Université d'Evry Val d'Essonne
M2 GM CS - Année universitaire 2013/2014

**CS93 - Calcul Haute Performance**

**TP Final - Programmation C++ d'un solveur itératif préconditionné parallèle**

**Consignes :**

• tout document (cours, internet...) autorisé,
• travail individuel,
• 1/4 de la note est lié à la progression observée durant la séance,
• 3/4 de la note porte sur le compte-rendu qui sera transmis sous format .pdf au plus tard le 13/12/13,
• le compte-rendu détaillera de manière concise et pertinente la démarche de modélisation et proposera des commentaires critiques sur les résultats,
• tous les fichiers utiles (cours, scripts) sont disponibles : http ://lmee.univ-evry.fr/∼hello/ENS/CS93/
• pour toute question ou remarque : gaetan.hello@ufrst.univ-evry.fr

## 1/ Solveur itératif

L'objet de ce premier problème consiste à résoudre numériquement un système linéaire au moyen de l'algorithme itératif de "plus forte pente" (*steepest descent*).

**1.1** Rappeler sous quelle(s) condition(s) il est possible d'utiliser cette approche pour résoudre un système linéaire.

**1.2** Récupérer les fichiers C++ .cpp et .h associés aux classes *Vecteur*, *Matrice*, *Probleme*.

**1.3** Créer un projet Visual C++ contenant ces fichiers.

**1.4** Programmer l'algorithme de plus forte pente dans la méthode *AlgoSD* de la classe *Probleme* en utilisant à votre choix les opérateurs surchargés ou les méthodes appropriées des classes *Vecteur* et *Matrice*.

**1.5** Pour $n = 100$, un nombre d'itération maximum de $10^5$ et une tolérance sur la norme du résidu de $10^{-4}$, tracer la courbe d'évolution de cette norme au cours des itérations.

## 2/ Effet du pré-conditionnement

On se propose ici d'observer l'influence du préconditionnement sur le comportement itératif de la méthode de plus forte pente.

**2.1** Compléter la classe *Probleme* de sorte à gérer le préconditionnement de Jacobi.

**2.2** Pour les mêmes conditions que **1.5**, tracer l'évolution de la norme du résidu au fil des itérations.

## 3/ Effet de la parallélisation

On se propose enfin d'accélérer la résolution du problème en exploitant les directives de parallélisation openMP.

**3.1** Rechercher dans l'ensemble programme les zones susceptibles d'êtres parallélisées.

**3.2** Etudier les effets de la parallélisation (nombres de threads...) sur les durées d'exécution du solveur initial et du solveur préconditionné.

**3.3** Minimiser le plus possible le temps d'exécution du solveur préconditionné (bonus au meilleur score !).