TD1: Méthode de Monte-Carlo

https://gitlab.com/nestordemeure/sgi2018

Vous devez compléter les fonctions suivantes jusqu'à ce que le code fonctionne sans erreur puis analyser les résultats.

Les fonctions à compléter ont été remplacées par des exceptions `Function not yet implemented`. Le code compile mais crash dès qu'il essaye d'appeler une de ces fonctions.

Les commentaires de chaque fonction contiennent les informations (spécification et fonctions utiles) nécessaires à leur implémentation.

1. Génération de nombres aléatoire

Compléter les fonctions `randu` et `randn` situé dans '`src/tools/matlab.h`' en vous appuyant sur les algorithmes de génération de nombres aléatoires que vous avez développés lors des TD précédents.

2. Suivi de plusieurs valeurs propres

Compléter les deux versions de la fonction `linearPath` située dans '`src/follower/transform.h`'. Vous paralléliserez son exécution avec OpenMP.

Facultatif : compléter la fonction `treePath` qui utilise un arbre couvrant minimal pour accélérer les calculs en recyclant une partie du suivi. Mesurer les speed-up obtenus (en fonction du nombre de valeurs à suivre et de la taille des matrices). *Il y aura un bonus pour l'implémentation, correcte, la plus rapide.*

3. Analyse des données

Le programme suit 9 valeurs propres complexes (les 18 colonnes du fichier '`phase_montecarlo.csv`') pour 1000 valeurs de paramètre (`xiVector`) tirées suivant une loi normale. Vous pouvez utiliser et modifier le fichier `plot.R` pour afficher les sorties du programme.

Étudier la distribution des parties réelles et imaginaires de chaque valeur propre (vous pouvez faire plus de 1000 tirages) pour en déduire les incertitudes sur leur valeur en fonction du paramètre complexe.

Facultatif : vous pouvez changer les conditions initiales du problème en donnant la valeur `true` au paramètre `Generator::use_constant_continuation`. Vous devriez alors voir des valeurs propres échanger leurs valeurs respectives créant des distributions multimodales.