Fiche 3 – Simulation de modèles stochastiques et Méthode de Monte-Carlo

«Rien à signaler sergent ? Si mon capitaine, le simulateur est mort cette nuit à deux heures.» Humour militaire.

1 Introduction

De nombreuses approches peuvent être mises en œuvre pour exploiter un modèle mathématique d'une situation réelle et étudier ses propriétés. Il est parfois possible de calculer explicitement les quantités auxquelles on s'intéresse au moyen de formules fermées, ou tout au moins de calculer celles-ci numériquement, de manière exacte si l'on exclut les questions de précision finie des calculs sur ordinateur, pour des valeurs fixées des paramètres. Ce n'est malheureusement possible, la plupart du temps, que pour des modèles très simplifiés, possédant par exemple de nombreuses propriétés d'invariance et de symétrie. On peut éventuellement obtenir des résultats du type précédent, mais dans un cadre asymptotique, caractérisant le comportement des quantités étudiées lorsque certains paramètres tendent vers une valeur limite (par exemple, lorsque la taille du système modélisé, ou l'échelle de temps considérée, tend vers l'infini, ou encore, lorsque certains paramètres correspondant à des interactions tendent vers zéro,...). Ou encore, on peut étudier des approximations (plus ou moins bien contrôlées) du modèle initial, qui se prêtent mieux à des approches du type précédent. Enfin, des résultats plus qualitatifs sur le comportement du modèle peuvent parfois être obtenus.

La simulation, qui consiste à reproduire artificiellement le fonctionnement du modèle étudié, constitue l'une des approches importantes permettant d'exploiter celui-ci. Elle permet notamment de valider ou d'invalider des hypothèses[1], d'obtenir des informations quantitatives (qui peuvent venir affiner des informations qualitatives si l'on en possède), de valider certaines approximations, d'évaluer la sensibilité d'un modèle à certaines hypothèses ou à certains paramètres, ou tout simplement d'explorer le comportement d'un modèle lorsque celui-ci est mal connu ou mal compris.

[1] Il peut par exemple s'agir de valider des hypothèses de modélisation, ou plus généralement un modèle dans son ensemble, en comparant le comportement obtenu par simulation au comportement du système réel que l'on cherche à modéliser ; il peut également s'agir de valider des hypothèses sur le système lui-même, dans la mesure où l'on considère que le modèle le représente convenablement.

La simulation de modèles **stochastiques** nécessite le recours à des nombres pris au hasard, et est connue sous le nom générique de méthode de Monte-Carlo (par référence aux jeux de hasard des casinos). De nombreux problèmes numériques *a priori* sans rapport aucun avec le hasard ou les phénomènes aléatoires (évaluation d'intégrales, résolution de systèmes linéaires ou d'équations aux dérivées partielles) peuvent cependant, de manière plus ou moins artificielle, être traduits en termes de modèles stochastiques, et la portée des méthodes de Monte-Carlo dépasse donc très largement le cadre de la modélisation de phénomènes aléatoires (voir par exemple les ouvrages [ES], [F], [LPS], [M], [Y]).

Nous avons dans les deux fiches précédentes abordé le problème de la simulation d'objets aléatoires relativement simples : familles i.i.d. de variables aléatoires unidimensionnelles (on dit également univariées), discrètes ou continues. Un modèle stochastique est en général spécifié au moyen de multiples variables aléatoires de ce type, présentant entre elles des relations de dépendance complexes qui traduisent la façon dont les différents éléments aléatoires composant le modèle interagissent. Selon la manière exacte dont le modèle est spécifié, il est plus ou moins aisé de mettre en œuvre les techniques abordées dans les fiches précédentes, l'utilisation de techniques plus sophistiquées (telles que les méthodes de Monte-Carlo par chaînes de Markov) étant parfois incontournable. Cette fiche aborde la question de la simulation stochastique dans un cadre assez général, les techniques et les applications plus spécialisées (Monte-Carlo par chaînes de Markov, résolution approchée d'e.d.p.) étant destinées à être traitées ailleurs. Elle contient essentiellement des recommandations pratiques concernant les points importants à prendre en considération lors des étapes de conception, de vérification, et d'exploitation des simulations. Une excellente référence bibliographique sur la simulation, et la modélisation en général est le classique de Law et Kelton [LK]. Nous ne saurions trop recommander la lecture de cet ouvrage afin d'approfondir ce cours.

2 Conception, vérification et exploitation des programmes de simulation

De nombreux outils logiciels sont disponibles pour écrire des programmes de simulation. L'utilisation d'un langage compilé (tel que C, Java, Fortran) garantit

en général une exécution très rapide, ce qui peut être crucial pour effectuer des simulations d'une certaine envergure, et permet de contrôler le fonctionnement de la simulation dans tous ses détails. En revanche, la conception du programme, l'écriture du code, sa vérification et la correction des erreurs qu'il contient constituent des tâches lourdes et parfois délicates. L'utilisation de logiciels spécialisés de simulation (voir par exemple l'ouvrage de [LK] pour la description de plusieurs d'entre eux), par exemple pour la simulation des systèmes à événements discrets, peut rendre la mise en œuvre d'une simulation beaucoup plus facile, au prix d'un moindre contrôle des opérations effectuées, et de limitations éventuelles de capacité (sans oublier le coût d'achat du logiciel et le nécessaire apprentissage de son fonctionnement). Enfin, l'utilisation de logiciels de portée générale comme Matlab ou R dans ce contexte est surtout adaptée à un usage pédagogique[2]. Dans ce qui suit, nous donnons quelques recommandations courantes (et minimales) concernant la conception, la vérification et l'exploitation des programmes de simulation. Une grande partie d'entre elles ne sont en fait pas propres à la simulation, mais devraient également s'appliquer de manière générale à l'écriture de programme et à l'exploitation de données.

Insistons sur le fait que ces recommandations ne sont pas vraiment facultatives, et que les mettre en œuvre n'est pas une perte de temps ou d'énergie, mais plutôt un gain, par rapport au temps et à l'énergie nécessaires pour :

- rattraper des erreurs de conception une fois le codage entamé ;
- détecter, localiser et corriger des erreurs de programmation, une fois le code écrit ;
- rattraper des erreurs de principe dans l'exploitation des simulations, une fois celles-ci mises en œuvre.

Ne pas respecter ces quelques prescriptions, c'est tout simplement ne pas accomplir correctement le travail de développement et d'exploitation d'une simulation, et priver de toute crédibilité, si ce n'est de validité, les résultats obtenus.

Le schéma est donc le suivant : analyse, écriture du code, vérification, puis exploitation de celui-ci, l'exploitation pouvant mettre au jour des erreurs qui nécessitent de reprendre l'écriture, puis la vérification, puis l'exploitation. Par ailleurs, si la simulation est utilisée pour valider un modèle, il se peut que celui-ci soit modifié en

[2]Insistons sur le fait qu'une utilisation efficace de ces logiciels pour effectuer des tâches itératives doit autant que possible bannir les boucles, et exploiter à la place les fonctions vectorielles disponibles.

fonction des résultats obtenus, et conduise donc à reprendre la totalité du processus.

2.1 Simulation de systèmes à événements discrets

Les systèmes à événements discrets (tels que les files d'attente), qui modélisent des phénomènes se déroulant en temps réel, mais dont les évolutions se produisent en des instants ponctuels, constituent une classe importante de modèles stochastiques, et interviennent dans de nombreux domaines d'application (voir [LK] et les références qui s'y trouvent), la simulation étant souvent un moyen privilégié pour leur étude.

Une méthode générale pour concevoir et réaliser ce type de simulation est celle du **déroulement en temps (pseudo-)réel**. Concrètement, cela signifie que la simulation se déroule en faisant «sauter» un compteur de temps, d'un événement affectant l'évolution du système à celui qui le suit immédiatement, tout en mettant à jour à chaque «saut» les différentes variables caractérisant l'état du système, ainsi que les différents compteurs statistiques (qui enregistrent des informations annexes sur l'évolution du système). Ceci nécessite de tenir à jour une **liste d'événements** contenant, pour chaque type d'événement possible (par exemple l'arrivée et la sortie d'un élément dans une file d'attente constituent deux types différents d'événements), la liste des dates des prochains événements programmés de chacun des types, et de s'assurer qu'un événement est toujours programmé avant que le compteur de temps dépasse sa date de survenue. Un travail d'analyse préalable est donc nécessaire pour déterminer correctement les variables d'état du système, les différents types d'événements, les différents compteurs statistiques, et les diverses procédures de mise à jour et de programmation des événements. En plus de s'appuyer sur une structure naturelle vis-à-vis du type de système simulé, cette méthode a le gros avantage d'éviter les passes multiples et les retours en arrière, déroulant en une seule fois la totalité de l'évolution du système (comme un film !) et calculant simultanément la totalité des quantités auxquelles on s'intéresse. De plus, la possibilité de mettre à jour en temps réel les diverses variables et compteurs statistiques limite considérablement l'espace mémoire nécessaire à la simulation. Dans certains cas, il peut toutefois être avantageux de ne pas respecter à la lettre cette méthode afin de pouvoir tirer parti de propriétés particulières au système considéré.

Un certain nombre de logiciels permettent d'effectuer des simulations de ce type à partir d'une spécification conviviale du modèle à simuler, avec les avantages (tels que

la simplicité et la facilité d'utilisation, l'évitement de la lourde tâche de conception, de programmation et de vérification de la simulation) et les inconvénients (contrôle beaucoup moins détaillé de la simulation, impossibilité de mettre en place certains raffinements ou certaines optimisations) que cela présente.

2.2 Concernant l'élaboration du code

Le meilleur conseil général qui puisse être donné est de réfléchir au préalable : avant l'écriture du code proprement dite, il est indispensable de réfléchir aux structures de données importantes, aux paramètres, à la structure globale du programme et aux procédés de programmation qui pourront être employés, en ayant en tête les contraintes, et les objectifs poursuivis, ainsi que les extensions possibles (une fois le code rédigé, il sera trop tard pour rattraper les oublis ou erreurs dus à une absence de réflexion préalable, tels que la non-prise en compte de certains cas de figure). Ainsi, il faut avoir une idée précise de la logique et du principe de fonctionnement du programme que l'on projette d'écrire, et la spécifier avec un minimum de détail de façon à pouvoir la vérifier avant de commencer la rédaction du code proprement dite.

Mentionnons quelques points importants en pratique :

- structurer : pour tout programme de quelque envergure, il est fortement recommandé de structurer le programme en unités plus petites (fonctions, procédures, sous-programmes, méthodes, etc...) pouvant être programmées (et donc vérifiées, et éventuellement réutilisées dans un autre contexte ou réintégrées ailleurs) indépendamment, et contribuant chacune à l'accomplissement d'une partie de la tâche globale du programme ;
- documenter : chaque unité du programme doit être assortie d'une documentation qui spécifie ses conditions d'utilisation (types des arguments, domaines de variation de ceux-ci), son principe de fonctionnement (avec si possible des références), ses éléments principaux d'organisation (variables et structures de données importantes, parties importantes, etc...), les problèmes rencontrés ou les précautions particulières à prendre ;
- commenter : le programme doit également être commenté avec soin. Intercaler régulièrement des commentaires explicatifs sur la signification des lignes ou des petits blocs de code permet de se relire aisément (même bien après l'écriture

du code) ;

- nommer : il est fortement recommandé de donner des noms génériques aux paramètres, plutôt que d'employer directement les valeurs numériques particulières qu'ils sont censés prendre, ce qui facilite les modifications et limite les risques d'erreur. De même, donner aux variables des noms rappelant leur signification dans le contexte améliore la lisibilité du code, et permet d'éviter certaines confusions (i,j,k, ne sont pas nécessairement les meilleurs noms de variables de contrôle de boucles) ;
- dimensionner : le dimensionnement des divers objets (tableaux, types des variables, valeurs limites) ne doit jamais être laissé au hasard, au risque de laisser se produire des dépassements incontrolés de capacité au cours de l'exécution, ou de gaspiller inutilement de l'espace-mémoire (ceci peut être particulèrement important lorsque plusieurs versions du même code sont amenées à fonctionner en parallèle). Si une contrainte de capacité apparaît clairement lors de la conception du programme, il faut mentionner son existence en la documentant précisément, et introduire une procédure de contrôle dans le programme, renvoyant un message d'erreur circonstancié (pour mieux en découvrir la cause) en cas de dépassement, et limitant les dégâts dûs au dépassement, par exemple en interrompant l'exécution. Autant que possible, il est souhaitable de dimensionner à l'avance les différents éléments du programme, mais des expériences peuvent également être menées pour ajuster empiriquement le dimensionnement. De plus, il peut être avantageux d'avoir recours à des techniques d'allocation dynamique (plutôt que statique) lorsque le dimensionnement nécessaire dans le pire des cas dépasse de beaucoup le dimensionnement nécessaire dans les cas typiques de fonctionnement.
- s'assurer de la portabilité : certaines opérations ne sont pas totalement portables d'une machine à l'autre, d'un système d'exploitation à l'autre, etc... et il est donc important de vérifier la portabilité des codes réalisés, ou, tout au moins, de documenter les problèmes qui peuvent être identifiés avec celle-ci.
- programmer de manière simple et transparente : éviter autant que possible les bidouilles et petits arrangements ad hoc, pour laisser le mieux possible transparaître la structure et la logique sous-jacente au code, afin de maintenir une certaine lisibilité, et de ménager la possibilité de corriger, d'améliorer, d'étendre le code, etc... sans avoir à le reprendre à zéro.

2.3 Concernant la vérification

Vérifier un programme, par exemple un programme de simulation, c'est vérifier que celui-ci effectue bien la tâche qu'il est censé effectuer. Soulignons que le simple fait qu'un code soit compilé sans erreurs signalées, ou s'exécute sans signaler d'erreurs d'exécution, n'a en général aucun rapport avec le fait que le programme s'acquitte effectivement de la tâche qui lui est assignée. Ces deux conditions sont bien entendu nécessaires, mais il est indispensable de procéder en plus à une vérification détaillée, sans quoi la validité du programme doit être considérée comme à peu près nulle. De fait, il est quasiment impossible qu'un programme d'une certaine complexité venant d'être codé ne comporte pas de multiples erreurs, de gravités diverses. C'est pourquoi il est nécessaire d'avoir recours à des procédés systématiques visant à prévenir et à déceler la présence d'erreurs.

Il ne faut donc en aucun cas négliger la phase de vérification du code une fois celui-ci écrit pour passer directement à l'exploitation de celui-ci, et il est indispensable de procéder soigneusement plusieurs types de vérification, afin de pouvoir utiliser ou distribuer le code avec un minimum de confiance dans sa validité. Insistons une fois de plus : la phase de vérification du code n'est pas moins importante que les autres étapes du processus de modélisation et de simulation, car elle aussi conditionne totalement la validité des résultats obtenus, et il serait absurde de consacrer beaucoup de temps à la formulation d'un modèle, à la conception et à l'écriture du code, ou à son utilisation, sans s'assurer de la correction de celui-ci. Ceci vaut, plus généralement, pour toute production de code informatique à visée sérieuse.

Soulignons que la première vérification à effectuer est celle du principe, de la logique de fonctionnement de la simulation que l'on compte écrire, avant de se lancer dans l'écriture du code, et en complément de l'analyse préalable à celle-ci. Une erreur à ce stade pourra être particulièrement difficile à rattraper sans avoir à réécrire la totalité du code.

Si l'on utilise un langage compilé, il est nécessaire, une fois le code écrit, de procéder à la compilation (éventuellement par morceaux séparés), et à l'édition de liens du programme. Soulignons que les erreurs qui apparaissent à ce stade sont souvent des erreurs de syntaxe minimes, mais qui doivent être corrigées avec soin, et non pas à toute vitesse (sous peine d'introduire de nouvelles erreurs, parfois plus graves et plus difficiles à déceler.)

Les avertissements donnés par le compilateur (et pas seulement les erreurs) doivent également être examinés, car ils permettent parfois d'identifier une erreur de programmation ou des problèmes potentiels d'éxécution.

Un principe général est de vérifier le programme par modules, avant de vérifier son fonctionnement global. On vérifie donc séparément le fonctionnement des différentes unités constituant le programme. Si une unité fait appel à d'autres unités non-testées (ou non encore codées), on peut remplacer ces unités par des unités «triviales», n'accomplissant aucune action, ou renvoyant une valeur par défaut (compatible avec les spécifications).

Pour chacun des modules, et pour le programme global (qu'il faut également vérifier en tant que tel, une fois que tous ses modules ont été vérifiés), voici quatre méthodes complémentaires de vérification possibles. La première, et non la moindre, consiste à vérifier soigneusement le code. Les trois suivantes reposent sur des exécutions contrôlées[3] du programme. On peut (et doit) effectuer les vérifications suivantes :

- relecture détaillée du code : relire ligne-à-ligne, sans complaisance, et si possible à plusieurs, l'intégralité du code, en ne passant à la ligne suivante que lorsque l'intégralité des relecteurs est convaincue du caractère correct de la ligne courante ;
- vérification «globale» : exécuter le programme (ou le module testé) pour différentes valeurs des entrées, en affichant (ou en enregistrant) toutes les quantités importantes (compteurs divers, variables d'état, etc...), et s'assurer que le comportement de ces quantités correspond globalement à ce qui est attendu : ordre de grandeur, signe et sens de variation, symétries, recoupements élémentaires ; parfois, certaines quantités peuvent être calculées exactement ou approximativement, au moins pour certains types d'entrées, et peuvent également servir à vérifier la cohérence des résultats obtenus ;
- vérification «locale» : exécuter le programme (ou le module testé) pour différentes valeurs des entrées, en affichant (ou en enregistrant) toutes les quantités

[3] Au sens où le bon déroulement de l'exécution du programme est contrôlée, mais également au sens où l'on peut utiliser des entrées contrôlées, en lieu et place des entrées normalement utilisées par le programme (par exemple des nombres pseudo-aléatoires), ce qui permet de placer le programme dans des conditions de fonctionnement choisies à l'avance pour vérifier son comportement dans ces conditions.

importantes (compteurs divers, variables d'état, etc...), et reproduire manuellement le déroulement de l'algorithme censé être mis en œuvre par le programme, dans les mêmes conditions, en s'assurant que l'on obtient **exactement** les mêmes résultats pour toutes les quantités considérées ;

– vérification dans des conditions limites : placer le programme dans les conditions limites où celui-ci est censé fonctionner (valeurs égales à zéro ou proches de zéro, égales à leurs limites supérieures ou inférieures de variation, quantités toutes égales entre elles,...), et plus généralement dans les cas non-typiques.

On ne peut se dispenser d'utiliser aucune des méthodes de vérification décrites ci-dessus. Chacune doit être mise en application, en tentant de la rendre à chaque fois la plus efficace possible. En particulier, lors de la relecture du code, il ne faut pas hésiter à faire des figures pour représenter le déroulement du programme, et des vérifications numériques approfondies sur des exemples en cas de doute ; penser également à vérifier de manière systématique le comportement du programme dans les conditions d'initialisation, d'entrée et de sortie de boucles.

En ce qui concerne les vérifications de fonctionnement du programme (qui ne portent pas directement sur le code, mais sur les résultats produits par son exécution), la redondance des quantités affichées n'est pas un inconvénient mais un avantage, car elle permet de vérifier davantage d'éléments de fonctionnement du programme. D'autre part, une vérification globale ne saurait dispenser d'une vérification locale : il est fréquent qu'un programme incorrect présente, dans des conditions moyennes, un comportement à première vue indiscernable d'un programme satisfaisant. Qui plus est, pour toute vérification, il faut s'assurer que les entrées utilisées pour le test conduisent effectivement le programme à passer effectivement dans tous les régimes de fonctionnement prévus (par exemple, pour un programme simulant une file d'attente, il faut s'assurer que le programme gère correctement l'apparition d'une file, son évolution, sa disparition, sa réapparition ultérieure, les instants où il n'y a aucun client, etc... et non pas simplement, par exemple, le cas où tous les clients sont immédiatement servis, sans qu'il y ait création de file d'attente), et que les différents éléments qui constituent le programme soient tous effectivement testés par les vérifications effectuées (par exemple, il ne suffit pas de tester la cohérence des résultats obtenus avec un unique compteur portant sur les temps d'arrivée des clients). Enfin, il est parfaitement possible qu'un programme se comporte correctement dans des conditions moyennes, mais que les conditions limites de fonctionnement ne soient pas

correctement gérées. C'est pourquoi il est indispensable de vérifier également le fonctionnement du programme dans ce type de conditions. Il est utile de réfléchir *a priori* aux conditions les plus susceptibles de gêner le bon fonctionnement du programme (par exemple, mais pas seulement, les cas de figure que l'on n'est pas absolument certain d'avoir pris explicitement en compte dans l'élaboration du programme). Plus globalement, il est nécessaire de tester le code sur des exemples répétés, et d'une certaine envergure. Ainsi, On pourra parfois, même dans des conditions normales de fonctionnement, faire apparaître des situations non-correctement prises en compte dans le code. Bien entendu, il est préférable de graduer l'envergure des tests menés, en commençant par de petits cas faciles et rapides à traiter et à vérifier, et susceptibles de révéler rapidement les erreurs les plus grossières.

Quoiqu'il en soit, notons que toute bizarrerie, anomalie, incohérence ou différence constatée entre le fonctionnement souhaité et le fonctionnement observé, si minime soit-elle, doit être prise au sérieux, et reflète très probablement une erreur, qui doit être systématiquement recherchée. Même si elle n'affecte que peu les résultats dans les circonstances ou elle est observée, une telle erreur peut avoir des conséquences dramatiques dans des circonstances même légèrement différentes. Un programme fonctionnant « à peu près », ou « la plupart du temps », ou encore « dans des conditions moyennes»[4] est inacceptable !

2.4 Sources d'erreurs les plus fréquentes

De multiples raisons peuvent faire que l'exécution d'un programme ne produit pas le résultat souhaité. Du côté de l'élaboration du code, de nombreux types d'erreurs peuvent être rencontrés. Pour n'en citer que quelques-uns :

- les erreurs typographiques (par exemple une parenthèse fermée au mauvais endroit), qui ne conduisent pas toujours à des erreurs de syntaxe, et ne sont donc pas toujours décelées par le compilateur ;
- les confusions d'objets et les erreurs d'analyse légères (par exemple l'emploi d'une variable à la place d'une autre, ou d'une même variable pour deux usages différents, ou encore l'utilisation d'une boucle finissant en n+1 au lieu de n) ;

[4]Il existe des programmes dont la conception même entraîne qu'ils peuvent ne pas fonctionner avec une certaine probabilité, ou dans certains cas particuliers, mais, dans ces cas, il s'agit d'une situation normale, attendue, et documentée.

– les erreurs de conception, liées, par exemple, à une mauvaise compréhension du modèle à simuler, ou de la définition exacte des indicateurs à calculer, ou encore à une traduction erronnée de celui-ci en algorithme ;

De plus, les circonstances dans lesquelles le code a été écrit (par exemple, des modifications trop rapides d'un code déjà existant, ou des utilisations hasardeuses de morceaux de code importés et/ou incorrectement documentés) peuvent favoriser la présence d'erreurs diverses.

Tout ce qui précède est très classique ; insistons sur un autre type d'erreurs : les erreurs liées aux conversions, aux passages d'arguments, aux problèmes de formatage des données, aux dépassements de capacité, à une mauvaise prise en compte des domaines de variation des variables et de leurs valeurs limites. Il est indispensable de prendre en considération ces questions lors de l'élaboration du programme, car les erreurs qui y sont liées peuvent avoir des conséquences particulièrement lourdes, et s'avérer, de plus, difficiles à déceler et pénibles à localiser. Introduire dans le programme des vérifications systématiques portant sur le domaines dans lequels varient les quantités qu'il manipule peut être utile. Il faut également se documenter sur la gestion des conversions et la représentation par la machine des différents objets et variables manipulés. Enfin, même si elles ne font pas partie à proprement parler des erreurs de programmation, les conséquences des erreurs d'arrondi accumulées doivent également être prises en considération lors de l'écriture de programmes manipulant des données numériques.

D'autres erreurs peuvent être liées à des problèmes logiciels (avec le compilateur, le système d'exploitation, par exemple lorsque la portabilité du code ou de parties du code n'est pas assurée), ou même matériels. Cependant, l'écrasante majorité des erreurs proviennent du code lui-même, et c'est donc dans le code qu'il faut les rechercher en priorité.

Un autre exemple d'erreur parfois rencontré est lié à une mise en œuvre incorrecte de la répétition des simulations : le code correspondant à une unique simulation est correct, mais certaines quantités ne sont pas correctement réinitialisées ou mises à jour d'une simulation à l'autre (par exemple, il ne faut en aucun cas réinitialiser la graine du générateur, mais il faut réinitialiser les variables locales utilisées pour une simulation individuelle). Par ailleurs, la définition exacte et la mise en œuvre correcte des critères d'arrêt employés dans des simulations de systèmes évoluant dans le temps peuvent également soulever des difficultés.

2.5 Méthodes de localisation d'erreurs

Une fois qu'a été détectée une erreur, par exemple à l'aide de l'une des méthodes précédentes, il faut encore localiser, puis corriger l'erreur en question. Soulignons quelques recommandations générales à ce sujet. Tout d'abord, la principale méthode de localisation d'erreurs est, bien entendu, l'exécution pas-à-pas, avec affichage des quantités importantes et suivi détaillé du déroulement du programme (dans quelle boucle passe-t-on ? quelle condition teste-t-on ? etc...) Il est alors très utile de pouvoir réexécuter le programme dans des conditions exactement identiques à celles ayant produit l'erreur que l'on a détecté (plutôt que d'attendre qu'elle se reproduise dans des conditions différentes)[5], et de disposer d'un système interactif de débuggage (qui permet de modifier interactivement les valeurs de certaines variables, de vérifier la valeur de certaines d'entre elles, dont on n'avait pas initialement envisagé de suivre l'évolution, etc...). En répétant et en affinant le procédé, on parvient en général à localiser de plus en plus précisément (quitte à revérifier en détail certaines parties du code) le problème. La nature de l'erreur observée, la connaissance de la structure du programme et de la façon dont il a été codé, ainsi qu'une certaine expérience de la programmation et des erreurs les plus couramment rencontrées, peuvent contribuer à orienter efficacement cette recherche. Soulignons enfin qu'il est nécessaire, une fois l'erreur localisée, de réfléchir attentivement et posément à la correction que l'on apporte, afin de ne pas remplacer l'erreur détectée par une autre (qui ne sera pas forcément de même nature, et n'affectera pas nécessairement le fonctionnement du programme de la même façon). De plus, une erreur peut en cacher une autre, et un programme comporte en général plusieurs erreurs (il peut être intéressant de se demander si l'erreur que l'on vient de corriger peut effectivement être à l'origine du problème constaté.) Pour (au moins) ces deux raisons, il faut à nouveau tester le programme de manière approfondie une fois qu'une erreur a été corrigée (insistons une fois de plus : une correction erronnée peut en apparence résoudre le problème constaté, mais en faire apparaître un autre, de nature complètement différente.)

[5] Les générateurs algorithmiques de nombres pseudo-aléatoires possèdent justement cet avantage de reproductibilité, puisque les suites qu'ils produisent sont entièrement déterminées par la graine.

2.6 Exploitation des programmes de simulation

La simulation peut être définie comme une expérimentation numérique menée à l'aide d'un modèle. À ce titre, les méthodes d'exploitation des programmes de simulation partagent nombre de traits communs avec les méthodes classiques de planification et d'exploitation d'expériences (nous renvoyons aux références présentées dans [LK] pour une discussion de ces méthodes). Cependant, la simulation, et plus particulièrement la simulation stochastique, présente un certain nombre de spécificités qui méritent d'être soulignées. Entre autres :

- les contraintes physiques et de coûts liées à l'expérimentation sur des systèmes réels sont totalement absentes, les seules limites étant le temps de calcul et les capacités de stockage (et éventuellement le temps et le coût nécessaires pour développer la simulation) : il est possible par simulation, d'étudier des systèmes qui n'existent pas encore (par exemple, une modification projetée d'un système existant), de contrôler complètement les conditions dans lesquelles le système évolue (et donc d'étudier le comportement du système dans des conditions hypothétiques, difficiles ou dangereuses ou coûteuses à produire dans la réalité), d'expérimenter sur des échelles (de temps, d'espace, de taille des systèmes,...) arbitraires, d'avoir accès à une information complète sur l'état du système (tandis qu'il peut être extrêmement difficile en pratique de mesurer certaines grandeurs), d'interrompre l'évolution du système ou de revenir en arrière dans celle-ci, pour modifier la valeur de certains paramètres puis la faire reprendre ;
- la simulation n'est qu'une expérimentation portant sur un modèle d'un système réel, et non pas sur la réalité. Par conséquent, la validité des conclusions qui peuvent en être tirées est conditionnée non seulement par le caractère correct du programme de simulation vis-à-vis du modèle, mais essentiellement par la fidélité avec laquelle le modèle décrit la réalité : le domaine de validité du modèle, et la précision des informations qu'il est possible d'en extraire sont des éléments primordiaux à prendre en considération pour exploiter correctement les résultats de simulation. Souvent, la mise en œuvre d'importants moyens de calcul et l'apparence scientifique des résultats obtenus (par ordinateur, avec de nombreuses décimales) font perdre de vue cette limitation fondamentale, et conduisent à accorder une confiance excessive (voire illimitée) aux résultats de simulation, en oubliant d'évaluer la qualité du modèle employé, ce qui est

aberrant[6] ! La validation d'un modèle n'a rien à voir avec la vérification d'un code de simulation de ce modèle. La première consiste, en confrontant le comportement du modèle (éventuellement étudié par simulation) à ce qui est connu de la situation modélisée, à accréditer l'utilisation de celui-ci, tandis que la seconde consiste simplement à vérifier l'adéquation entre le code de simulation et le modèle étudié. Rappelons qu'un modèle ayant passé avec succès diverses étapes de validation n'est pas pour autant assuré de fournir des résultats corrects, à plus forte raison s'il est employé dans des conditions ou pour étudier des caractéristiques du système éloignées de celles qui ont servi à sa validation, ou si l'on ne tient pas correctement compte du degré de précision que l'on peut en attendre.

– la simulation de modèles stochastiques a ceci de particulier qu'elle produit en principe, lors de chaque exécution, des résultats aléatoires. Par conséquent, sauf dans certaines situations particulières, **aucune conclusion ne peut être tirée directement d'une seule exécution de la simulation** , et il est nécessaire d'avoir recours à un **grand nombre d'exécutions indépendantes**. On peut ainsi estimer des quantités qui ne dépendent que du modèle, et non pas du hasard propre à telle ou telle simulation. N'effectuer qu'une seule exécution de la simulation et traiter les résultats obtenus comme «les vraies valeurs», ou «les valeurs typiques», par exemple pour effectuer des comparaisons de performances entre deux systèmes différents, ou pour juger de la pertinence des indicateurs employés, est à peu près équivalent au fait de tirer à pile ou face pour décider de la réponse que l'on fournit ! Cette question est discutée en détail dans la partie «Du bon usage de la loi des grands nombres»
– le recours à des procédés de génération de nombres pseudo-aléatoires est également l'un des problèmes importants de la simulation stochastique, qui conditionne la validité des résultats que celle-ci permet d'obtenir (voir la fiche 1).

[6]En particulier, une grave confusion consiste à ne pas distinguer entre la précision des informations numériques obtenues par simulation sur le comportement du modèle, qui dépend notamment de l'importance des moyens de calculs déployés, et la précision des informations que l'on peut espérer en déduire sur le comportement du système réel, qui dépend de la qualité du modèle employé. Améliorer un modèle est une tâche parfois difficile (en tout cas plus difficile que d'augmenter simplement le temps de calcul), mais souvent beaucoup plus efficace que de s'acharner à estimer avec précision le comportement d'un modèle incorrect.

2.7 Indicateurs numériques

Lors d'une simulation, les valeurs d'un certain nombre d'indicateurs numériques empiriques (par exemple, la longueur maximale atteinte par une file d'attente au cours de la simulation, la température atteinte par le système en différents points) caractérisant le comportement du système étudié sont mesurées. Dans le contexte des modèles stochastiques, ces indicateurs empiriques apparaissent comme des variables aléatoires, dont chaque exécution de la simulation fournit une réalisation. **Les valeurs mesurées de ces indicateurs dépendent du hasard de la simulation, et ce sont en fait seulement les lois de probabilité de ces indicateurs** (éventuellement, il peut s'agir de la loi jointe de plusieurs indicateurs), **qui ont un sens dans le cadre du modèle**. Qui plus est, les indicateurs empiriques peuvent eux-mêmes constituer des résumés (par exemple, le temps moyen attendu par un client dans une file d'attente au cours d'une simulation, plutôt que la liste complète des temps d'attente de chacun des clients) de données plus complexes provenant de la simulation.

En plus de s'assurer du caractère correct du code de simulation (les quantités calculées correspondent-elles effectivement à la définition des indicateurs), une exploitation correcte de résultats de simulation suppose de tenir compte et du caractère aléatoire des simulations menées (les résultats que l'on obtient sont des variables aléatoires, un grand nombre de simulations indépendantes est indispensable pour obtenir des informations fiables sur le système), du caractère de résumé qu'ont la plupart des indicateurs numériques employés, et du caractère pertinent des quantités mesurées pour tenter de répondre à la question posée.

De manière générale, soulignons qu'il n'est pas raisonnable d'utiliser un résumé numérique sans s'être au préalable assuré un minimum de la pertinence de l'information qu'il apporte. Par exemple, la moyenne d'une liste de valeurs, ou l'espérance d'une variable aléatoire, peuvent n'apporter que très peu d'information exploitable, et conduire à des interprétations erronées, si on les confond avec d'autres notions telles que : médiane, valeur typique, valeur la plus représentée, ordre de grandeur, avec lesquelles elle ne coïncident pas en général (l'espérance joue cependant un rôle privilégié lorsque les quantités pertinentes apparaissent comme des sommes de variables aléatoires indépendantes et de même loi, et la variance lorsque l'on a affaire à des quantités dont la répartition est proche d'une gaussienne).

L'utilisation et l'assemblage de quantités non directement numériques (données de classement ou de rang, par exemple), ou la quantification arbitraire (de niveaux de satisfaction, par exemple) pour former des indicateurs, est également très problématique, et pas seulement en simulation.

Une préconisation minimale est de toujours examiner (ou explorer) les données brutes à l'aide de divers outils de représentation graphique (tels que histogrammes, boxplots, ACP,...) avant de se lancer dans le calcul et l'exploitation d'indicateurs synthétiques, afin de juger de la pertinence de ceux-ci et de la meilleure manière de les choisir.

Enfin, la simulation d'un modèle est en général destinée à apporter une réponse (ou, tout au moins, une contribution à la réponse) à des questions spécifiques, qui ne sont pas forcément formulées de manière totalement précise. Par exemple, tel système est-il plus efficace, ou plus rentable, que tel autre ? Dans quel système la durée de traitement d'une tâche est-elle la plus faible ?

Répondre à ce type de question est rarement simple, et suppose une réflexion préalable, ainsi qu'une étude approfondie des résultats de simulation.

Au minimum, tous les indicateurs présentés doivent être définis de manière exacte (sous la forme d'une définition mathématique formelle) afin d'éviter les ambiguités quant à ce qui a été calculé, et de permettre un examen critique des résultats obtenus. Cela ne suffit pas. Il faut également assortir ces résultats de marges d'erreur (inévitables lorsque l'on étudie des systèmes stochastiques), et indiquer la procédure (généralement approchée) employée pour obtenir ces marges d'erreur. Cela ne suffit pas. Des indicateurs, même précisément définis, peuvent très bien ne pas donner une image fidèle des résultats de la simulation, et il ne faut jamais confondre la présentation de résultats objectifs (ceux obtenus par simulation) de l'interprétation qui est faite de ceux-ci dans le but de tirer des conclusions. Autant que possible, la pertinence des indicateurs employés doit être étudiée et critiquée, en rapport avec la question posée, et avec les résultats obtenus. Un indicateur pertinent dans un certain contexte, ou un certain domaine de paramètres, pourra perdre de sa validité dans un autre.

Encore faut-il avoir conscience de ces difficultés et ne pas utiliser automatiquement ou à l'aveugle les indicateurs pour tirer des conclusions ou prendre des décisions.

2.8 Fiabilité des estimations obtenues : du bon usage de la loi des grands nombres

Formellement, un indicateur numérique non-empirique portant sur un modèle stochastique $(\Omega, \mathbb{P})$ se met toujours sous la forme de l'espérance d'une quantité aléatoire relativement au modèle. En d'autres termes, il s'agit toujours d'estimer une quantité de la forme :

$$\mathbb{E}(f) = \sum_{\omega \in \Omega} f(\omega)\mathbb{P}(\omega),$$

la formule ci-dessus devant être modifiée lorsque l'on sort du cadre des modèles discrets. On saisit bien ici le caractère limité de l'information apportée par une unique simulation : elle produit un unique $\omega \in \Omega$, variant bien entendu d'une simulation à l'autre, alors que la quantité à laquelle on s'intéresse apparaît comme une somme portant sur l'ensemble des éléments de l'espace Ω. La manière la plus simple de résoudre le problème est de répéter un grand nombre N de fois des simulations indépendantes[7] du modèle, et de calculer la quantité

$$S_N(f) = \frac{1}{N}\left(f(\omega_1) + \cdots + f(\omega_N)\right),$$

chaque ω_i représentant l'élément de Ω produit par la simulation numéro i. C'est la loi des grands nombres qui nous autorise à penser que, pour N assez grand, on aura l'approximation

$$S_N(f) \approx \mathbb{E}(f).$$

2.8.1 Position du problème

La forme forte de la loi des grands nombres affirme que, si $(\omega_i)_{i\geq 0}$ constitue une suite i.i.d. d'éléments de Ω distribués selon $\mathbb{P}$, et si l'espérance $\mathbb{E}(f)$ existe, la convergence suivante a lieu avec une probabilité égale à 1 :

$$\lim_{N \to +\infty} S_N(f) = \mathbb{E}(f).$$

[7] Il faut donc bien vérifier que l'état du générateur de nombres pseudo-aléatoires utilisé à la fin de l'une des simulations est utilisé comme graine de la simulation suivante (une erreur classique est de réinitialiser la graine à la même valeur au début de chaque simulation), sans quoi, l'hypothèse d'indépendance mutuelle des simulations, fondamentale pour l'application de la loi des grands nombres, a toutes les chances d'être prise en défaut.

Le caractère asymptotique de ce résultat limite sa portée pratique : il ne fournit aucune indication quant à la manière de choisir N pour que la variable aléatoire

$$d_N(f) = \left| \frac{1}{N} S_N(f) - \mathbb{E}(f) \right|,$$

qui représente l'erreur commise en estimant $\mathbb{E}(f)$ par $S_N(f)$, soit effectivement inférieure à une limite ϵ donnée. Autrement dit, on ne possède *a priori* aucune estimation de l'erreur d'approximation, et par conséquent **renvoyer simplement les valeurs de $S_N(f)$ calculées par simulation sans tenter d'évaluer l'erreur commise ne donne aucune information utilisable !** Deux remarques importantes concernant cette question : tout d'abord, il est intuitivement clair que plus une variable aléatoire a tendance à fluctuer, plus la valeur de N nécessaire pour que $S_N(f)$ se stabilise autour de sa valeur moyenne doit être élevée. Par conséquent, il ne peut y avoir de valeur de N garantissant une faible valeur de d_N quelle que soit f. D'autre part, sauf cas dégénéré, il n'existe en général aucune valeur de N (même $N = 10^{10^{10^{10}}}$) telle que d_N soit inférieur à ϵ de façon certaine. La variable aléatoire d_N a toujours une probabilité non-nulle de dépasser la valeur ϵ, aussi grand N soit-il. En revanche, lorsque N est grand, cette probabilité est extrêmement faible : c'est ce qu'affirme la loi faible des grands nombres. En pratique, on se contente donc de choisir une valeur de N suffisament grande pour que la probabilité que d_N dépasse ϵ soit très faible. On obtient ainsi une estimation de $\mathbb{E}(f)$ qui n'est fiable (c'est-à-dire approchée à $\pm\epsilon$ près) qu'avec une forte probabilité, le principe même de l'estimation par simulation ne permettant pas de dépasser cette limitation. La valeur à partir de laquelle on décide qu'une probabilité est suffisament faible pour pouvoir en pratique être considérée comme négligeable dépend bien entendu du contexte, et des conséquences qu'entraînerait une mauvaise évaluation de $\mathbb{E}(f)$[8].

Pour résumer, la question pratique qui se pose à présent est :

(1) étant donnés $\epsilon > 0$ et $\alpha > 0$, comment choisir N de telle façon que
$$\mathbb{P}(d_N \geq \epsilon) \leq \alpha \,?$$

Éventuellement, on se la posera sous la forme légèrement différente suivante :

(2) étant donnés N et α, pour quelle valeur de ϵ peut-on affirmer que
$$\mathbb{P}(d_N \geq \epsilon) \leq \alpha \,?$$

[8]En particulier, il n'y a aucune raison de choisir $\alpha = 5\%$ sans prendre en considération le contexte.

Ne pas se poser ce type de question, en se contentant d'extrapoler à une valeur de N finie le résultat asymptotique qu'est la loi forte des grands nombres, est hasardeux et inacceptable, d'autant plus que la méthodologie classique exposée dans la partie suivante permet facilement, quasiment sans accroître le coût de calcul de la simulation, d'y apporter des réponses.

2.8.2 Erreur relative – erreur absolue

Rappelons, à toute fins utiles, que l'on distingue en général deux types d'erreurs d'approximation d'une quantité q par une autre $\hat{q}$. L'erreur absolue est tout simplement la valeur absolue de la différence, c'est-à-dire

$$e_{abs.} = |q - \hat{q}|.$$

L'erreur relative rapporte l'erreur absolue à la valeur de la quantité que l'on cherche à estimer :

$$e_{rel.} = \frac{|q - \hat{q}|}{q}.$$

Suivant les cas, l'une ou l'autre définition qui peut être pertinente, les différences entre les deux types d'erreur survenant lorsque q prend des valeurs très faibles ou très grandes en valeur absolue. Par exemple, si l'objectif est de majorer $|q|$ lorsque celle-ci est petite, et non pas d'estimer précisément son ordre de grandeur, on peut se contenter de contrôler l'erreur absolue.

2.8.3 Estimation de l'erreur : la méthodologie classique

La méthode la plus couramment employée pour aborder cette question est basée sur le théorème de la limite centrale. Dans notre contexte, celui-ci affirme que, si la variance de f existe, lorsque N tend vers l'infini, la distribution de probabilité de $\sqrt{N}D_N$ est proche d'une distribution gaussienne de moyenne nulle et de variance égale à la variance de f, $\mathbb{V}(f) = \mathbb{E}\left[(f - \mathbb{E}(f))^2\right]$ (ceci n'a aucun rapport avec le fait que la loi de f soit elle-même gaussienne : le résultat est valable quelque soit la loi de f pourvu sa variance existe). Ainsi, en désignant par t_α l'unique réel positif tel qu'une variable aléatoire gaussienne centrée réduite Y vérifie $\mathbb{P}(Y \in [-t_\alpha, +t_\alpha]) = 1 - \alpha$ (on a par exemple $t_{5\%} \lessapprox 1,96$), on constate que l'on peut choisir

$$\epsilon = t_\alpha \sqrt{\frac{\mathbb{V}(f)}{N}}.$$

La valeur de $\mathbb{V}(f)$ étant la plupart du temps inconnue (rappelons que l'on cherche à estimer $\mathbb{E}(f)$, donc au moins la valeur de cette dernière quantité n'est pas connue exactement), on utilise la simulation pour estimer celle-ci[9] :

$$\mathbb{V}(f)_{estim.} = \frac{1}{N}\sum_{i=0}^{N}\left(f(\omega_i) - S_N(f)\right)^2,$$

et l'on choisit

$$\epsilon = t_\alpha\sqrt{\frac{\mathbb{V}(f)_{estim.}}{N}}.$$

(Il est possible, comme pour la moyenne empirique, de calculer progressivement la variance empirique, au fur et à mesure que les ω_i sont générés, ce qui permet, si on le souhaite, de ne pas avoir à stocker la liste des ω_i.)

Dans ce cadre, il est plus naturel de se poser la question sous la forme (2) ci-dessus, que sous la forme (1), car le choix de N dans (1) nécessite *a priori* la connaissance de $\mathbb{V}(f)$, alors que celle-ci est en général estimée au moyen des N répétitions (bien entendu, il est possible de faire une première estimation de la variance avec un nombre limité de répétitions pour avoir une idée du nombre d'itérations nécessaires pour atteindre une précision fixée, et ainsi répondre à la question (1)).

Pour résumer, dans le cadre de la méthodologie classique, la largeur des intervalles de confiance obtenus dépend de trois facteurs bien distincts :

- le niveau de confiance souhaité $1-\alpha$, par l'intermédiaire de t_α (la loi gaussienne étant universelle dans ce contexte) l'intervalle étant naturellement d'autant plus large que le niveau de confiance demandé est élevé ;
- le nombre N de répétitions indépendantes effectuées N, la largeur de l'intervalle décroissant avec N proportionnellement à $1/\sqrt{N}$;
- les fluctuations de la variable étudiée f, par l'intermédiaire de sa variance. Cette dernière est propre au problème considéré, et doit être évaluée.

Soulignons, que, à quelques approximations discutées dans la partie suivante près, l'intervalle obtenu ci-dessus ne peut être amélioré (c'est-à-dire réduit) : les fluctuations aléatoires de $S_N(f)$ autour de la valeur moyenne $\mathbb{E}(f)$ sont réellement de l'ordre de la taille de l'intervalle ci-dessus. La diminution en $\frac{1}{\sqrt{N}}$ de la taille de l'intervalle

[9] L'estimateur empirique usuel de la variance utilise un facteur $\frac{1}{N-1}$ plutôt que $\frac{1}{N}$ pour assurer un biais nul. Les valeurs de N couramment employées en simulation sont suffisament grandes pour que la différence entre les deux formules ne soit pas significative dans notre contexte (mais elle l'est par ailleurs !).

est une propriété générale des méthodes de Monte-Carlo, qui conditionne de manière systématique la précision qu'il est possible d'atteindre en y faisant appel.

2.9 Raffinement théorique

En général, la méthode décrite ci-dessus pour obtenir des intervalles de confiance n'est pas exacte, et repose (au minimum) sur une double approximation : celle de la loi de $\sqrt{N}d_N$ par une loi gaussienne, et celle de la variance de f par sa valeur estimée. Pour rendre plus rigoureuse cette méthode, et puisque nous cherchons à dépasser la simple extrapolation à des valeurs finies de N de résultats asymptotiques, il convient de s'interroger sur la validité de ces approximations pour des valeurs finies de N.

Concernant l'approximation par une loi gaussienne, on a le résultat suivant (inégalité de Berry-Esséen, voir [S]) :

$$\left| \mathbb{P}\left(\frac{\sqrt{N}}{\sqrt{\mathbb{V}(f)}} d_N \leq \epsilon \right) - \int_{-\epsilon}^{\epsilon} e^{-x^2/2} \frac{dx}{\sqrt{2\pi}} \right| \leq \frac{0,8}{\sqrt{N}} \times \frac{\mathbb{E}(|f - \mathbb{E}(f)|^3)}{\mathbb{V}(f)^{3/2}},$$

qui permet de contrôler la qualité de l'approximation obtenue, à condition de disposer d'informations sur les moments d'ordre 2 et 3 de f. Même des estimations relativement grossières sur f peuvent être utilisées avec succès.

Le cas de l'estimation de la variance de f est plus troublant : afin d'évaluer l'erreur commise en estimant par simulation l'espérance de f, on emploie une estimation de l'espérance de $(f - E(f))^2$ par simulation. Mais comment évaluer à présent l'erreur commise sur l'estimation de l'espérance de $(f - E(f))^2$? Il semble y avoir là un cercle vicieux, que l'on peut en fait éviter au moyen de la remarque suivante : même une erreur relative assez importante sur l'estimation de $\mathbb{V}(f)$ (disons, de 40%) permet de conserver une estimation raisonnable de l'ordre de grandeur probable de la largeur de l'intervalle de confiance, même une erreur absolue assez importante (de l'ordre de quelques unités) sur l'estimation de $\mathbb{V}(f)$ ne modifie que peu (en valeur absolue) la largeur de l'intervalle de confiance, du fait du facteur $\frac{1}{\sqrt{N}}$. Notons que le problème le plus grave est la possibilité de sous-estimer $\mathbb{V}(f)$, qui conduit à proposer un intervalle de confiance trop étroit, et donc à surévaluer la précision de l'estimation de $\mathbb{E}(f)$ obtenue par simulation. Or, moyennant des estimations *a priori* assez grossières sur f, on peut contrôler l'erreur d'estimation commise sur $\mathbb{V}(f)$, et s'assurer en général (mais pas toujours, il faut vérifier les hypothèses !) qu'une erreur relative ou absolue importante sur l'estimation de $\mathbb{V}(f)$ est extrêmement improbable.

On peut par exemple partir de l'inégalité suivante (voir [McD]), valable pour toute famille de variables aléatoires indépendantes $Y_1, \ldots, Y_N$ vérifiant avec probabilité 1 le fait que $a \leq Y_i \leq b$ pour tout i : pour tout $t > 0$,

$$\mathbb{P}\left(|Y_1 + \cdots + Y_N - \mathbb{E}(Y_1 + \cdots + Y_N)| \geq t\right) \leq 2\exp\left[\frac{-2t^2}{N(b-a)^2}\right].$$

En choisissant $X_i = f(\omega_i) - \mathbb{E}(f)$, et en réécrivant $\mathbb{V}(f)_{estim.}$ sous la forme :

$$\mathbb{V}(f)_{estim.} = \frac{1}{N}\sum_{i=0}^{N} X_i^2 - \left(\frac{X_1 + \cdots + X_N}{N}\right)^2,$$

puis en appliquant l' inégalité ci-dessus à $Y_i = X_i$ et $Y_i = X_i^2$, on voit (comment ?) que, si l'on suppose que l'on connaît une borne *a priori* de la forme $a \leq f - \mathbb{E}(f) \leq b$, on a, pour tout $t > 0$

$$\mathbb{P}\left(|\mathbb{V}(f)_{estim.} - \mathbb{V}(f)| \geq t\right) \leq 2\exp\left[-\frac{Nt^2}{2(b-a)^4}\right] + 2\exp\left[-\frac{Nt}{(b-a)^2}\right].$$

Dans le cas où l'on ne dispose pas de bornes sur $f - \mathbb{E}(f)$, on peut avoir recours à d'autres types d'estimations, par exemple de moments d'ordre 4, qui conduisent à d'autres types d'inégalités. De manière générale, plus on dispose d'informations *a priori* sur f, plus on dispose d'inégalités permettant de contrôler précisément l'estimation de $\mathbb{V}(f)$. Bien entendu, on n'étudie en général par simulation que des variables aléatoires sur lesquelles peu d'informations *a priori* sont disponibles, d'où l'intérêt de présenter des inégalités valables dans un cadre très général.

2.10 Raffinement empirique

Dans la prochaine version des notes...

On parlera de Jacknife, de bootstrap, de vérification graphique...

L'idée est de calculer des indicateurs empiriques permettant (parfois) d'alerter l'utilisateur sur le fait que les conditions ne sont pas réunies pour que l'approximation gaussienne utilisée soit fiable...

2.11 Discussion

La méthodologie classique visant à estimer une espérance par simulation permet de calculer, pour un niveau de confiance $1 - \alpha$ donné, des intervalles contenant la valeur recherchée avec une probabilité supérieure à $1 - \alpha$. Cette méthodologie, en

plus de ne produire des estimations vouées à n'être valable qu'avec une certaine probabilité[10], repose sur un certain nombre d'approximations, qui peuvent être validées à condition de disposer au minimum d'estimations *a priori* assez grossières sur la variable aléatoire étudiée. En toute généralité, les intervalles de confiance ainsi obtenus ne sont donc eux-mêmes que des intervalles approchés. Qui plus est, et il s'agit probablement d'une restriction plus importante encore, toute la théorie permettant de produire ces intervalles repose sur la modélisation probabiliste par des variables aléatoires i.i.d. des nombres $f(\omega_i)$ générés au cours de la simulation. Du fait que la simulation repose sur l'emploi de nombres pseudo-aléatoires, rien ne saurait garantir absolument la validité de cette modélisation (voir la fiche 1 pour une discussion de cette question). Les restrictions habituelles concernant les erreurs d'arrondi et les erreurs de programmation/exécution s'appliquent naturellement. Enfin, rappelons que les résultats produits par simulation portent sur un modèle d'un phénomène, et non pas sur la réalité du phénomène elle-même. Leur validité pratique repose donc fondamentalement sur la validité du modèle comme représentation du phénomène. Pour toutes ces raisons, il est nécessaire de prendre avec précaution les résultats produits par simulation, et d'être conscient des diverses sources d'erreur pouvant affecter la validité des résultats obtenus. Les intervalles de confiance obtenus ne sont pas parole d'Évangile !

2.12 Phénomènes stationnaires

De nombreux modèles stochastiques décrivent le comportement de collections de variables aléatoires $(X_t)_{t\in T}$ qui sont en un sens analogues entre elles. Par exemple, $T = \{t_1 < \cdots < t_N\}$ peut représenter une succession d'instants, et $(X_t)_{t\in T}$ les valeurs successives d'une quantité X évoluant aléatoirement avec le temps. Ou encore, $T = \{-N, \ldots, +N\} \times \{-N, \ldots, +N\}$ peut représenter des coordonnées spatiales, et $(X_t)_{t\in T}$ l'ensemble des valeurs prises par une même quantité X en ces différentes coordonnées. La plupart du temps, les variables aléatoires $(X_t)_{t\in T}$ ne forment pas une famille i.i.d., et il n'y a pas de raison *a priori* pour que la variable aléatoire correspondant à la valeur moyenne empirique des $(X_t)_{t\in T}$, c'est-à-dire

$$\frac{\sum_{t\in T} X_t}{|T|},$$

[10] Il est donc normal, lorsque l'on calcule un grand nombre d'intervalles de confiance, qu'une proportion d'environ α d'entre eux ne contiennent pas la valeur que l'on souhaite estimer !

(où $|T|$ représente le cardinal de l'ensemble T) converge vers une quantité constante lorsque $|T|$ tend vers l'infini. Cependant, un certain nombre de modèles possèdent cette propriété : avec probabilité 1,

$$\lim_{|T|\to+\infty} \frac{\sum_{t\in T} X_t}{|T|} = E_\infty(X),$$

$E_\infty(X)$ étant appelée la moyenne stationnaire de la quantité X. On peut alors développer des méthodes analogues à celles reposant sur la loi des grands nombres habituelle pour évaluer les quantités stationnaires telles que $E_\infty(X)$. Certaines méthodes (notamment les méthodes dites de Monte-Carlo par chaîne de Markov) s'appuient sur ce principe, en mettant les quantités d'intérêt sous la forme de quantités stationnaires par rapport à un modèle que l'on peut facilement simuler.

Notez bien cependant la différence qui existe entre le fait d'effectuer une unique simulation du modèle décrivant $(X_t)_{t\in T}$, qui produit en une seule fois la collection de variables aléatoires $(X_t)_{t\in T}$, à partir desquelles on peut former la variable aléatoire $\frac{\sum_{t\in T} X_t}{|T|}$, et le fait d'effectuer N simulations indépendantes d'un modèle, produisant à chaque fois une variable aléatoire $f(\omega_i)$, à partir desquelles on peut former la variable aléatoire $\frac{\sum_{i=1}^{N} f(\omega_i)}{N}$. Une erreur courante consiste traiter la liste $(X_t)_{t\in T}$ (par exemple, la liste des temps d'attente des clients successifs dans une file d'attente) comme une suite de v.a.i.i.d., en appliquant les formules classiques de calcul des intervalles de confiance . On obtient ainsi des valeurs numériques certes précises (avec tout plein de décimales !) qui ne représentent...rien, car les hypothèses permettant d'appliquer la méthodologie classique ne sont pas satisfaites, et les intervalles obtenus sont donc vides de sens.

2.13 Erreurs fréquentes

Voici une petite liste de confusions et erreurs fréquentes (NON-EXHAUSTIVE) :

- confondre validation (du modèle) et vérification (du code de simulation) ;
- oublier que même un modèle ayant passé avec succès les étapes de validation peut être incorrect (en particulier, mais pas seulement, quand on extrapole son fonctionnement) ;
- oublier que même un code de simulation ayant passé avec succès les étapes de vérification peut encore contenir des erreurs ;
- oublier de revalider après des modifications ;

- oublier que la correction de chacune des étapes (récapituler) conditionne la correction de l'ensemble ;
- confondre la précision des estimations que la simulation fournit sur le modèle avec la précision des estimations que celle-ci peut fournir sur des grandeurs réelles (cf. la taille de l'empereur de Chine) ;
- confondre la précision (i.e. le nombre de décimales) d'un nombre fourni par l'ordinateur avec toute autre notion de précision mesurant l'écart entre ce nombre et une quantité donnée que celui-ci est censé approcher ou représenter ;
- oublier de prendre en compte le caractère aléatoire des phénomènes, des modèles, et des simulations (à vous de détailler pour chacune les redoutables conséquences).
- oublier de vérifier le caractère pertinent des indicateurs employés (cf. classer deux compagnies d'aviation en fonctions de leurs taux d'avions arrivant en retard sans tenir compte de la durée de ces retards, qui peut tout changer au classement !)
- avoir omis de valider correctement le calcul des indicateurs.

3 Réduction de variance

D'après ce qui précède, la précision avec laquelle il est possible d'estimer l'espérance d'une variable aléatoire f par Monte-Carlo est essentiellement déterminée par le nombre d'itérations N effectué, et par la variance de f. Lorsque celle-ci est importante, le nombre d'itérations nécessaires pour obtenir une précision sastisfaisante peut se révéler prohibitif. Diverses méthodes, dont l'applicabilité et l'efficacité peuvent dépendre fortement du problème considéré, permettent de réduire, parfois significativement, la variance des estimations obtenues, et donc de gagner d'autant en précision, ou en temps de calcul, en modifiant soit f, soit le modèle $(\Omega, \mathbb{P})$ que l'on simule, soit la méthode d'estimation employée, soit une combinaison des trois (on est obligé de modifier quelque chose, sans quoi la variance de f est fixée, et l'efficacité de la méthode l'est également).

Soulignons que ces méthodes ne constituent pas simplement des «recettes» commodes, mais que certaines d'entre elles reposent sur des idées dont la portée est assez générale dans le domaine de la simulation stochastique. Parfois, l'utilisation d'une méthode de réduction de variance permet de traiter des problèmes qui né-

cessiteraient, en utilisant une méthode naïve, et pour fournir une estimation d'une précision acceptable, un coût prohibitif en temps de calcul. Souvent, on parvient à améliorer très significativement la précision d'un calcul approché. Cependant, la mise en œuvre d'une telle méthode suppose une analyse un peu (voire beaucoup) plus détaillée du problème considéré, et accroît généralement le coût des étapes de simulation. Il y a donc (et là encore, ceci est assez général en simulation, voire en modélisation) un compromis à trouver entre la précision de la méthode et le coût de sa mise en œuvre. La liste qui suit ne prétend pas à l'exhaustivité, mais présente les méthodes dont la portée semble la plus large. Seul le principe général des méthodes est présenté, des exemples étant destinés à être traités en TP.

3.1 Échantillonnage préférentiel («importance sampling» en anglais)

L'espérance de f s'écrit[11]

$$\mathbb{E}(f) = \sum_{\omega\in\Omega} f(\omega)\mathbb{P}(\omega).$$

Partant d'une fonction positive $g : \Omega \to [0, +\infty[$ telle que $g(\omega) > 0$ pour tout $\omega \in \Omega$ tel que $\mathbb{P}(\omega) \neq 0$, on peut réécrire cette espérance sous la forme :

$$\mathbb{E}(f) = \sum_{\omega\in\Omega} \frac{f(\omega)}{g(\omega)} g(\omega)\mathbb{P}(\omega).$$

Si g vérifie en outre $\sum_{\omega\in\Omega} g(\omega)\mathbb{P}(\omega) = 1$, autrement dit, si $\mathbb{E}(g) = 1$, on peut définir une nouvelle probabilité sur Ω par $\mathbb{Q}(\omega) = g(\omega)\mathbb{P}(\omega)$, et l'espérance de f se réécrit (on ajoute un indice inférieur à l'espérance pour préciser par rapport à quel modèle, $(\Omega, \mathbb{P})$ ou $(\Omega, \mathbb{Q})$ celle-ci est calculée, le $\mathbb{E}$ de départ, celui auquel on s'intéresse, est ainsi noté $\mathbb{E}_{\mathbb{P}}$) :

$$\mathbb{E}_{\mathbb{P}}(f) = \mathbb{E}_{\mathbb{Q}}\left(\frac{f}{g}\right).$$

Si, au lieu de simuler le modèle $(\Omega, \mathbb{P})$ pour estimer $\mathbb{E}_{\mathbb{P}}(f)$, on simule le modèle $(\Omega, \mathbb{Q})$ pour estimer $\mathbb{E}_{\mathbb{Q}}\left(\frac{f}{g}\right)$, la variance qui conditionne la précision des estimations est la variance de $\frac{f}{g}$ dans le modèle $(\Omega, \mathbb{Q})$. Ceci suggère de choisir g de façon à rendre

[11] Dans le cas discret. Dans le cas continu, il est immédiat d'effectuer les modifications qui s'imposent.

le rapport $\frac{f}{g}$ le plus constant possible, de façon à atteindre une faible variance. À la limite, dans le cas où f est strictement positive, le choix

$$g = \frac{f}{\mathbb{E}_{\mathbb{P}}(f)}$$

permet de rendre le rapport $\frac{f}{g}$ constant et donc d'atteindre une variance égale à zéro...mais ce choix suppose que l'on connaisse déjà $\mathbb{E}_{\mathbb{P}}(f)$. Pour pouvoir appliquer la méthode, il est nécessaire d'utiliser une fonction g qui réalise un compromis entre le fait de «coller» raisonnablement bien à f, d'être effectivement calculable, et de rendre le modèle $(\Omega, \mathbb{Q})$ raisonnablement facile à simuler. Assez naturellement, plus on dispose d'informations sur la variable aléatoire f, plus il est possible d'améliorer la précision de la méthode. Attention : une erreur fréquente consiste à simuler effectivement une suite de réalisations indépendantes $\omega_1, \ldots, \omega_N$ du modèle $(\Omega, \mathbb{Q})$, mais à calculer

$$\frac{1}{N}\left(f(\omega_1) + \cdots + f(\omega_N)\right)$$

au lieu de

$$\frac{1}{N}\left(\frac{f(\omega_1)}{g(\omega_1} + \cdots + \frac{f(\omega_N)}{g(\omega_N}\right).$$

Ce que l'on cherche à estimer est $\mathbb{E}_{\mathbb{Q}}\left(\frac{f}{g}\right)$ et non pas $\mathbb{E}_{\mathbb{Q}}\left(f\right)$!

3.2 Variable de contrôle

L'idée de la méthode, dans le contexte le plus simple, est qu'il peut être plus aisé, pour évaluer $\mathbb{E}(f)$, d'estimer par simulation l'espérance d'une variable aléatoire différente de f, par exemple h, nommée variable de contrôle. Par exemple, si $\mathbb{E}(f-h)$ est connue, et si la variance de h est inférieure à celle de f. Quoique d'apparence simple, cette idée peut se révéler très efficace !

3.3 Stratification

L'idée est de décomposer l'espace Ω en strates de probabilités connues, et d'échantillonner séparément f dans chacune des strates. On peut ainsi augmenter le nombre de points d'échantillonnages dans les strates où la variance de f est élevée, tout en le diminuant dans celles où la variance est faible.

Formellement, supposons que l'espace Ω soit découpé en une partition $A_1, \ldots, A_p$. On a alors

$$\mathbb{E}(f) = \sum_{i=1}^{p} \mathbb{E}(f|A_i)\mathbb{P}(A_i).$$

En supposant que l'on connaît les probabilités $\mathbb{P}(A_i)$ et que l'on peut échantillonner f séparément sur chacun des A_i, c'est-à-dire simuler des variables aléatoires dont la loi est celle de f conditionnelle à la réalisation de A_i, on peut appliquer la méthode de Monte-Carlo sur chacun des A_i, et regrouper les estimations obtenues pour obtenir une estimation de $\mathbb{E}(f)$. Plus précisément, pour chaque i, on estime $\mathbb{E}(f|A_i)$ par la méthode de Monte-Carlo avec N_i expériences indépendantes. On obtient ainsi une estimation $Y(i)$, de variance $\mathbb{V}(f|A_i)/N_i$, et l'on en déduit l'estimation de $\mathbb{E}(f)$:

$$\sum_{i=1}^{p} \mathbb{P}(A_i)Y(i),$$

dont la variance est égale, en admettant que les expériences menées sur chacun des A_i sont mutuellement indépendantes, à la somme

$$\sum_{i=1}^{p} (\mathbb{P}(A_i))^2 \times \frac{\mathbb{V}(f|A_i)}{N_i}.$$

Si le nombre total d'expériences réalisé est fixé à N, on constate facilement que le choix optimal des N_i, c'est-à-dire celui qui minimise la variance de l'estimation de $\mathbb{E}(f)$, est donné par :

$$N_i = N \times \frac{\mathbb{P}(A_i)\left(\mathbb{V}(f|A_i)\right)^{1/2}}{\sum_{j=1}^{p} \mathbb{P}(A_j)\left(\mathbb{V}(f|A_j)\right)^{1/2}}.$$

Le nombre de points d'échantillonnage qu'il convient d'attribuer à une strate est donc proportionnel à la variabilité de f dans cette strate, mesurée par son écart-type, et au poids de cette strate, ce qui se comprend assez bien intuitivement.

On montre alors que cette variance est inférieure à la variance $\mathbb{V}(f)/N$ que l'on obtiendrait en utilisant une méthode de Monte-Carlo sans stratification. Lorsque, ce qui est le cas en général, cette équation ne conduit pas à des nombres entiers, on peut toujours remplacer les valeurs par leurs parties entières. Un problème plus sérieux est que les variances $\mathbb{V}(f|A_i)$ sont rarement connues. Il est alors éventuellement possible de les estimer à l'aide d'une méthode de Monte-Carlo (de plus faible envergure que

celle que l'on envisage de mener pour estimer I). Si l'on ne connaît que les valeurs de $\mathbb{P}(A_i)$, on peut encore montrer que le choix

$$N_i = N\mathbb{P}(A_i)$$

conduit à une variance inférieure à $\mathbb{V}(f)/N$. Notons qu'en tout cas, il est important de ne pas choisir n'importe comment les N_i, sous peine d'augmenter la variance de l'estimateur de $\mathbb{E}(f)$!

3.4 Conditionnement

L'idée de cette méthode, qui peut s'appliquer dans un contexte beaucoup plus vaste, est que, si l'on sait calculer l'espérance de f conditionnellement à la valeur d'une autre variable aléatoire Z, c'est-à-dire, si l'on connaît une fonction ϕ telle que :

$$\phi(z) = \mathbb{E}(f|Z = z),$$

on peut obtenir en simulant $\phi(Z)$ une estimation de $\mathbb{E}(f)$ de variance inférieure à celle de f. En effet, d'une part :

$$\mathbb{E}(\phi(Z)) = \sum_z \mathbb{P}(Z = z)\mathbb{E}(f|Z = z) = \mathbb{E}(f),$$

et, d'autre part, en posant $\mathbb{E}(f|Z) = \phi(Z) = \sum_z \mathbb{E}(f|Z = z)\mathbf{1}(Z = z)$ et $\mathbb{V}(f|Z) = \sum_z \mathbb{V}(f|Z = z)\mathbf{1}(Z = z)$, on vérifie que

$$\mathbb{V}(f) = \mathbb{V}(\mathbb{E}(f|Z)) + \mathbb{E}(\mathbb{V}(f|Z)).$$

Ainsi, la variance de f s'écrit comme la somme de la variance de $\phi(Z)$ et d'un terme de variance conditionnelle à Z. Plus la variance conditionnelle $\mathbb{V}(f|Z)$ est grande, plus la réduction de variance ainsi obtenue est importante.

3.5 Variables aléatoires communes

Cette méthode est surtout intéressante lorsque l'on souhaite comparer entre eux deux systèmes différents, par exemple en comparant l'espérance d'une variable aléatoire T_1 relative au premier système à l'espérance d'une variable aléatoire T_2 du second système. Plutôt que d'estimer séparément chacune des deux espérances en simulant indépendamment chacun des deux systèmes, et en calculant ensuite la différence des deux estimations obtenues, il peut être intéressant de simuler les deux

systèmes simultanément, en utilisant en partie les mêmes nombres pseudo-aléatoires pour faire évoluer les deux systèmes. Ceci revient, formellement, à écrire T_1 et T_2 comme des variables aléatoires définies sur le même espace de probabilité $(\Omega, \mathbb{P})$. On peut alors estimer directement par simulation l'espérance de $T_1 - T_2$, et le paramètre pertinent est la variance $\mathbb{V}(T_1 - T_2) = \mathbb{V}(T_1) + \mathbb{V}(T_2) - 2cov(T_1, T_2)$. Si T_1 et T_2 sont simulées de manière indépendante, cette variance est égale à $\mathbb{V}(T_1) + \mathbb{V}(T_2)$. Si la simulation simultanée (en utilisant partiellement les mêmes nombres pseudo-aléatoires) de T_1 et de T_2 permet de corréler positivement ces deux variables, on obtient donc une réduction de variance, d'autant plus significative que T_1 et T_2 sont plus corrélées. Intuitivement, si T_1 et T_2 ont un comportement similaire en présence des mêmes entrées pseudo-aléatoires, la corrélation obtenue sera importante. Typiquement on peut s'attendre à ce que deux systèmes différents mais voisins réagissent de manière comparable en présence de conditions de fonctionnement identiques, les conditions de fonctionnement (par exemple les temps d'arrivée de clients dans une file d'attente) étant en général fournies par des nombres pseudo-aléatoires. Cependant la méthode peut conduire à un accroissement de la variance si la corrélation est négative !

4 Méthodes de Monte-Carlo et analyse numérique

Nous avons présenté la méthode de Monte-Carlo dans le cadre, naturel, de la simulation de modèles stochastiques. Cependant, un champ important d'application est la résolution de problèmes d'analyse numérique tels que calculs d'intégrales, résolution d'e.d.p. ou de systèmes linéaires, a priori déterministes (dépourvus de caractère stochastique), mais pour lesquels les quantités d'intérêt peuvent néanmoins se mettre sous la forme d'espérances relatives à un modèle stochastique, et donc permettre d'approcher ces quantités par simulation de ce modèle. Vous pouvez consulter [Y] pour une introduction à différentes méthodes de ce type. (La possibilité d'exprimer ainsi un problème déterministe en termes d'un modèle stochastique n'est pas toujours due au hasard, mais résulte parfois du fait que plusieurs modélisations, par exemple à des échelles différentes, d'un même phénomène, peuvent exister, certaines stochastiques, et d'autres pas). Pour en apprendre (bien) plus sur les liens entre e.d.p. et méthodes de Monte-Carlo, vous pouvez consulter l'ouvrage [LPS] cité dans la bibliographie.

Les méthodes génériques de réduction de variance, ainsi que des méthodes plus spécifiques au problème considéré, peuvent être mises en œuvre.

5 Méthodes de Monte-Carlo par chaînes de Markov

L'idée de cette méthode est que, pour simuler (et ainsi étudier) les propriétés d'une loi de probabilité μ définie sur un ensemble S, on peut tenter de simuler pendant suffisament de pas une chaîne de Markov ergodique sur S et dont la loi invariante est μ. Un exemple très simple de cette méthode est la pratique consistant à battre un jeu de cartes, en effectuant des étapes répétées de transformations aléatoires affectant l'ordonnancement des cartes, dans le but de parvenir à une permutation aléatoire de loi uniforme sur l'ensemble des permutations possibles du jeu de cartes considéré (ce que l'on peut raisonnablement appeler un jeu parfaitement battu).

Une remarque importante est que, lorsque μ n'est connue qu'à une constante près (ce qui est par exemple le cas des mesures de Gibbs en physique statistique, ou de probabilités conditionnelles à des événements complexes) il est possible de définir et de simuler relativement facilement de telles chaînes. Une question cruciale et complexe est alors de déterminer combien de pas de simulation sont nécessaires pour que la loi de la chaîne soit raisonnablement proche de μ, au moins vis-à-vis de l'application projetée de la simulation (en d'autres termes, combien de fois faut-il battre le jeu de cartes pour que celui-ci puisse être considéré comme bien – à défaut de l'être parfaitement – battu).

Voir par exemple [M] pour une présentation simple de la méthode.

6 Intérêt des méthodes de Monte-Carlo

La simulation d'un modèle stochastique semble être une méthode tout-à-fait naturelle pour obtenir des informations sur celui-ci. Cependant, comme le montre par exemple la méthode d'échantillonnage préférentiel, il peut être intéressant de simuler un modèle différent du modèle initial afin d'améliorer la précision des estimations obtenues. Plus généralement, en notant que la simulation se base toujours sur une suite de variables aléatoires i.i.d. uniformes $U_1, U_2, \ldots$, et que l'objectif précis est en général d'estimer l'espérance d'une ou de plusieurs variables aléatoires, on constate que le problème numérique principal que la simulation tente de résoudre est le calcul

de quantités de la forme

$$\mathbb{E}F(U_1, \ldots, U_d),$$

où F est une fonction transformant les d nombres pseudo-aléatoires i.i.d. uniformes employés par la simulation en la variable aléatoire à laquelle on s'intéresse. ce qui peut se réécrire sous la forme

$$\int_{[0,1]^d} F(x_1, \ldots, x_n)\mathrm{d}x_1 \ldots \mathrm{d}x_d.$$

Autrement dit, le problème se ramène au calcul d'une intégrale en grande (en général, une exécution de la simulation nécessite le recours à de nombreuses entrées pseudo-aléatoires) dimension. Or, pour résoudre ce type de problèmes, l'utilisation d'une discrétisation déterministe de $[0,1]^d$ peut également être utilisée, et apparaît donc comme une alternative aux méthodes de Monte-Carlo. Plus généralement, lorsque les méthodes de Monte-Carlo sont utilisées afin de traiter des problèmes d'analyse numérique (calcul d'intégrales, résolution d'e.d.p., résolution de systèmes linéaires) pour lesquelles des méthodes déterministes sont disponibles, la question de savoir si celles-ci sont compétitives se pose.

Les meilleures discrétisations connues à l'heure actuelle donnent typiquement des erreurs d'approximation dans le pire cas en $O(N^{-1}\log(N)^{d-1})$, où N désigne le nombre de points de discrétisation utilisés. Par comparaison, en vertu du théorème de la limite centrale, l'utilisation de N simulations indépendantes (chaque simulation jouant ici le rôle d'un point de discrétisation) conduit à une erreur en $O(N^{-1/2})$, qui a la propriété remarquable de ne pas dépendre de la dimension d du problème considéré. A priori, lorsque N tend vers l'infini, les méthodes déterministes donnent lieu à une erreur beaucoup plus faible. Cependant, pour des valeurs raisonnables de N (disons, de l'ordre de 10^6) et des valeurs importantes de d (disons de l'ordre de quelques dizaines), le terme en $\log(N)^d$ domine le comportement de l'erreur d'approximation déterministe, et celle-ci est incomparablement plus élevée que celle de la méthode de Monte-Carlo. On retiendra globalement que les méthodes de Monte-Carlo sont en général compétitives, voire les seules utilisables, lorsque la dimension du problème considéré est grande (la relative lenteur de convergence, en $N^{-1/2}$, de la méthode de Monte-Carlo ne permet cependant pas, en général, d'obtenir des estimations extrêmement précises). Un autre intérêt des méthodes de Monte-Carlo est leur relative insensibilité à l'irrégularité des fonctions intervenant dans le problème.

Cette discussion n'est toutefois que globale, et, dans diverses situations, la dimension effective d'un problème peut se révéler bien inférieure à sa dimension apparente, et des méthodes déterministes se révéler plus efficaces. Des écritures du modèle à partir de variables aléatoires de loi autre qu'uniforme (par exemple gaussiennes, ou Bernoulli) sont parfois possibles, et peuvent conduire à d'autres types de discrétisations et d'approximations.

7 Bibliographie

[ES] M. Evans, T. Swartz. Approximating Integrals via Monte-Carlo and Deterministic Methods. Oxford University Press, 2000.

[F] G. Fishman. Monte-Carlo concepts algorithms and applications. Springer-Verlag, 1996.

[LPS] B. Lapeyre, E. Pardoux, R. Sentis. Méthodes de Monte-Carlo pour les équations de transport et de diffusion. Springer-Verlag, 1997.

[LK] A. Law, W. Kelton. Simulation Modeling and Analysis. 3ème édition, Mc Graw Hill 2000.

[M] N. Madras, Lectures on Monte Carlo Methods, (Fields Institute Monographs Series), American Mathematical Society, Providence, 2002.

[McD] C. McDiarmid. Concentration. In Probabilistic Methods for Algorithmic Discrete Mathematics. Springer-Verlag, 1998.

[S] A. Shiryaev. Probability. Springer-Verlag, 1995.

[Y] B. Ycart, Modèles et algorithmes markoviens. Springer-Verlag, 2002.