Université Pierre et Marie Curie
Master 2 IFMA 2014-2015

Projet : résolution d'équations aux dérivées partielles

À rendre pour le 10 décembre 2014, par mail à `raphael.roux@upmc.fr`. *Les réponses aux questions ne comportant pas de programmation sont à regrouper dans un fichier au format PDF.*

Le but de se projet est de présenter deux méthodes numériques permettant de résoudre une équation aux dérivée partielles *parabolique*, c'est-à-dire une équation de la forme

$$\partial_t u(t,x) = Lu(t,x),\ t>0,\ x\in\mathbb{R},$$

où L est un opérateur différentiel d'ordre 2. Plus précisément, on va considérer ici l'opérateur

$$Lf = \partial_x^2 f - \partial_x V \partial_x f$$

où V est une fonction $\mathcal{C}^1$. On s'intéressera aussi à l'équation parabolique associée à l'opérateur dit "dual" défini par

$$L^* f = \partial_x^2 f + \partial_x(f\partial_x V).$$

Enfin, pour assurer l'unicité de la solution, on se fixe une condition initiale $u_0(x)$.

Pour éviter les problèmes en $x\to\pm\infty$, on va supposer le problème *périodique*, c'est-à-dire que les deux fonctions u_0 et V sont 1-périodique. En conséquence, la fonction $x\mapsto u(t,x)$ sera elle aussi 1-périodique pour tout $t>0$.

Au final, on regarde les deux problèmes suivants :

$$\begin{cases}\partial_t u(t,x) = \partial_x^2 u(t,x) - \partial_x u(t,x)\partial_x V(x), & (t,x)\in\mathbb{R}\times]0,\infty[,\\ u(0,x) = u_0(x), & x\in\mathbb{R},\end{cases} \tag{1}$$

et

$$\begin{cases}\partial_t \mu(t,x) = \partial_x^2 \mu(t,x) + \partial_x(\mu(t,x)\partial_x V(x)), & (t,x)\in\mathbb{R}\times]0,\infty[,\\ \mu(0,x) = \mu_0(x), & x\in\mathbb{R}.\end{cases} \tag{2}$$

Dans la suite, on choisira $V(x)=\sin(2\pi x)$.

Note : on va être amenés à calculer les valeurs prises par une fonction sur une grille. Un moyen simple pour afficher ce résultat est d'utiliser le logiciel gnuplot, *qui peut être lancé en entrant* `gnuplot` *dans un terminal.*

Une fois gnuplot lancé, la commande `plot "monfichier" with lines` *permet d'afficher le contenu du fichier* `monfichier` *dans lequel une première colonne correspond à la variable x et une deuxième correspond à la variable y.*

Par ailleurs, la commande `plot sin(x)` *affiche le graphe de la fonction* sin *(par exemple). Il est également possible de modifier la zone d'affichage avec la commande* `set xrange [-2:2]` *(pour afficher la fonction sur l'intervalle [-2 , 2]).*

Pour tracer plusieurs graphes sur la même figure, remplacer `plot` *par* `replot`.

Enfin, les commandes `set terminal png` *puis* `set output "fichier.png"` *permettent de tracer la figure dans un fichier* `fichier.png` *plutôt que de l'afficher.*

Note : on va également vouloir que nos programmes puissent écrire dans un fichier. Pour cela, on a deux possibilités :

- *ou bien rediriger toute la sortie du programme vers un fichier, en lançant dans le terminal la commande* `./monprogramme > monfichier` ;
- *ou bien en utilisant la bibliothèque* `fstream`. *La syntaxe* `ofstream monflux("monficher")` *crée une variable* `monflux` *de type flux permettant d'écrire vers le fichier* `monfichier`. *Pour cela, on utilise l'opérateur* `<<` *de la même manière que l'on écrit sur la sortie standard avec le flux* `cout`. *On referme ensuite le fichier avec la syntaxe* `monflux.close()`.

1 Étude théorique

Le but de cette partie est de se faire une intuition du comportement des solutions des équations (1) et (2). Les réponses rigoureuses aux questions peuvent nécessiter certains outils complexes, aussi *on ne demande pas de justification rigoureuse des calculs*. Notamment, on admettra que les équations (1) et (2) admettent une *unique solution* aussi régulière que voulu.

1. Montrer que si la fonction u_0 est constante, alors la solution de l'équation (1) vérifie $u(t,x) = u_0(x)$ pour tout $t > 0$.

2. Montrer que si la fonction μ_0 est donnée par $\mu_0(x) = e^{-V(x)}$, alors pour tout $t > 0$ on a $\mu(t,x) = \mu_0(x)$.

3. Montrer que
$$\partial_t \int_0^1 |u(t,x)|^2 e^{-V(x)} \mathrm{d}x = -\int_0^1 |\partial_x u(t,x)|^2 e^{-V(x)} \mathrm{d}x,$$
En déduire que la quantité $\int_0^1 |u(t,x)|^2 e^{-V(x)} \mathrm{d}x$ est décroissante en t et converge quand $t \to \infty$. De plus montrer que la quantité $\int_0^1 |\partial_x u(t,x)|^2 e^{-V(x)} \mathrm{d}x$ converge vers 0 quand t tend vers l'infini (en admettant que cette quantité converge vers une limite).

Le fait que $\int_0^1 |\partial_x u(t,x)|^2 e^{-V(x)} \mathrm{d}x$ tend vers 0 revient à dire que $\partial_x u(t,x)$ tend vers 0 en un certain sens. Il est donc naturel que la fonction $x \mapsto u(t,x)$ se rapproche d'une fonction constante. Ceci est à rapprocher de la question 1. De la même manière, on pourrait montrer que $\mu(t,x)$ tend vers $Ce^{-V(x)}$ pour une certaine constante C.

On va maintenant donner une interprétation probabiliste des équations (1) et (2) à l'aide d'un processus stochastique. On définit le processus $(X_t)_{t\geq 0}$ par

$$\begin{cases} \mathrm{d}X_t &= \sqrt{2}\mathrm{d}W_t - \partial_x V(X_t)\mathrm{d}t, \\ X_0 & \text{ de loi donnée,} \end{cases} \tag{3}$$

(ici encore, on admettra qu'un tel processus existe et est unique). Pour l'intuition on peut comprendre $(X_t)_{t\geq 0}$ comme la position d'une particule qui subit un bruit thermique (dû au terme en $\mathrm{d}W_t$) ainsi qu'une force de rappel vers les minima de la fonction V (due au terme $-\partial_x V(X_t)\mathrm{d}t$).

4. En appliquant la formule d'Itō, montrer que le processus $(u(T-t, X_t))_{t\geq 0}$ est une martingale (ici u est la solution de (1)).

5. En déduire que $u(t,x) = \mathbb{E}[u_0(X_t)|X_0 = x]$. Retrouver à partir de cela le résultat de la question 1.

6. On suppose que X_t est distribué selon une mesure de probabilité $\tilde{\mu}(t,x)\mathrm{d}x$, En appliquant la formule d'Itō au processus $(\varphi(X_t))_{t\geq 0}$, montrer que
$$\int_0^t \int_0^1 \Big(\partial_t \tilde{\mu}(t,x) - \Delta\tilde{\mu}(t,x) - \partial_x(\tilde{\mu}(t,x)\partial_x V(t,x))\Big)\varphi(x)\mathrm{d}x\mathrm{d}t = 0$$
quelle que soit la fonction φ.

7. En déduire que si $\tilde{\mu}(0,x) = \mu_0(x)$, alors $\tilde{\mu}(t,x) = \mu(t,x)\mathrm{d}x$ où μ est la solution de (2).

On va maintenant présenter deux méthodes permettant de résoudre les équations (1) et (2). La première méthode, déterministe, sera directement inspirée des équations elles-mêmes. La deuxième méthode, probabiliste, sera basée sur le lien entre les équations (1) et (2) et le processus défini par (3).

2 Méthode déterministe

Le principe est de *discrétiser* les équations (1) et (2) en remplaçant la variable de temps $t \in [0, \infty[$ et la variable d'espace $x \in \mathbb{R}$ par des variables discrètes, respectivement $n\delta t$, avec $n \geq 0$, et $k\delta x$, avec $k \in \mathbb{Z}$, où $\delta t > 0$ et $\delta x > 0$ sont un *pas de temps* et un *pas d'espace* fixés à l'avance. On va donc approcher la famille $(u(t,x))_{t\geq 0, x\in\mathbb{R}}$ par une famille $(u_n^k)_{n\in\mathbb{N},\ k\in\mathbb{Z}}$ avec l'objectif

$$u_n^k \simeq u(n\delta t, k\delta x).$$

Pour conserver la périodicité, on va supposer δx de la forme $\delta x = \frac{1}{M}$ pour M un entier > 0 et on supposera la famille $(u_n^k)_{n\in\mathbb{N},k\in\mathbb{Z}}$ M-périodique par rapport à la variable k.

L'équation (1) peut alors être approchée par

$$\frac{u_{n+1}^k - u_n^k}{\delta t} = \frac{u_n^{k+1} - 2u_n^k + u_n^{k-1}}{\delta x^2} - V'(k\delta x)\frac{u_n^{k+1} - u_n^{k-1}}{2\delta x}.$$

avec la condition initiale $u_0^k = u_0(k\delta x)$ et l'équation (2) par

$$\frac{\mu_{n+1}^k - \mu_n^k}{\delta t} = \frac{\mu_n^{k+1} - 2\mu_n^k + \mu_n^{k-1}}{\delta x^2} + \frac{V'((k+1)\delta x)\mu_n^{k+1} - V'((k-1)\delta x)\mu_n^{k-1}}{2\delta x}.$$

avec la condition initiale $\mu_0^k = \mu_0(k\delta x)$.

1. Montrer que la famille $(u_n^k)_{n\in\mathbb{N},k\in\mathbb{Z}}$ est également M-périodique en k et qu'elle est entièrement déterminée par le vecteur $(u_n^k)_{0\leq k<M}$. Montrer qu'il existe des coefficients α^k, β^k et γ^k, qui sont M périodiques par rapports à k et tels que

$$\forall n \geq 0, \forall k \in \mathbb{Z},\ u_{n+1}^k = \alpha^k u_n^k + \beta^k u_n^{k-1} + \gamma^k u_n^{k+1}.$$

2. Programmer cette méthode. Pour cela, définir deux variables globales `M` et `dt` correspondant à $M = 1/\delta x$ et δt, et coder une fonction de prototype

```
void solution(double T, double *u0, double *u)
```

qui prend en argument un réel `T` et les adresse `u` et `u0` de deux vecteurs de `M` réels. La fonction `solution` va remplir le vecteur `u` avec la solution approchée $(u_n^k)_{0\leq k\leq M}$, où n a été choisi de sorte que la valeur de T soit dans $[n\delta t, (n+1)\delta t[$, et pour laquelle la condition initiale choisie était donnée par le vecteur `u0`.

3. Écrire une fonction de prototype

```
void affiche(double *u)
```

qui prend en argument l'adresse d'un vecteur (de `M` réels) et affiche (sur la sortie standard ou dans un fichier) sur deux colonnes les valeurs `k/(double)M` et `u[k]`, pour `k` variant de `0` à `M-1`. Utiliser cette fonction et le logiciel gnuplot pour afficher la solution approchée $(u_n^k)_{0\leq k<M}$ pour un n fixé.

Quelle remarque peut-on faire sur le comportement de la solution calculée numériquement en fonction des quantités $\frac{\delta t}{\delta x^2}$ et $\frac{\delta t}{\delta x}$?

4. Écrire une fonction de prototype

```
double norme2(double *u)
```

qui prend en argument un vecteur et renvoie la quantité

$$\left(\frac{1}{M}\sum_{k=0}^{M-1} |u_k|^2 e^{-V(k/M)}\right)^{1/2}$$

qui est une approximation de $\left(\int_0^1 |u(x)|^2 e^{-V(x)}\mathrm{d}x\right)^{1/2}$ si le vecteur $(u_k)_{k\in\mathbb{Z}}$ est une approximation de la fonction u.

5. Un résultat théorique est que la solution u de (1) tend vers la constante $C = \frac{\int_0^1 u_0(x)e^{-V(x)}\mathrm{d}x}{\int_0^1 e^{-V(x)}\mathrm{d}x}$. À l'aide de la fonction `norme2`, donner le comportement asymptotique de la quantité

$$\int_0^1 |u(t,x) - C|^2 e^{-V(x)}\mathrm{d}x$$

quand t tend vers ∞.

6. Reprendre les questions précédentes pour étudier la fonction μ solution de (2). Attention, dans ce cas, la limite en temps long de μ est $e^{-V(x)} \times \frac{\int_0^1 \mu_0(x)\mathrm{d}x}{\int_0^1 e^{-V(x)}\mathrm{d}x}$.

3 Méthode probabiliste

On va approcher le processus (3) par une suite de variables aléatoires. Plus précisément, en définissant la suite $(\bar{X}_n)_{n\in\mathbb{N}}$ par

$$\bar{X}_{n+1} - \bar{X}_n = W_n\sqrt{2\delta t} - V'(\bar{X}_n)\delta t,$$

où $(W_n)_{n\in\mathbb{N}}$ est une suite de variables aléatoires Gaussiennes centrées réduites, on s'attend à avoir

$$X_{n\delta t} \simeq \bar{X}_n.$$

1. Programmer la méthode de Box-Muller pour la simulation de variables aléatoires Gaussiennes, avec une fonction de prototype

   `double gaussienne(doule var, double moy)`,

   où les variables `var` et `moy` correspondent à la variance et à la moyenne de la variable à simuler. On donnera à la fonction `gaussienne` les paramètres par défaut var=1 et moy=0.

2. Écrire une fonction de prototype

   `double processus(double x, double t)`

   qui a deux réels `x` et `t` associe une réalisation de la variable aléatoire $\bar{X}_n$ pour laquelle $\bar{X}_0$ valait `x` et où n est choisi pour que `t`$\in [n\delta t, (n+1)\delta t[$.

3. Écrire une fonction

   `void MonteCarlo(int K, double u0(double), double *u, double *erreur)`

   qui implémente une méthode de Monte Carlo *avec intervalle de confiance* pour calculer la solution de l'équation (1) en chacun des points de la forme `k/(double)M` avec `k` variant de `0` à `M-1`, avec la condition initiale `u0`. Le paramètre `K` est le nombre de simulations faites. On rappelle que $u(t,x) = \mathbb{E}[u_0(X_t)|X_0 = x]$.

   Le vecteur `u` contiendra l'approximation de la densité, et le vecteur `erreur` la largeur de l'intervalle de confiance.

4. Écrire une fonction

   `void densite(int K, double t, double *mu, double *erreur)`

   qui simule `K` réalisations de la variable $\bar{X}_n$ (avec n tel que `t`$\in [n\delta t, (n+1)\delta t[$) et qui place dans le vecteur `mu` l'approximation suivante de la loi de $\bar{X}_n$: la quantité `mu[i]` correspond à la proportion de réalisations qui sont tombées dans l'intervalle $[i/M, (i+1)/M[$, pour i variant de 0 à M.

   La quantité `erreur[i]` contient la largeur de l'intervalle de confiance pour `mu[i]`.

   On rappelle que la loi de X_t est donnée par $\mu(t,x)\mathrm{d}x$. Il est donc intéressant de comparer le résultat de cette question avec la simulation déterministe de $\mu(t,x)$, ou avec la loi limite $e^{-V(x)}\mathrm{d}x \times (\int_0^1 e^{-V(x)}\mathrm{d}x)^{-1}$.