SIMULATION ET OPTIMISATION **ISC - HEIG-VD**
J.-F. Hêche, E. Rappos **Automne 2025-2026**

Travail pratique **Simulation de Monte-Carlo et ruine du joueur**

Objectif

Le but de ce travail pratique est de mettre en place une simulation de Monte-Carlo permettant d'estimer l'espérance d'une performance X et de calculer un intervalle de confiance pour cette espérance dont le seuil de couverture et la demi-largeur sont fixés par l'utilisateur.

Méthodologie

Pour l'essentiel, vous devez programmer la méthode `simulateTillGivenCIHalfWidth` de la classe `MonteCarloSimulation`. Cette méthode doit permettre de simuler une expérience aléatoire donnée (implémentant l'interface `Experiment` dans les sources) jusqu'à ce que la demi-largeur de l'intervalle de confiance au seuil $1 - \alpha$ (`level` dans les sources) pour la mesure de performance retournée par l'expérience passe en dessous d'une valeur maximale fixée.

La méthode `simulateTillGivenCIHalfWidth` prend sept paramètres :

- ▷ l'expérience `exp` à simuler ;
- ▷ le seuil `level` $(1 - \alpha)$ de couverture de l'intervalle de confiance pour la performance asspociée à l'expérience ;
- ▷ la demi-largeur maximale `maxHalfWidth` ($\Delta_{\max}$) de l'intervalle de confiance produit ;
- ▷ le nombre `initialNumberOfRuns` (N_{init}) de réalisations à générer initialement ;
- ▷ le nombre `additionalNumberOfRuns` (N_{add}) de réalisations supplémentaires à générer tant que la borne $\Delta_{\max}$ pour la demi-largeur de l'intervalle de confiance n'est pas atteinte ;
- ▷ une source aléatoire `rnd` à passer à l'expérience à simuler ;
- ▷ un objet de type `StatCollector` permettant d'accumuler les résultats simulés et de calculer les estimateurs de base.

Afin d'obtenir un intervalle de confiance au seuil $1 - \alpha$ dont la demi-largeur ne dépasse pas la valeur $\Delta_{\max}$, la démarche suivante doit être utilisée :

1) On commence par réaliser N_{init} simulations de l'expérience (à l'aide de la méthode `simulateNRuns`).
2) À partir des données récoltées on calcule une estimation du nombre N de réalisations à générer afin d'obtenir un intervalle de confiance au seuil $1 - \alpha$ dont la demi-largeur ne dépasse pas $\Delta_{\max}$ (voir la page 48 du cours). Cette valeur de N est ensuite arrondie, vers le haut, au plus proche multiple de N_{add}.
3) La simulation est **poursuivie** jusqu'à atteindre N réalisations de l'expérience.

4) Si la demi-largeur de l'intervalle de confiance, calculée sur la base de ces N réalisations, est inférieure ou égale à $\Delta_{\max}$ le processus s'arrête. Sinon N_{add} simulations supplémentaires sont effectuées avant de recalculer un nouvel intervalle de confiance et de retester la condition d'arrêt. Ce processus est répété jusqu'à ce que la condition d'arrêt soit satisfaite.

Expérience à simuler : La ruine du joueur

Afin de tester votre mise en œuvre vous ajouterez à votre projet une classe implémentant l'interface `Experiment` et permettant de simuler l'expérience de Bernoulli suivante :

1) Un joueur dispose d'une fortune initiale de F francs.
2) À chaque partie il mise un franc. Avec probabilité p il gagne deux francs (sa mise et un franc de gain), avec probabilité $q = 1 - p$ il perd sa mise.
3) Le jeu s'arête en cas de ruine du joueur ou dès qu'il arrive à doubler sa fortune initiale.

Expériences à réaliser

Vous compléterez votre programme principal afin de calculer une estimation et un intervalle de confiance pour la probabilité $P(\text{doubler})$ de doubler sa fortune en commençant le jeu avec F francs et en ayant une probabilité p de gagner à chaque partie.

Vous commencerez par calculer un intervalle de confiance au seuil $1 - \alpha$ dont la demi-largeur ne dépasse pas 10^{-4} puis vous recommencez le processus à deux reprises en divisant, à chaque fois, par deux la demi-largeur maximale souhaitée.

Votre programme affichera pour chaque expérience, au minimum, l'estimation de la probabilité de doubler sa fortune, le nombre de réalisations générées en tout, la demi-largeur de l'intervalle de confiance (et l'intervalle de confiance lui-même si vous voulez) et le temps de calcul nécessaire.

Les valeurs des différents paramètres à utiliser sont les suivantes :

- ▷ seuil de confiance `level` $= 1 - \alpha = 95\%$;
- ▷ demi-largeur maximale initiale `maxHalfWidth` $= \Delta_{\max} = 10^{-4}$;
- ▷ nombre initial de réalisations `initialNumberOfRuns` $= N_{\text{init}} = 10^6$;
- ▷ nombre de réalisations supplémentaires `additionalNumberOfRuns` $= N_{\text{add}} = 10^5$;
- ▷ fortune initiale : $F = 5$ francs ;
- ▷ probabilité de gain à chaque partie : $p = 18/37$;
- ▷ graine du générateur de nombres pseudo-aléatoires : `0x1350185` (à utiliser au début de chacune des trois simulations).

Modalités et délais

- ▷ Le travail de programmation est à effectuer par groupe de deux, en Java, version 23.
- ▷ L'archive contenant les sources à compléter, est disponible sur le site Cyberlearn du cours.

- ▷ Vous devez rendre une archive (au format `zip`) contenant toutes les sources de votre projet complété. Vous prêterez une attention toute particulière aux commentaires de votre implémentation de la méthode `simulateTillGivenCIHalfWidth`. Votre archive contiendra également un fichier texte (ou markdown) contenant les résultats pour les trois simulations demandées.
- ▷ Vous devez rendre votre travail sur Cyberlearn au plus tard le **dimanche 2 novembre 2025** (avant minuit).