Université d'Angers – Mathématiques – L2 MPCIE SCILAB 2013-2014

Examen final

13 Mai 2014 – Durée : 3 heures

Modalités : *seul le mémento Scilab est autorisé. Vous ouvrirez en début de séance un fichier nommé* `nom-prenom.sce` *dans lequel vous enregistrerez le code et les fonctions* SCILAB *demandées. En fin de séance, enregistrez la dernière version de votre fichier sur le disque* `D` *et ne pas éteindre votre poste. Dans tous les cas, ne partez pas avant que l'enseignant ait vérifié l'enregistrement de votre fichier. Il est bien sûr vivement recommandé d'enregistrer régulièrement vos fichiers. Le barème est indicatif. Les exercices sont indépendants. Si vous êtes bloqué(e) dans un exercice, traitez le suivant ; passez un temps raisonnable à chercher d'éventuelles erreurs.*

Exercice 1 : [5 points] Soit $f : [a,b] \to \mathbb{R}$ une fonction continue. Cet exercice présente la *méthode de Monte-Carlo* la plus simple pour calculer l'intégrale $I = \int_a^b f(x)\,\mathrm{d}x$. Pour cela on choisit une subdivision *régulière* $a = a_0 < a_1 < \cdots < a_n = b$ de $[a,b]$ en n sous-intervalles de longueur $h = \frac{b-a}{n}$; par la relation de CHASLES on se ramène à calculer I sur des petits intervalles $[a_i, a_{i+1}]$:

$$I = \sum_{i=0}^{n-1} \int_{a_i}^{a_{i+1}} f(x)\,\mathrm{d}x$$

Dans la méthode de Monte-Carlo présentée ici, sur chaque intervalle $[a_i, a_{i+1}]$, la fonction f est approchée par $f(\xi_i)$ où ξ_i est un nombre pris au hasard dans $[a_i, a_{i+1}]$. On forme alors le rectangle adéquat puis on somme l'aire de tous ces rectangles, comme pour les méthodes des rectangles usuelles. Puisque la subdivision est régulière la formule d'approximation s'écrit :

$$I_n = \frac{b-a}{n} \sum_{i=0}^{n-1} f(\xi_i)$$

La loi faible des grands nombres permet de justifier le fait que lorsque $n \to +\infty$, alors $I_n \to I$.

▶ Question 1 : [1 point] La fonction `rand` de SCILAB renvoie un nombre pseudo-aléatoire dans $[0,1]$ selon une loi uniforme. A l'aide de la fonction `rand`, créez une fonction `alea(u,v)` qui prend en entrée deux réels u et v et donne en sortie un réel pseudo-aléatoire dans $[u,v]$.

▶ Question 2 : [4 points] Réalisez une fonction `calculMonteCarlo(f,a,b,n)` qui calcule l'intégrale I selon la méthode de Monte-Carlo. Testez votre fonction pour diverses fonctions f simples.

Exercice 2 : [8 points] Soit P la fonction polynomiale de degré n définie par :

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

On cherche à *evaluer* cette fonction en un point $x_0 \in \mathbb{R}$, c'est-à-dire à calculer $P(x_0)$.

▶ Question 1 : [1 point] Programmez naïvement (en utilisant l'opérateur puissance de SCILAB) cette évaluation par une fonction `evaluation(P,x0)`. *Indication* : Vous représenterez le polynôme P dans l'ordinateur par le vecteur $[a_0, a_1, \ldots, a_n]$ de ses coefficients.

En pratique une multiplication est plus coûteuse qu'une addition dans l'ordinateur. Avec l'algorithme précédent, le calcul de l'évaluation est d'environ n^2 multiplications. Ceci peut être amélioré en utilisant l'algorithme

d'*exponentiation rapide* qui est basé sur la remarque suivante :

$$x^k = \begin{cases} \left(x^{\frac{k}{2}}\right)^2 & \text{si } k \text{ est pair} \\ x \cdot \left(x^{\frac{k-1}{2}}\right)^2 & \text{si } k \text{ est impair} \end{cases}$$

Par exemple, pour calculer x^8, on remarque que $x^8 = x^4 \cdot x^4$, donc si on a déjà calculé x^4, il suffit de faire une multiplication par x^4 encore pour obtenir x^8. De même $x^4 = x^2 \cdot x^2$ et donc si on connait x^2, il suffit de faire une multiplication par x^2 encore pour obtenir x^4. Enfin pour x^2 il suffit de multiplier x par lui-même. Au total, on a effectué 3 multiplications au lieu de 7. Si on utilise cet algorithme dans notre évaluation, le coût est dominé par $n \log_2 n$ multiplications au lieu de n^2 ce qui est bien meilleur.

▶ Question 2 : [4 points] Programmez une fonction `expo(x,k)` mettant en oeuvre cet algorithme et modifiez la fonction d'évaluation. *Indication* : créez une fonction récursive et utilisez `modulo`.

On peut encore faire mieux en utilisant l'*algorithme de* Hörner, qui est basé sur la remarque suivante :

$$P(x_0) = \Big(\big(\ldots \big((a_n x_0 + a_{n-1}) \cdot x_0 + a_{n-2} \big) \cdot x_0 + \ldots \big) \cdot x_0 + a_1 \Big) \cdot x_0 + a_0$$

Autrement dit, on commence avec a_n. Ensuite on multiplie par x_0 et on ajoute a_{n-1}. Puis on multiplie ce nombre par x_0 et ajoute a_{n-2}, et ainsi de suite jusqu'à avoir ajouté a_0. Ainsi on a seulement besoin de n multiplications !

▶ Question 3 : [3 points] Programmez cet algorithme dans une fonction `horner(P,x0)`. *Indication* : il est recommandé d'utiliser une boucle `for` (éventuellement décroissante).

Exercice 3 : [7 points] Dans les années 1920, Lotka, qui travaillait sur la modélisation de réactions chimiques autocatalytiques, et Volterra, qui travaillait sur l'analyse statistique de la pêche en Adriatique ; sont tous deux à l'origine du premier modèle proie-prédateur connu aujourd'hui sous le nom du *modèle de* Lotka-Volterra. Soit $t \mapsto S(t)$ une population de sardines et soit $t \mapsto R(t)$ une population de requins : les requins (prédateurs) mangent des sardines (proies). Au temps $t_0 = 0$ le nombre de sardines est noté $S(0) = S_0$ et le nombre de requins est $R(0) = R_0$. Après avoir fait certaines hypothèses environnementales et comportementales sur ces populations, on montre que ce système biologique peut être modélisé par le système différentiel non-linéaire autonome suivant :

$$\begin{cases} \dfrac{\mathrm{d}S}{\mathrm{d}t}(t) = aS(t) - bS(t)R(t) \\ \\ \dfrac{\mathrm{d}R}{\mathrm{d}t}(t) = cS(t)R(t) - dR(t) \end{cases}$$

pour $0 \leq t \leq T$ avec a, b, c et d des coefficients qui ont un sens biologique.

▶ Question 1 : [7 points] Fabriquez une fonction `resoudreEuler(S0,R0,T,N,a,b,c,d)` qui résout le système de Lotka-Volterra ci-dessus par la méthode d'Euler en N pas. De plus votre fonction doit afficher les courbes des populations de sardines et de requins avec deux couleurs différentes (option de `plot2d`) et une légende pour les courbes et les axes (via les instructions `legends` et `xlabel`). *Indication* : appliquez la méthode d'Euler à chacune des *deux* lignes du système, qui correspondent à *deux* équations différentielles scalaires. Pour tester votre fonction vous prendrez pour paramètres : $S_0 = 5$, $R_0 = 2$, $T = 50$, $N = 1000$, $a = 1$, $b = 0.2$, $c = 0.04$ et $d = 0.5$.

Vous constaterez que les populations augmentent anormalement à chaque cycle. Ceci est dû au fait que la méthode d'Euler est inefficace dans la conservation de l'énergie du système.

Fin