(x-\epsilon)$ following the same reason stated in the first section of the course. In the slides given in the first section, we know that:

- When ϵ is too big, the error from finite difference may strongly increase
- When ϵ is too small, the variance of the Monte Carlo estimator may explode

To demonstrate this, Table 3 and 4 show the results for approximation of the two payoffs. We can clearly see that for **payoff1**, when epsilon reduces to 10^{-14}, the approximated value of both **Delta1** and its empirical variance explode. For **payoff2**, we only need to reduce epsilon to 10^{-4} to observe the jump in value of **Delta2** and the empirical variance.

ϵ (log10)	Delta1	Empirical Variance	Confidence interval		
			Lbound	Ubound	LCI
-14	0.793698	1.237391	0.786804	0.800593	0.013789
-12	0.562079	0.285614	0.558767	0.565392	0.006625
-10	0.562758	0.288857	0.559427	0.566089	0.006662
-8	0.560204	0.288358	0.556876	0.563533	0.006657
-6	0.563384	0.288799	0.560053	0.566714	0.006661
-4	0.559856	0.288389	0.556527	0.563184	0.006657
-2	0.560262	0.288272	0.556934	0.56359	0.006656
-1	0.560636	0.287515	0.557313	0.563959	0.006646
0	0.560521	0.277667	0.557255	0.563787	0.006532

*Table 3: Finite difference approximation results for option with **payoff1** (M = 50; N = 100,000)*

ϵ (log10)	Delta	Empirical variance	Confidence interval		
			Lbound	Ubound	LCI
-4	0.048522	706.32811	-0.116203	0.213247	0.32945
-3	0.019409	28.252842	-0.013536	0.052354	0.06589
-2	0.010675	3.060651	-0.000168	0.021518	0.02169
-1	0.009899	0.291381	0.006553	0.013244	0.00669
0	0.007924	0.028136	0.006884	0.008963	0.00208

*Table 4: Finite difference approximation results for option with **payoff2** (M = 250; N = 100,000)*

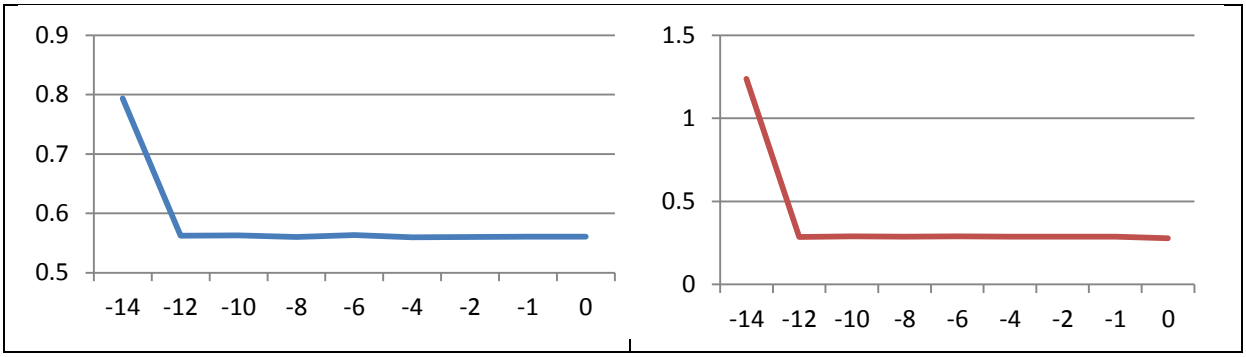

*Figure 4: Value of **Delta1** (left) and its empirical variance (right) as epsilon varies (log10),*

With M = 50 and N = 100,000

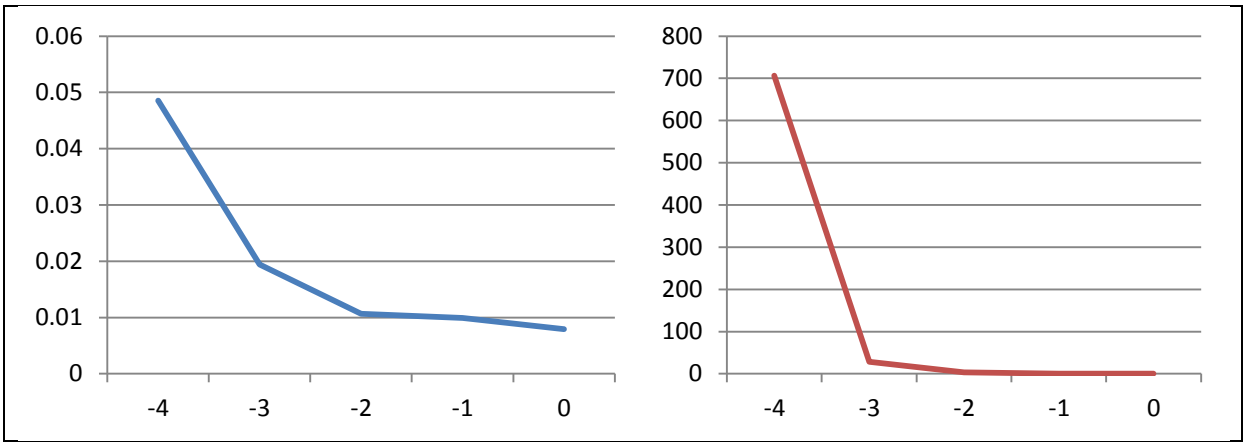

*Figure 5: Value of **Delta2** (left) and its empirical variance (right) as epsilon varies (log10),*

With M = 250 and N = 100,000

(c)

We use the file **`partCc_methodA.m`** for $\Delta(x)$ approximate using method in part A and **`partCc_methodB.m`** for method in part B. The results obtained are demonstrated in the following tables.

M	Delta1	Empirical variance	Confidence interval		
			Lbound	Ubound	LCI
50	0.539449	11.56202	0.518374	0.560525	0.042151
150	0.544409	34.040827	0.550999	0.622469	0.07147
250	0.542436	54.784029	0.49656	0.588311	0.091751

*Table 5: Monte Carlo approximation results for **Delta1** using **method A** (N = 100,000)*

M	Delta2	Empirical variance	Confidence interval		
			Lbound	Ubound	LCI
50	0.009449	0.033928	0.008307	0.01059	0.00228
150	0.008149	0.106192	0.00613	0.010169	0.00404
250	0.009869	0.174631	0.007279	0.012459	0.00518

*Table 6: Monte Carlo approximation results for **Delta2** using **method A** (N = 100,000)*

M	Delta	Empirical Variance	Confidence interval		
			Lbound	Ubound	LCI
50	0.548058	1.758114	0.53984	0.556276	0.016436
150	0.555897	1.818413	0.547539	0.564255	0.016716
250	0.556335	1.825174	0.547961	0.564708	0.016747

*Table 7: Monte Carlo approximation results for **Delta1** using **method B** (N = 100,000)*

M	Delta	Empirical Variance	Confidence interval		
			Lbound	Ubound	LCI
50	0.008837	0.000994	0.008641	0.009032	0.00039
150	0.008822	0.001021	0.008624	0.00902	0.0004
250	0.008729	0.001023	0.008531	0.008927	0.0004

*Table 8: Monte Carlo approximation results for **Delta2** using **method B** (N = 100,000)*

(d)

For graphical comparison purpose, we will fix M = 50, ϵ = 0.1 in the finite difference approximation for both options and make the number of simulations N increase from N = 1000 to N = 53000 by 4000. The reference value is taken by calculating the MC simulations for 3 methods at N = 1,000,000 and then taking the average.

With the option of **payoff1**, we notice that the Delta calculated with Finite Difference method and Method B shows better results than with Method A. The case is not the same for option of **payoff2**, as it is clear from the graphs that Method B

brings the best approximation and Finite Difference the worst, while using Method A also shows good results.

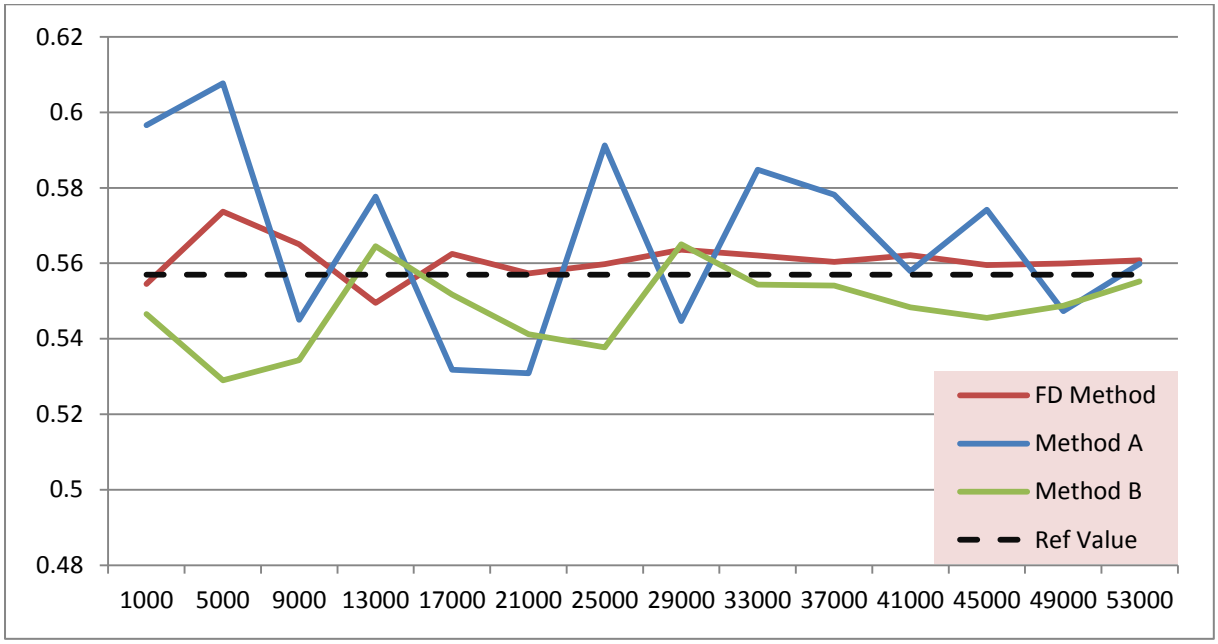

*Figure 6: Value of **Delta1** approximated by 3 methods (M = 50, ϵ = 0.1)*

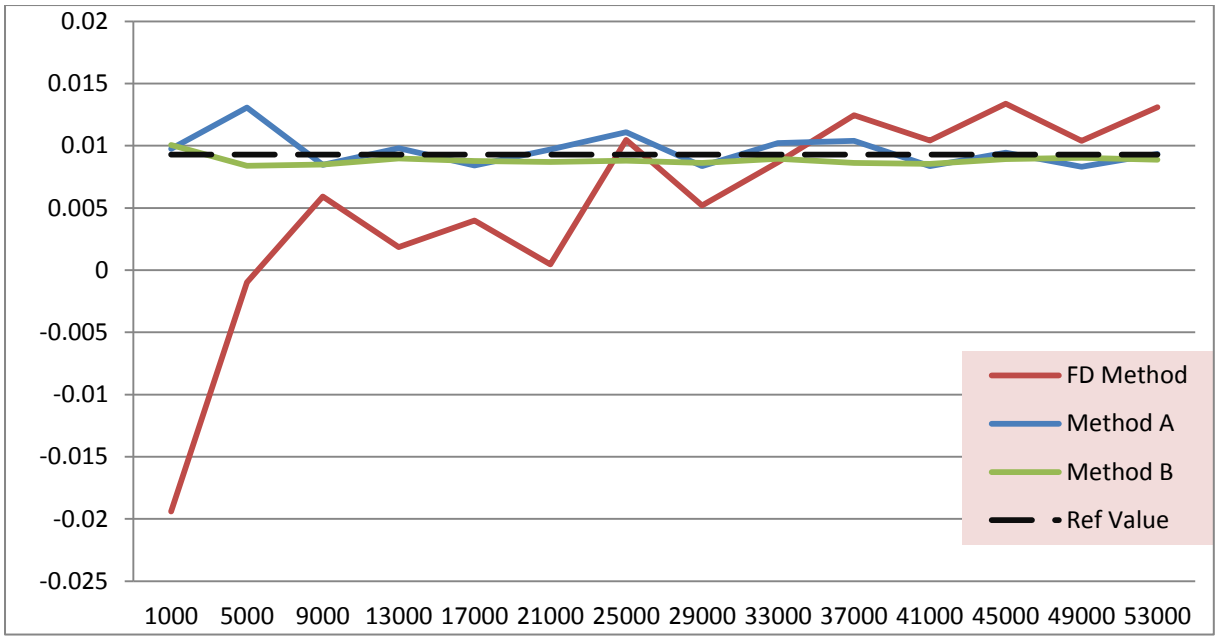

*Figure 7: Value of **Delta2** approximated by 3 methods (M = 50, ϵ = 0.1)*

In terms of empirical variance, we notice the same differences when comparing results from **payoff1** and **payoff2**. With **Delta1**, Method A performs the worst since its empirical variance is extremely large compare to the other two methods. Finite

difference method performs the best for this type of option, with Method B brings relatively small variance as well. However, for option with **payoff2**, FD method performs worst with large variance comparing with Method A and especially with Method B with the empirical variance line almost close to zero.

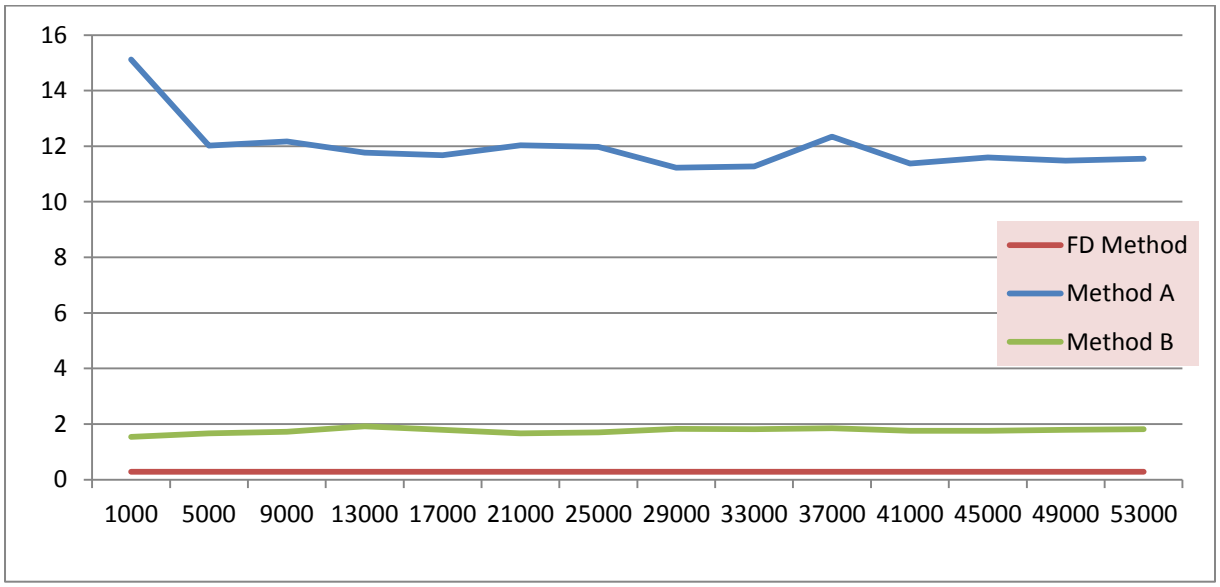

*Figure 8: Value of **Empirical Variance** of **Delta1** approximated by 3 methods (M = 50, $\epsilon = 0.1$)*

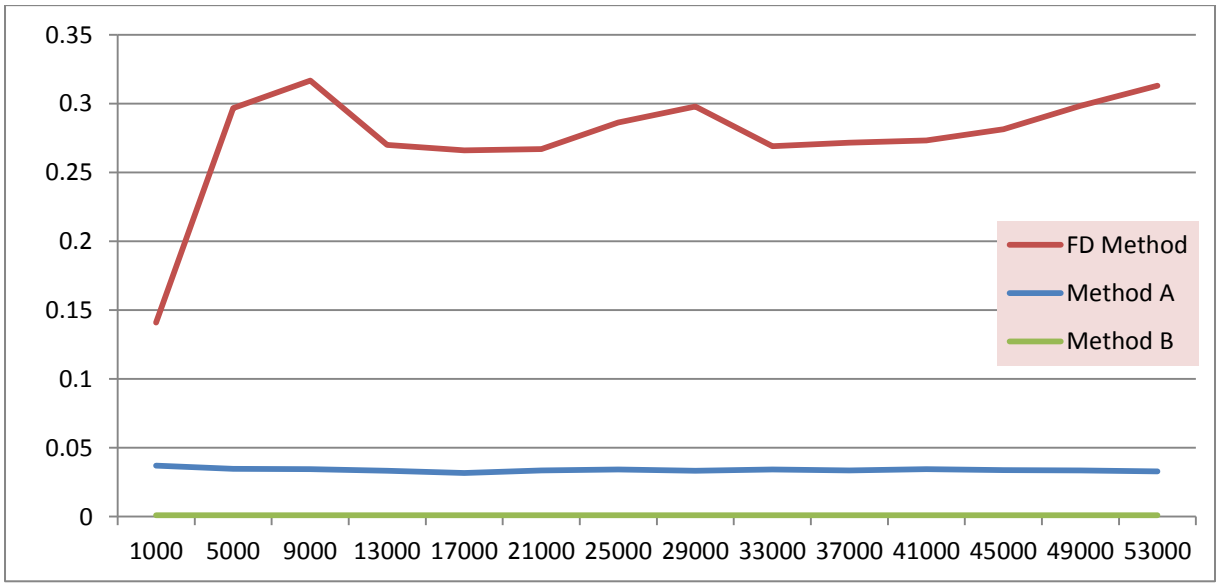

*Figure 9: Value of **Empirical Variance** of **Delta2** approximated by 3 methods (M = 50, $\epsilon = 0.1$)*

(e) Conclusion:

- For calculating Delta of option with **payoff1**, Finite Difference method performs really well. This can be explained by the facts that **payoff1** function

is continuous. However with **payoff2**, which is a binary option with discontinuous payoff, FD approximation results are the worst.

- However using FD method can be tricky since the choice of ϵ can be difficult. We showed that the variance and value of Delta approximated can explode when ϵ is small enough.
- Approximation of Delta with Monte Carlo simulation using weight Π showed in part A and part B is easier to implement. However method A does not perform really well when calculating delta of option with **payoff1**.
- In our opinion Method B is the best method out of 3 in terms of the ease in implementation and in empirical results. Unlike FD method, Method B performs numerically really well with both types of payoff function. The empirical variance is also really small, and in fact for **payoff2** it provides the smallest variance.

REFERENCES

1. Lecture notes and slides from the course
2. Nualart, D.: *Malliavin Calculus and Related Topics. (Probability and its Applications)* Berlin Heidelberg New York: Springer (1995)
3. Fournie et al., *Applications of Malliavin calculus to Monte Carlo methods in finance* , Finance and Stochastic (1999)

APPENDIX

A. Matlab code

partCa.m

```
% using MC simulation to calculate the price of Asian option
%------------------------
%- FINANCIAL PARAMETERS
%------------------------

global s0 r sigma T K1 K2
s0= 100; T= 1; r= 0.03; sigma= 0.2;
K1= 100; K2= 110;

%------------------------
%- NUMERICAL DATA
%------------------------

M= 'Enter the number of timesteps M: ';
M= input(M);
%M= 250;
N= 'Enter the number of simulations N: ';
N= input(N);
%N= 100000;
%------------------------
%- IMPLEMENTATION
%------------------------
dt = T/M; % timestep
% declare the drift and volatilty following the settings
drift = exp(dt * (r - 0.5 * sigma ^2 ));
vol = sqrt(dt) * sigma;
spot_price= ones(M,1)*s0;
asian_price = 0;
payoff1 = 0; % payoff for Asian fixed strike call
varpayoff1 = 0;
payoff2 = 0; % payoff for Asian binary option
varpayoff2 = 0;
%
tic();
for i=1:N
    for j=2:M
        spot_price(j)= spot_price(j-1) * drift * exp(vol*randn());

    end
    asian_price = mean(spot_price);
    payoff1 = payoff1 +  max(asian_price-K1,0);
    varpayoff1 = varpayoff1 + (max(asian_price-K1,0))^2;
    if K1 < asian_price && asian_price < K2
        payoff2 = payoff2 + 1;
        varpayoff2 = varpayoff2 + 1;
    end
end

payoff1 = 1/N*exp(-r*T)*payoff1;
varpayoff1 = N/(N-1)* (1/N*exp(-2*r*T)*varpayoff1 - (payoff1)^2);

payoff2 = 1/N*exp(-r*T)*payoff2;
varpayoff2 = N/(N-1)* (1/N*exp(-2*r*T)*varpayoff2 - (payoff2)^2);

%confidence interval (for 95%)
lbound1 = payoff1 - 1.96*sqrt(varpayoff1)/sqrt(N);
```

```
ubound1 = payoff1 + 1.96*sqrt(varpayoff1)/sqrt(N);
lbound2 = payoff2 - 1.96*sqrt(varpayoff2)/sqrt(N);
ubound2 = payoff2 + 1.96*sqrt(varpayoff2)/sqrt(N);
total_time = toc();
%------------------------
%- PRINTING RESULTS
%------------------------

fprintf('======================\n')
fprintf('Monte Carlo simulation\n\n')
%fprintf('======================\n')
fprintf('Parameters: K1= %3i, K2= %3i, s0= %3i, r= %3.2f, sigma= %3.2f\n',
K1, ...
        K2, s0, r, sigma)
fprintf('Number of time steps M= %5i, number of simulations N= %5i\n', M,
N)
fprintf('Total time running (sec) = %5.2f\n\n', total_time)
%fprintf('======================\n')
fprintf('Price1= %5.6f, varPrice1= %5.6f\n', payoff1, varpayoff1)
fprintf('95 percent CI = [%5.6f, %5.6f]\n\n', lbound1, ubound1)

fprintf('Price2= %5.6f, varPrice2= %5.6f\n', payoff2, varpayoff2)
fprintf('95 percent CI = [%5.6f, %5.6f]\n', lbound2, ubound2)
fprintf('======================\n')
```

partCb.m

```
% calculating delta using finite difference method and MC estimation for
% price function
%------------------------
%- FINANCIAL PARAMETERS
%------------------------

global s0 r sigma T K1 K2
s0 = 100; T= 1; r= 0.03; sigma= 0.2;
K1= 100; K2= 110;

%------------------------
%- NUMERICAL DATA
%------------------------

M= 'Enter the number of timesteps M: ';
M= input(M);
%M= 50;
N= 'Enter the number of simulations N: ';
N= input(N);
%N= 100000;
%-------------------------
%- IMPLEMENTATION
%-------------------------
dt = T/M; % timestep
% declare the drift and volatilty following the settings
epsi= 'Enter the value of epsilon: ';
epsi= input(epsi);
%epsi= 0.1;
drift = exp(dt * (r - 0.5 * sigma ^2 ));
vol = sqrt(dt) * sigma;

%used for calculating P(x-epsilon) and P(x+epsilon)
spot_price_low= ones(M,1)*(s0-epsi);
spot_price_high= ones(M,1)*(s0+epsi);
```

```
asian_price_low = 0;
asian_price_high = 0;

delta1 = 0; % delta for Asian fixed strike call
vardelta1 = 0;

payoff2 = 0;
delta2 = 0; % delta for Asian binary option
vardelta2 = 0;

% MC loop
tic();
for i=1:N
    A = randn(M,1); %generate Guassian vector - using same random sample as
                    %explained in the lecture note
    for j=2:M
        spot_price_low(j)= spot_price_low(j-1) * drift * exp(vol*A(j));
        spot_price_high(j)= spot_price_high(j-1) * drift * exp(vol*A(j));
    end
    asian_price_low = mean(spot_price_low);
    asian_price_high = mean(spot_price_high);
    delta1 = delta1 + (max(asian_price_high-K1,0) - max(asian_price_low-
K1,0))/(2*epsi);
    vardelta1 = vardelta1 + 1/(4*epsi^2)*(max(asian_price_high-K1,0) -
max(asian_price_low-K1,0))^2;
    if K1 < asian_price_low && asian_price_low < K2
        payoff2_low = 1;
    else
        payoff2_low = 0;
    end
    if K1 < asian_price_high && asian_price_high < K2
        payoff2_high = 1;
    else
        payoff2_high = 0;
    end
    delta2 = delta2 + (payoff2_high - payoff2_low)/(2*epsi);
    vardelta2 = vardelta2 + 1/(4*epsi^2)*(payoff2_high - payoff2_low)^2;
end

delta1 = 1/N*exp(-r*T)*delta1;
delta2 = 1/N*exp(-r*T)*delta2;
vardelta1 = N/(N-1) * (1/N*exp(-2*r*T)*vardelta1 - (delta1)^2);
vardelta2 = N/(N-1) * (1/N*exp(-2*r*T)*vardelta2 - (delta2)^2);

%confidence interval (for 95%)
lbound_delta1 = delta1 - 1.96*sqrt(vardelta1)/sqrt(N);
ubound_delta1 = delta1 + 1.96*sqrt(vardelta1)/sqrt(N);
lbound_delta2 = delta2 - 1.96*sqrt(vardelta2)/sqrt(N);
ubound_delta2 = delta2 + 1.96*sqrt(vardelta2)/sqrt(N);

total_time = toc();

%------------------------
%- PRINTING RESULTS
%------------------------

fprintf('======================\n')
fprintf('Monte Carlo simulation\n')
fprintf('======================\n')
```

```
fprintf('Parameters: K1= %3i, K2= %3i, s0= %3i, r= %3.2f, sigma= %3.2f\n',
K1, ...
        K2, s0, r, sigma)
fprintf('Number of time steps M= %5i, number of simulations N= %5i\n', M,
N)
fprintf('Total time running (sec) = %5.2f\n', total_time)
fprintf('======================\n')
fprintf('With epsilon = %5.6f\n\n', epsi)
fprintf('delta1= %5.6f, vardelta1= %5.6f\n', delta1, vardelta1)
fprintf('95 percent CI = [%5.6f, %5.6f]\n\n', lbound_delta1, ubound_delta1)

fprintf('delta2= %5.6f, vardelta2= %5.6f\n', delta2, vardelta2)
fprintf('95 percent CI = [%5.6f, %5.6f]\n', lbound_delta2, ubound_delta2)
fprintf('======================\n')
```

partCc_methodA.m

```
% using MC simulation to calculate the price of Asian option
%------------------------
%- FINANCIAL PARAMETERS
%------------------------

global s0 r sigma T K1 K2
s0= 100; T= 1; r= 0.03; sigma= 0.2;
K1= 100; K2= 110;

%------------------------
%- NUMERICAL DATA
%------------------------

%M= 'Enter the number of timesteps M: ';
%M= input(M);
M= 50;
N= 'Enter the number of simulations N: ';
N= input(N);
%N= 100000;
%------------------------
%- IMPLEMENTATION
%------------------------
dt = T/M; % timestep
% declare the drift and volatilty following the settings
drift = exp(dt * (r - 0.5 * sigma ^2 ));
vol = sqrt(dt) * sigma;
spot_price= ones(M,1)*s0;
asian_price = 0;

delta1 = 0; %delta for Asian fixed strike call
vardelta1 = 0;

delta2 = 0; %delta for Asian binary option
vardelta2 = 0;
%
tic();
for i=1:N
    A = randn(); % use this to generate B_t1 to calculate Pi;
    B_t1 = sqrt(dt) * A;
    spot_price(2)= spot_price(1) * drift * exp(vol*A);
    for j=3:M
        spot_price(j)= spot_price(j-1) * drift * exp(vol*randn());
    end
    asian_price = mean(spot_price);
```

```
    multiplier1 = max(asian_price-K1,0)*B_t1/(s0*sigma*dt);
    delta1 = delta1 + multiplier1;
    vardelta1 = vardelta1 + multiplier1^2;
    if K1 < asian_price && asian_price < K2
        multiplier2 = B_t1/(s0*sigma*dt);
    else
        multiplier2 = 0;
    end
    delta2 = delta2 + multiplier2;
    vardelta2 = vardelta2 + multiplier2^2;
end

delta1 = 1/N*exp(-r*T)*delta1;
vardelta1 = N/(N-1)* (1/N*exp(-2*r*T)*vardelta1 - (delta1)^2);

delta2 = 1/N*exp(-r*T)*delta2;
vardelta2 = N/(N-1)* (1/N*exp(-2*r*T)*vardelta2 - (delta2)^2);

%confidence interval (for 95%)
lbound1 = delta1 - 1.96*sqrt(vardelta1)/sqrt(N);
ubound1 = delta1 + 1.96*sqrt(vardelta1)/sqrt(N);
lbound2 = delta2 - 1.96*sqrt(vardelta2)/sqrt(N);
ubound2 = delta2 + 1.96*sqrt(vardelta2)/sqrt(N);
total_time = toc();
%------------------------
%- PRINTING RESULTS
%------------------------
fprintf('======================\n')
fprintf('Monte Carlo simulation\n')
fprintf('======================\n')
fprintf('Parameters: K1= %3i, K2= %3i, s0= %3i, r= %3.2f, sigma= %3.2f\n',
K1, ...
        K2, s0, r, sigma)
fprintf('Number of time steps M= %5i, number of simulations N= %5i\n', M,
N)
fprintf('Total time running (sec) = %5.2f\n', total_time)
fprintf('======================\n')
fprintf('delta1= %5.6f, vardelta1= %5.6f\n', delta1, vardelta1)
fprintf('95 percent CI = [%5.6f, %5.6f]\n\n', lbound1, ubound1)

fprintf('delta2= %5.6f, vardelta2= %5.6f\n', delta2, vardelta2)
fprintf('95 percent CI = [%5.6f, %5.6f]\n', lbound2, ubound2)
fprintf('======================\n')
```

partCc_methodB.m

```
% using MC simulation to calculate the price of Asian option
%------------------------
%- FINANCIAL PARAMETERS
%------------------------

global s0 r sigma T K1 K2
s0= 100; T= 1; r= 0.03; sigma= 0.2;
K1= 100; K2= 110;

%------------------------
%- NUMERICAL DATA
%------------------------

%M= 'Enter the number of timesteps M: ';
%M= input(M);
```

```
M= 50;
N= 'Enter the number of simulations N: ';
N= input(N);
%N= 100000;
%-------------------------
%- IMPLEMENTATION
%-------------------------
dt = T/M; % timestep
% declare the drift and volatilty following the settings
drift = exp(dt * (r - 0.5 * sigma ^2 ));
vol = sqrt(dt) * sigma;
spot_price= ones(M,1)*s0;
asian_price = 0;
delta1 = 0; % payoff for Asian fixed strike call
vardelta1 = 0;
delta2 = 0; % payoff for Asian binary option
vardelta2 = 0;
%
tic();
for i=1:N
    for j=2:M
        spot_price(j)= spot_price(j-1) * drift * exp(vol*randn());
    end
    asian_price = mean(spot_price);
    pai= 1/s0*(2/(asian_price*sigma^2)*(spot_price(M) - s0) - 2*r/(sigma^2)
+ 1);
    multiplier1 = max(asian_price-K1,0)*pai;
    delta1= delta1 + multiplier1;
    vardelta1 = vardelta1 + multiplier1^2;
    if K1 < asian_price && asian_price < K2
        multiplier2 = pai;
        delta2= delta2 + multiplier2;
        vardelta2 = vardelta2 + multiplier2^2;
    end

end

delta1 = 1/N*exp(-r*T)*delta1;
vardelta1 = N/(N-1)* (1/N*exp(-2*r*T)*vardelta1 - (delta1)^2);

delta2 = 1/N*exp(-r*T)*delta2;
vardelta2 = N/(N-1)* (1/N*exp(-2*r*T)*vardelta2 - (delta2)^2);

%confidence interval (for 95%)
lbound1 = delta1 - 1.96*sqrt(vardelta1)/sqrt(N);
ubound1 = delta1 + 1.96*sqrt(vardelta1)/sqrt(N);
lbound2 = delta2 - 1.96*sqrt(vardelta2)/sqrt(N);
ubound2 = delta2 + 1.96*sqrt(vardelta2)/sqrt(N);
total_time = toc();
%------------------------
%- PRINTING RESULTS
%------------------------

fprintf('======================\n')
fprintf('Monte Carlo simulation\n')
fprintf('======================\n')
fprintf('Parameters: K1= %3i, K2= %3i, s0= %3i, r= %3.2f, sigma= %3.2f\n',
K1, ...
        K2, s0, r, sigma)
fprintf('Number of time steps M= %5i, number of simulations N= %5i\n', M,
N)
```

```
fprintf('Total time running (sec) = %5.2f\n', total_time)
fprintf('======================\n')
fprintf('delta1= %5.6f, vardelta1= %5.6f\n', delta1, vardelta1)
fprintf('95 percent CI = [%5.6f, %5.6f]\n\n', lbound1, ubound1)

fprintf('delta2= %5.6f, vardelta2= %5.6f\n', delta2, vardelta2)
fprintf('95 percent CI = [%5.6f, %5.6f]\n', lbound2, ubound2)
fprintf('======================\n')
```

N	**Price1**	Confidence interval				**Price2**	confidence interval		
		Lbound	Ubound	LCI			Lboud	Ubound	LCI
1000	5.248577	4.787533	5.70962	0.922087		0.275607	0.24847	0.302743	0.054273
3000	5.259699	4.983791	5.535607	0.551816		0.290163	0.274262	0.306065	0.031803
5000	5.235414	5.022696	5.448132	0.425436		0.304138	0.291658	0.316617	0.024959
7000	5.322461	5.139417	5.505505	0.366088		0.290163	0.279754	0.300572	0.020818
9000	5.250511	5.094105	5.406917	0.312812		0.289948	0.28077	0.299125	0.018355
11000	5.171275	5.027525	5.315026	0.287501		0.284693	0.276436	0.292951	0.016515
13000	5.169182	5.039298	5.299066	0.259768		0.291358	0.283711	0.299004	0.015293
15000	5.233485	5.111176	5.355795	0.244619		0.289904	0.282796	0.297013	0.014217
17000	5.220234	5.106912	5.333555	0.226643		0.282799	0.27617	0.289429	0.013259
19000	5.155278	5.047232	5.263324	0.216092		0.289704	0.283389	0.296018	0.012629
21000	5.15174	5.049085	5.254395	0.20531		0.284433	0.278458	0.290408	0.01195
23000	5.20677	5.108444	5.305095	0.196651		0.292231	0.286477	0.297984	0.011507
25000	5.218117	5.123637	5.312597	0.18896		0.288533	0.283034	0.294032	0.010998
27000	5.143288	5.053105	5.233471	0.180366		0.290882	0.285579	0.296185	0.010606
29000	5.190168	5.10297	5.277365	0.174395		0.288356	0.283252	0.293461	0.010209
31000	5.18937	5.104739	5.274	0.169261		0.293951	0.288987	0.298915	0.009928
33000	5.17367	5.091539	5.2558	0.164261		0.289134	0.284345	0.293923	0.009578
35000	5.142047	5.063008	5.221086	0.158078		0.290052	0.285398	0.294707	0.009309
37000	5.127325	5.050615	5.204035	0.15342		0.290478	0.285949	0.295007	0.009058
39000	5.210002	5.134392	5.285613	0.151221		0.293174	0.288752	0.297597	0.008845
41000	5.185856	5.112403	5.259309	0.146906		0.294021	0.289704	0.298338	0.008634
43000	5.232629	5.160393	5.304865	0.144472		0.291247	0.287043	0.29545	0.008407
45000	5.211768	5.141885	5.281651	0.139766		0.291845	0.287733	0.295957	0.008224
47000	5.195759	5.127184	5.264334	0.13715		0.290535	0.286517	0.294553	0.008036
49000	5.239611	5.172227	5.306995	0.134768		0.288975	0.285046	0.292904	0.007858
51000	5.236761	5.170244	5.303278	0.133034		0.292846	0.28898	0.296712	0.007732

*Appendix B. Results of Monte Carlo simulations for option with **payoff1 (price1)** and **payoff2** (**price2**)*