IDM et Simulation TP 2

Ingénierie des modèles & Simulation

But du TP: Tester une technique de méta-programmation (génération de code par un autre programme). L'appliquer à une simulation de Monte Carlo (génération des valeurs pseudo-aléatoires) et évaluer les performances.

1. Chercher la dernière version du code source en C du générateur « original » Mersenne Twister (nommé MT ci-après de Makoto Mastumoto sur sa « Home page » (MT – Explanations & C code). Compiler ce code et **tester que le programme est bien porté. Les sorties doivent être conformes à ce qui est annoncé par l'auteur qui donne un exemple du résultat attendu. C'est aussi le principe de reproductibilité scientifique.** Le README de Matsumoto - les résultats que l'on doit obtenir quand le générateur a été bien porté sont fournis dans un fichier « output »).

2. Utiliser ce générateur pour produire un (des) code(s) source(s) en C avec simplement une déclaration et une initialisation de tableau de N nombres pseudo-aléatoires.

```
float tabMT-1[] = {
       premier nombre tiré,
       deuxième ombre,
       etc.,
       …
       } ;
```

3. Reprendre un code de simulation de Monte Carlo pour calculer PI (à programmer ou à trouver sur le Net). Tester ce code en faisant des appels au générateur MT en faisant 10 réplications séquentielles avec 1 000 000 de points (10^7 points au total). Mesurer le temps de calcul (avec le time Unix) et donner l'écart moyen avec la constante M_PI.

4. Reprendre le code du point 3 et intégrer le code généré au point 2 pour 10 millions de points – utiliser les nombres stockés dans le tableau plutôt que de réaliser les tirages en appelant le générateur (optimisation par dite par « lookup table »). On peut faire de la compilation séparée (ou pas). Noter le temps de compilation (et d'édition des liens). Attention : chaque point nécessite 2 nombres pseudo-aléatoires).

5. Comparer les performances en temps machine entre : (1) entre tirer des nombres pseudo-aléatoires avec MT et (2) lire ces nombres pré-calculé dans le tableau généré (prendre un nombre significatif de tirages dans les limites de la taille possible pour une compilation – zone de swap spécifique qui pourrait être étendue).

6. Ajouter un facteur 1000 et mesurer également le temps de calcul (soit 10^{10} points) en effectuant réellement les tirages (ne pas utiliser de tableau pré-calculé pour le milliard de points. Quels seraient les problèmes / contraintes si l'on essaye si l'on souhaite utiliser une technique de méta programmation pour la question 7 (générant 10 milliards de nombres) ?

7. Pour les plus avancés qui ont du temps: au lieu de générer un code source avec un tableau de nombres issus de MT, écrire ces nombres dans un fichier (binaire) et étudier ou redécouvrir la technique dite du « memory mapping » pour adresser le fichier binaire comme si il s'agissait d'un tableau en mémoire. Pour le contexte que nous venons de voir aux points 6) et 7) est-ce que cela règle / lève des contraintes.