Université Pierre et Marie Curie
Master 2 IFMA
Programmation en C++

Examen du 13 décembre 2016
Durée : 2 heures

— *Tous les fichiers rendus devront contenir vos* **nom et prénoms**.
— *Le sujet est à traiter* **seul**.
— *Il vous est interdit d'accéder à internet pendant l'épreuve. Vous pouvez toutefois télécharger au début de l'épreuve les notes de cours et les programmes d'exemples vus pendant le semestre. Vous avez aussi accès à vos notes de cours et à vos productions de TP.*
— *Vos fichiers seront envoyés à* `raphael.roux@upmc.fr`*, au maximum deux heures et cinq minutes après le début de l'épreuve (a priori à 15h05). Tout retard entraînera une note* **nulle**.
— *Il s'agit d'une epreuve de programmation et* **non de mathématiques**. *Si vous rencontrez une difficulté de compréhension des objets, n'hésitez pas à demander de l'aide.*
— *Le fait d'envoyer des fichiers qui ne se compilent pas correctement sera sanctionné.*

On va s'intéresser dans ce sujet à l'erreur commise par le *schéma d'Euler* pour la résolution numérique d'équations différentielles stochastiques.

1 Introduction

On va appliquer le schéma d'Euler à l'*équation d'Ornstein-Uhlenbeck*

$$\begin{cases} dX_t &= -X_t dt + dW_t, \\ X_0 &= 0. \end{cases} \tag{1}$$

Pour cette équation, une itération du schéma d'Euler de pas de temps δ prend la forme

$$\begin{aligned} \bar{X}_{(k+1)\delta} &= \bar{X}_{k\delta} - \delta \bar{X}_{k\delta} + (W_{(k+1)\delta} - W_{k\delta}) \\ &= (1-\delta)\bar{X}_{k\delta} + (W_{(k+1)\delta} - W_{k\delta}). \end{aligned}$$

Il se trouve que l'on est capables de simuler la solution exacte de l'équation d'Ornstein-Uhlenbeck sur la grille temporelle $(k\delta)_{k\in\mathbb{N}}$. En effet, on a l'expression

$$X_{(k+1)\delta} = e^{-\delta} X_{k\delta} + \int_{k\delta}^{(k+1)\delta} e^{-((k+1)\delta - s)} dW_s.$$

Par conséquent, pour simuler la suite de variables aléatoires $(X_{k\delta}, \bar{X}_{k\delta})_{k\in\mathbb{N}}$, il suffit d'être capable de simuler le couple

$$\left(\int_{k\delta}^{(k+1)\delta} e^{-((k+1)\delta - s)} dW_s, (W_{(k+1)\delta} - W_{k\delta}) \right). \tag{2}$$

On admettra que si G_1 et G_2 sont deux variables aléatoires de loi normale centrée réduites indépendantes alors la loi du couple

$$\left(\sqrt{\frac{1-e^{-2\delta}}{2}}G_1 \quad , \quad \sqrt{2\frac{1-e^{-\delta}}{1+e^{-\delta}}}G_1 + \sqrt{\delta - 2\frac{1-e^{-\delta}}{1+e^{-\delta}}}G_2\right) \tag{3}$$

est la même que celle de (2).

2 Implémentation

1. Dans un fichier `gauss.hpp`, écrire une fonction `double gauss(void)`, dont les appels successifs renverront une suite de variables aléatoires indépendantes de loi normale centrée réduite. On pourra utiliser la bibliothèque `random` du standard 2011, ou bien utiliser l'algorithme de Box-Muller (si U et V sont indépendantes de loi uniforme sur $[0,1]$ alors $\sqrt{-2\log(U)}\cos(2\pi V)$ suit la loi normale centrée réduite).
2. Dans un fichier `regression.hpp`, écrire une fonction `coeff_regression` prenant en argument deux listes de nombres `x` et `y` (représentées par une structure de donnée de votre choix), et renvoyant le coefficient α de la droite de regression linéaire des couples (x_i, y_i). On rappelle l'expression :

$$\alpha = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}, \quad \text{où } \bar{x} = \frac{1}{n}\sum_{i=1}^n x_i \text{ et } \bar{y} = \frac{1}{n}\sum_{i=1}^n y_i,$$

3. Dans un fichier `euler.hpp`, écrire une classe `Euler` contenant les membres et méthodes suivantes :
   — Comme membres privés, cinq variables de type `double`, nommées `dt`, `t`, `X0`, `Xt` et `Yt`, correspondant respectivement au pas de temps, au temps actuel de la simulation, à la condition initiale de l'équation, à la solution exacte au temps `t` et à la solution numérique au temps `t` obtenue par schéma d'Euler ;
   — Si besoin, des accesseurs pour les membres privés ;
   — Un constructeur prenant deux arguments correspondant à une initialisation de `dt` et `X0`. Les valeurs de `Xt` et `Yt` seront initialisées à la même valeur que `X0`, et `t` sera intialisé à `0` ;
   — Une méthode `reinitialisation` permettant de faire repartir la simulation de `t=0` ;
   — Une méthode `change_pas` permettant de changer le pas de temps ;
   — Une méthode `void un_pas(void)` qui fait avancer le temps de `dt` (avec le schéma pour `Yt` et avec la solution exacte pour `Xt`). C'est ici que l'on aura besoin du couple (3).
   — Une méthode `XT_YT` prenant en argument un double `T` et renvoyant (sous la forme de votre choix) le couple $(X_T, \bar{X}_T)$ composé de la solution exacte X_T au temps T et de la solution numérique $\bar{X}_T$ au temps T. Le pas de temps utilisé est la valeur actuelle de `dt`.

   Si nécessaire, on récrira le constructeur par copie, le destructeur et l'opérateur `=` pour cette classe.

3 Calcul et décroissance de l'erreur forte

On appelle *erreur forte* du schéma la quantité

$$\mathrm{err}_{T,\delta} = \mathbb{E}|X_T - \bar{X}_T|,$$

où $T = N\delta$. On admettra que cette quantité tend vers 0 quand δ tend vers 0, à une vitesse de l'ordre de $C\delta$.

4. Écrire une fonction `erreur_forte` prenant en argument un entier `M` et un réel `T` renvoyant une approximation par méthode de Monte Carlo avec `M` simulations de $\mathrm{err}_{T,\delta}$. Autrement dit, on renverra

$$\frac{1}{M}\sum_{k=1}^{M} |X_T^i - \bar{X}_T^i|,$$

les $(X^i, \bar{X}^i)$ étant des réalisations indépendantes de $(X, \bar{X})$.

5. Dans un fichier `schemas.cpp`, écrire un programme qui, à l'aide de la classe `Euler`, calcule (une approximation d)es couples $(\log(\delta), \log(\mathrm{err}_{T,\delta}))$ pour quelques valeurs de δ. On pourra prendre $\delta = 0.1 \times 2^{-k}$, $k \in \{0, \dots, 7\}$, avec $T = 5$ et $M = 500$ simulations. Calculer le coefficient de régression linéaire de $(\log(\delta), \log(\mathrm{err}_{T,\delta}))$ qui est une approximation du α tel que $\mathrm{err}_{T,\delta} = C\delta^{\alpha}$. On doit donc approximativement trouver 1.

4 Cas d'un coefficient de difusion non constant

Dans l'équation

$$\begin{cases} dX_t &= X_t(dt + dW_t), \\ X_0 &= 1. \end{cases} \tag{4}$$

dont la solution est $X_t = X_0 e^{W_t + t/2}$, le terme de bruit est multiplié par un terme *non constant* (contrairement au cas de l'équation (1)). Dans ce cas on peut montrer que l'erreur forte est de l'ordre de $C\delta^{1/2}$. On voit que le schéma d'Euler est donné par

$$\bar{X}_{(k+1)\delta} = \bar{X}_{k\delta}(1 + \delta + (W_{(k+1)\delta} - W_{k\delta})),$$

et que la solution exacte vérifie la relation

$$X_{(k+1)\delta} = X_{k\delta} e^{(W_{(k+1)\delta} - W_{k\delta}) + \delta/2}.$$

On peut donc simuler exactement la suite $(X_{k\delta}, \bar{X}_{k\delta})_{k\in\mathbb{N}}$ en simulant les $(W_{(k+1)\delta} - W_{k\delta})$ (qui sont indépendantes et de même loi Gaussienne de variance δ ; on peut donc les simuler comme $\sqrt{\delta}G_i$, où les G_i sont des $\mathcal{N}(0,1)$ indépendantes).

6. Écrire une classe `Euler2` dans un fichier `euler2.hpp` qui reprend la construction de la classe `Euler` dans le cas de l'équation (4). Compléter le fichier `schemas.cpp` pour calculer aussi le coefficient de régression linéaire de $(\log(\delta), \log(\mathrm{err}_{T,\delta}))$ pour cette équation (qui doit valoir approximativement 0.5).
7. Comment aurait-on pu réutiliser les programmes écrits dans la classe `Euler` pour faire moins de recopie ? Le faire (dans des fichiers séparés).