

ALASCA — Architecture logicielles avancées pour les systèmes cyber-physiques autonomiques

© Jacques Malenfant

Master informatique, spécialité STL – UFR 919 Ingénierie

Sorbonne Université
Jacques.Malenfant@lip6.fr

Cours 5

Simulation des systèmes cyber-physiques et autonomiques

Objectifs pédagogiques du cours 5

- Comprendre la simulation comme outil permettant de modéliser puis d'**exécuter virtuellement** un système pour mieux le comprendre, le mettre au point et l'optimiser.
- Distinguer et comprendre les trois grandes familles de modèles de simulation : la **simulation par événements discrets**, la **simulation continue** et la **simulation hybride**.
- Comprendre les principaux **algorithmes d'exécution** des simulateurs selon leur famille de modèle.
- Comprendre les relations entre les familles de modèles et leur capacité à se modéliser les uns les autres ainsi que leur capacité à être intégrés dans la simulation hybride.

Plan

- 1 Simulation des systèmes
- 2 Simulation par événements discrets
- 3 Simulation continue
- 4 Simulation hybride

- Comme d'autres formes de modélisation, les modèles de simulation cherchent à représenter le comportement d'un système en capturant ses caractéristiques essentielles, *mais il s'agit alors de **modèles exécutables***.
- Intrinsèquement, simuler vise à *imiter* le comportement du système, ce qui permet de l'*exécuter virtuellement*.
- Dit autrement, la simulation s'intéresse à reproduire synthétiquement le comportement des systèmes pour en capturer les propriétés essentielles qui pourront ensuite être analysées à moindre coût qu'en expérimentant avec le système réel.
- Cependant, si le degré de réalisme d'un modèle de simulation est certes garant de la *fidélité des résultats à la réalité*, c'est également un facteur d'augmentation de la *complexité* et des *temps de calcul* des exécutions nécessitant un *compromis*.

- Confrontée à des équations différentielles sans solution analytique, la simulation peut être vue comme le calcul de la trajectoire du système par l'intégration numérique de leurs équations : c'est la *simulation continue*.
- En parallèle, confrontée de systèmes discrets, la simulation peut aussi être vue comme le calcul des modifications de l'état du système uniquement aux occurrences d'événements : c'est la *simulation par événements discrets*.
 - La nature *stochastique* de plusieurs phénomènes (arrivées de clients, durée de tâches, etc.) a été intégrée beaucoup plus tôt dans la simulation par événements discrets que dans la simulation continue.
- La nécessité de simuler des systèmes combinants évolutions discrètes et continues a justifié une étude conjointe des deux approches précédentes pour donner la *simulation hybride*.
 - Interprétation continue de l'évolution discrète : fonction continue *en escalier*.
 - La principale difficulté de cette combinaison est la détection *précise* des événements déclenchés par le *franchissement de seuils* par les variables continues.

- *L'espace d'états du système* : modélisé par des variables dont les valeurs ponctuelles donnent l'état courant du système.
- *Les entités participant à la simulation* : leurs *attributs* font partie de l'espace d'états.
- *Le temps simulé* : dont l'*horloge de simulation* marque l'avancement du temps au fil de l'exécution virtuelle du système.
- *Les types d'événements* : dont les occurrences possèdent des attributs et en particulier des *instants d'occurrence* en temps simulé représentant des moments de sauts discrets ponctuels dans l'espace d'états.
- *Les trajectoires continues* : évolutions de variables d'état continues en fonction du temps simulé pouvant subir des discontinuités provoquées par des événements discrets.
- *La trajectoire globale* : composée de trajectoires continues et de sauts discrets des valeurs de variables d'état.
- *Les sorties du système* : associées aux états ou aux transitions.

- Exécuter une simulation calcule pas-à-pas, à chaque exécution, une *trajectoire globale* sur le temps simulé *i.e.*, un *exemple* du comportement en fonction :
 - de ses données d'entrée et de son état initial,
 - du comportement exprimé par le modèle de simulation,
 - et dans le cas stochastique, des réalisations particulières des probabilités d'occurrence d'événements ou des trajectoires continues aléatoires (parce qu'on peut inférer toute la trajectoire à partir de ces points).
- Mais pour comprendre un système, il faut étudier l'ensemble ou un *nombre représentatif* de ses trajectoires globales possibles.
- Une *campagne de simulation* vise donc à *échantillonner* les trajectoires globales possibles pour un système et ce en utilisant un échantillon représentatif de données et états initiaux puis en réalisant de nombreuses simulations à partir de ceux-ci pour en analyser les résultats globalement.
- Plus un modèle de simulation est complexe, plus son exécution exige de calculs, plus le coût et la durée des campagnes seront élevés : une autre raison d'accepter un *compromis* entre fidélité et temps de calcul.

- Donc, trois principales formes de simulation :
 - 1 Simulation par événements discrets
 - 2 Simulation continue
 - 3 Simulation hybride
- Mais aussi trois grandes modalités :
 - 1 simulation déterministe versus stochastique,
 - 2 simulation centralisée versus concurrente/répartie,
 - 3 simulation en temps simulé, versus en temps réel, versus en temps réel accéléré.
- Principaux concepts de mise en œuvre :
 - horloge de simulation ;
 - modèle(s) de simulation pour calculer l'état du système aux instants nécessaires et suffisants dans le temps simulé ;
 - collecte des mesures utiles comme résultats de la simulation ;
 - les simulations stochastiques échantillonnent les phénomènes aléatoires $\Rightarrow$ campagnes pour obtenir des estimations précises.

- 1 Simulation des systèmes
- 2 Simulation par événements discrets
- 3 Simulation continue
- 4 Simulation hybride

- Les systèmes à événements discrets sont des systèmes dont l'état n'évolue significativement qu'aux instants d'occurrence d'événements se produisant en temps continu.
- Puisque leur état ne change qu'aux occurrences d'événements, on peut les modéliser
 - par des variables dont les valeurs *n'évaluent que lors des occurrences des événements*, ou alors
 - dont on peut se contenter de *calculer leurs valeurs qu'aux instants d'occurrence des événements*.

Par exemple, en simulant une file d'attente, il suffit de calculer la longueur de la file qu'aux instants où des clients arrivent et leur durée de passage qu'aux instants où leur service se termine.

- Les algorithmes de base de simulation par événements discrets utilisent cette propriété pour calculer la trajectoire du système en *passant d'un événement au suivant et avançant l'horloge par sauts, d'un instant d'occurrence au suivant*.

- Il existe deux grandes modalités (familles d'algorithmes) d'exécution des simulations par événements discrets :
 - 1 Une modalité *basique* où à chaque occurrence d'événements le modèle détermine (uniquement) le prochain événement et son instant d'occurrence puis le moteur de simulation avance l'horloge à cet instant pour l'exécuter.
 - 2 Une modalité dite *avec ordonnancement d'événements* où le modèle engendre plusieurs événements futurs ordonnancés dans une liste ordonnée par instants d'occurrence croissants et le moteur de simulation avance d'un événement au suivant en prenant le premier événement dans la liste.
- La modalité *basique* a ceci d'intéressant qu'elle permet d'exprimer toutes les formes de simulation par événements discrets bien qu'au prix d'une complexité plus grande des modèles.
- La seconde modalité a été énormément utilisée (voir le livre de Law dont la référence est donnée au dernier transparent).

Modèles de simulation avec ordonnancement d'événements

Parmi les modèles à ordonnancement, il y a aussi deux approches duales :

- Des modèles *axés événements exécutables* explicites :

idée : modéliser les événements qui sont exécutés pour modifier l'état du système ainsi que pour induire de nouveaux événements ;

simuler : exécuter la séquence des événements les uns après les autres.

Exemple : les guichets d'une banque sont simulés en exécutant les événements « arrivée », « début » et « fin de service » des clients.

- Des modèles *axés entités exécutables* à événements implicites :

idée : modéliser le comportement individuel de chacun des types d'entités pendant l'exécution du système ;

simuler : exécuter en parallèle toutes les entités du système.

Exemple : l'entité « client » arrive à la banque et s'insère dans la file d'attente, attend, se présente au guichet, est servi, termine et part.

Nota : très proche de l'idée fondamentale de la programmation par objets, le premier langage, Simula, ayant servi à faire de la simulation par événements.

Modèles axés événements exécutables explicites

- Un tel modèle est composé d'*entités passives* :

- qui ont des propriétés capturées par les valeurs de leurs *attributs* et
- qui vont être impliqués dans des événements changeant leur état.

La collection de tous les attributs de toutes les entités ainsi que les variables globales du modèle forment l'*état* du système.

- L'*occurrence* d'événement est le moment où le nouvel état du système est recalculé et de nouveaux événements engendrés.
- Les événements débutent ou terminent des activités du système réel qui ont une certaine durée mais se déroulent en arrière-plan de la simulation car *rien de significatif* nécessite d'être observé sur l'état pendant celle-ci.
- Le comportement d'un système est simulé en calculant la trajectoire des valeurs d'attributs des entités et des variables globales au fil d'un temps discret, *restreint* aux occurrences d'événements.
- Un modèle est *déterministe*, si tous les événements, leurs relations causales et leurs impacts sur l'état sont déterministes, ou *stochastique* si certaines de ces relations ou de ces impacts sont aléatoires.

Étapes de modélisation axée événements

- Identification des *types d'entités* du système avec l'élicitation de leurs *propriétés* et définition de leurs *attributs*.
- Identification *types d'événements* ponctuant le déroulement de l'exécution du système en partant des activités et des entités qui y sont impliquées.
- Description de la façon dont l'exécution des types d'événements change l'état du système.
- Identification des *relations causales* entre les événements (ex. : le début de service d'un client cause la future occurrence de la fin de service et la fin de service cause le début de service du client suivant dans la file s'il y en a un).
 - Ajout éventuel de certaines relations causales *synthétiques* pour faciliter la simulation (ex. : l'arrivée d'un client engendre l'arrivée du client suivant).
- Description des relations causales entre événements pour déterminer comment engendrer les nouveaux événements et leurs instants d'occurrence (avec les lois de probabilités et leurs paramètres associés).
- Programmation du modèle sur la base d'un moteur de simulation par événements discrets.

Moteur de simulation axé événements basique

- Modèle proposant une fonction donnant :
 - le délai d'attente jusqu'à l'occurrence du prochain événement ;
 - l'état résultat de l'exécution d'un événement sur l'état courant.
- Algorithme d'exécution : soit t le temps simulé,
 - Récupérer le délai d jusqu'à l'occurrence du prochain événement.
 - Avancer l'horloge de simulation à $t + d$.
 - Exécuter le prochain événement pour obtenir un nouvel état.
 - Tant que la simulation n'est pas terminée, reprendre à 1.
- Gestion du temps simulé et de l'horloge de simulation :
 - en temps synthétique = extraction des occurrences d'événements du temps réel donnant une simple séquence d'instant de calcul exécutés par saut immédiat de l'un au suivant ;
 - en temps réel : horloge simulée = temps réel, maintien de la synchronie entre instant de calcul et instant du temps réel mais mise en sommeil d'un instant au suivant ;
 - en temps réel accéléré : contraction ou expansion du temps selon une relation linéaire constante entre l'horloge et le temps réel.

- Les modèles à ordonnancement d'événements permettent de *planifier à l'avance* plusieurs événements à venir.
- File des événements à venir : ensemble des événements triés en ordre croissant de leurs instants d'occurrence $t_1, t_2, \dots$
- Algorithme d'exécution :
 - Prendre le premier événement dans la file.
 - Avancer l'horloge à l'instant d'occurrence de cet événement.
 - Calculer les modifications qu'il entraîne sur l'état du système et appliquer ces valeurs pour modifier l'état
 - Générer les événements futurs causés par l'événement courant et les insérer dans la file des événements.
 - Tant que la file des événements n'est pas vide, reprendre à 1.
- Le moteur peut utiliser les mêmes modalités de gestion du temps simulé que les modèles de base.

- Dans les simulations par événements discrets, l'aléa apparaît principalement dans les instants d'occurrence des événements.
Ex.: arrivées des clients au guichet.
 - On modélise souvent cette forme d'aléa comme une durée aléatoire entre les occurrences d'événements.
Ex.: délais entre les arrivées des clients.
 - Comment obtenir des valeurs concrètes pour ces délais ?
 - On utilise des générateurs de nombres pseudo-aléatoires¹ qui, à partir d'une valeur initiale choisie, engendrent une séquence de nombres qui suivent une certaine densité de probabilité.
 - Pour chaque événement engendré, l'instant de son occurrence est obtenu en utilisant le prochain nombre dans la séquence obtenu du générateur.
- Ex.: à l'arrivée d'un client, on engendre l'arrivée du client suivant après un délai suivant une loi exponentielle.*

¹ Presque tous les langages offrent de tels générateurs, mais ils sont rarement bons ; il vaut mieux utiliser des bibliothèques comme common-maths.

- Modèle de file d'attente M/M/1 : arrivées markoviennes, service markovien, un serveur.
- Entités : requête, serveur, file d'attente
- Attributs :
 - requête : moment d'arrivée
 - serveur : libre ou occupé
 - file : séquence des requêtes en ordre d'arrivée
- Événements :
 - arrivée d'une requête : marque la requête par l'heure de son arrivée, et si le serveur est libre, provoque immédiatement un début de service pour cette requête, sinon la requête est ajoutée à la fin de la file d'attente.
 - début de service : passe le serveur à occupé et génère l'événement de fin de service pour la requête correspondante.

- fin de service : calcule le temps de service de la requête, puis si la file n'est pas vide, provoque immédiatement le début de service de la prochaine requête, sinon passe le serveur à libre.
- Causalité entre événements :
 - Les arrivées peuvent engendrer des débuts de service, lesquels engendrent les fins de service, qui elles-mêmes peuvent engendrer un début de service, mais comment engendrer les arrivées ?
 - Solution 1 : générer à l'avance toutes les arrivées et les insérer dans la file des événements ?
⇒ Long et coûteux en espace pour les simulations longues.
 - Solution 2 : générer la première arrivée, et ajouter une relation causale synthétique faisant que le traitement d'un événement arrivée génère l'événement arrivée suivant.
⇒ Beaucoup moins coûteux en espace.
⇒ Introduction d'une causalité *synthétique*.

Animation du modèle

Horloge : 0

File des événements

Arrivée Requete1 t : 2

Requete1 arrivée :

File d'attente

--

Serveur état : libre

Somme des temps :
nombre de requetes :

Animation du modèle

Horloge : 2

File des événements

Début de service Requete1 t : 2	Arrivée Requete2 t : 3
------------------------------------	---------------------------

Requete1 arrivée : 2

Requete2 arrivée :

File d'attente

--

Serveur état : libre

Somme des temps :
nombre de requetes :

Animation du modèle

Horloge : 2

File des événements

Arrivée Requete2 t : 3	Fin de service Requete1 t : 5
---------------------------	----------------------------------

Requete1 arrivée : 2

Requete2 arrivée :

File d'attente

--

Serveur état : occupé

Somme des temps :
nombre de requetes :

Animation du modèle

Horloge : 3

File des événements

Fin de service Requete1 t : 5	Arrivée Requete3 t : 6
----------------------------------	---------------------------

Requete1 arrivée : 2

Requete2 arrivée : 3

Requete3 arrivée :

File d'attente

--

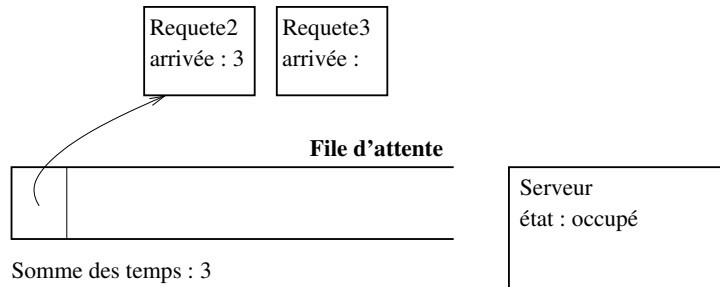
Serveur état : occupé

Somme des temps :
nombre de requetes :

Horloge : 5

File des événements

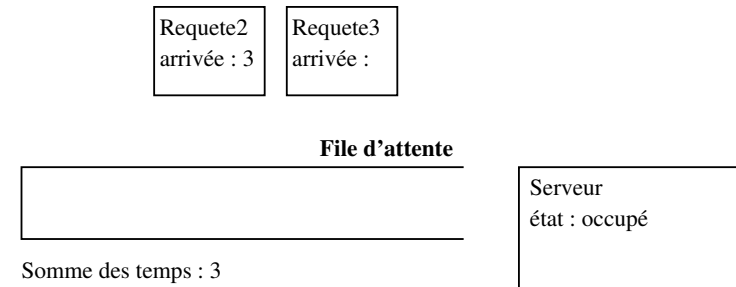
Début de service Requete2 t : 5	Arrivée Requete3 t : 6	
------------------------------------	---------------------------	--



Horloge : 5

File des événements

Arrivée Requete3 t : 6	Fin de service Requete2 t : 9	
---------------------------	----------------------------------	--



- Et ainsi de suite jusqu'au temps de fin de simulation ou une liste d'événements vide...

- Dans la simulation par événements classique, les événements sont ordonnancés puis déclenchés par leur *moment d'occurrence* dans le temps simulé.
- Une extension autorise des événements déclenchés par des *modifications de valeur des variables* selon deux modalités :
 - dès qu'une modification est observée, peu importe laquelle ;
 - dès que la ou les valeurs d'une ou plusieurs variables satisfont à une certaine condition logique.
- Notons encore qu'en simulation par événements, les valeurs des variables ne changent qu'aux instants ponctuels où les événements se produisent.
- Il suffit de vérifier les conditions au fur et à mesure où les valeurs de variables changent, lors de l'exécution des événements.
 - On ajoute dans l'algorithme d'exécution la vérification des conditions de déclenchement pour générer les événements concernés.

- Attention toutefois à « l'effet Zeno » :
 - Un événement change les valeurs de variables, ce qui déclenche immédiatement un autre événement qui lui-même change les valeurs de variables et peut en déclencher encore un autre, etc.
 - L'effet dit *Zeno* se produit quand un système cherche à exécuter un nombre potentiellement illimité d'événements à un même instant ponctuel, la simulation ne pouvant alors plus faire progresser l'horloge.
- L'effet Zeno peut se produire dès le modèle autorise un événement à causer d'autres événements *au même instant*, y compris ceux engendrés avec un délai d'occurrence égal à 0.
- Un modèle de simulation correct doit donc toujours s'assurer de ne pas pouvoir engendrer un nombre potentiellement illimité d'événements au même instant.
 - Un peu un équivalent en simulation par événements de la boucle ou de la récursivité qui ne s'arrêtent pas en programmation.

- Modèle d'interruptions du réseau :
 - deux événements, interruption et reprise, s'engendrent l'un l'autre avec des délais aléatoires (durées) entre eux.
- Modèle Tic :
 - Chaque événement tic engendre le suivant après une durée fixe, paramètre du modèle.
- Par rapport à des automates hybrides, la simulation par événements est une modalité pour les exécuter qui évite d'avoir à les simuler en temps continu.
 - Il serait beaucoup trop coûteux de les voir comme des réveils-matins avec un délai jusqu'à leur déclenchement où on simulerait l'écoulement continu du temps pour détecter quand le déclenchement doit se faire...

- 1 Simulation des systèmes
- 2 Simulation par événements discrets
- 3 Simulation continue
- 4 Simulation hybride

- (Rappel) Moteur de simulation continue : calcul numérique, et en particulier l'*intégration numérique*.
- Malgré son nom, elle part toujours d'une *discrétisation*, compte tenu de la nature discrète de tout calcul sur ordinateur.
- L'approche la plus courante consiste à *discrétiser le temps* et alors l'algorithme général de base consiste à :
 - implanter les équations différentielles en code exécutable,
 - adopter un pas d'intégration Δt (avancement du temps),
 - calculer la trajectoire des variables, de proche en proche, par leurs valeurs et leurs dérivées à t_0 , $t_0 + \Delta t$, $t_0 + 2\Delta t$, ...
- L'intégration numérique est une technique d'analyse numérique pouvant traiter n'importe quelle forme d'intégrale e.g.,

$$y = \int_{x_0}^{x_1} f(x) dx$$
- Vision simulation : l'horloge simulée progresse du temps t à $t + \Delta t$, etc. en calculant à chaque fois la nouvelle valeur de la trajectoire.

- Le calcul de la trajectoire utilise une *méthode d'intégration* choisie en fonction du type d'équations différentielles et de la précision.
 - Elles approximent plus ou moins bien la fonction calculée.
 - Choix $\Rightarrow$ *compromis* entre *précision* et *quantité de calcul*.
- Dans sa forme la plus simple, une équation différentielle $y'(t) = f(t, y)$ permet aussi de calculer numériquement la dérivée de la fonction recherchée y en tout point t de définition.
- À partir d'une condition initiale $y(t_0) = y_0$, il devient possible de calculer itérativement les valeurs successives de y_i en t_i à partir de la valeur précédente y_{i-1} et de la dérivée calculée à l'instant précédent t_{i-1} , c'est-à-dire $y'(t_{i-1}) = f(t_{i-1}, y_{i-1})$.
- Le délai entre les valeurs successives $\Delta t = t_i - t_{i-1}$ est appelé *pas d'intégration*.

Ici, on présente le pas d'intégration Δt comme constant, mais plusieurs méthodes d'intégration numérique le font varier selon l'ampleur de la dérivée pour diminuer les erreurs de calcul (*méthodes adaptatives*).

- Pour obtenir des résultats plus précis que ceux de la méthode d'Euler, d'autres méthodes utilisent une *approximation quadratique*.
- La méthode quadratique la plus populaire est aujourd'hui celle de *Runge-Kutta d'ordre 4*, appelée ainsi car elle utilise quatre valeurs de la dérivée à chaque pas d'intégration (multipliant d'autant la quantité de calcul par rapport à Euler où une suffit).
- Par ailleurs, l'erreur de calcul sur la fonction ne dépend pas que de la méthode, mais aussi de la forme de la fonction dont on doit tenir compte pour les choix de méthode et de pas d'intégration.
- On trouve des bibliothèques de code pour résoudre les équations différentielles comme « common math » d'Apache (`org.apache.commons.math3.ode`).

- La méthode d'Euler est la méthode d'intégration numérique par quantum de temps la plus ancienne et la plus simple.
- Elle utilise une approximation linéaire pour faire le calcul suivant :

$$y_i = y_{i-1} + f(t_{i-1}, y_{i-1}) \Delta t$$

- où $\Delta t = t_i - t_{i-1}$ est un quantum de temps et le *pas d'intégration*.
- Les avantages de la méthode d'Euler sont sa simplicité et la faible quantité de calcul qu'elle nécessite à chaque pas d'intégration.
- L'approximation linéaire d'Euler entraîne cependant une erreur de calcul assez importante en général.
 - On peut réduire l'erreur en diminuant le pas d'intégration.
 - Mais cette réduction augmente la quantité totale de calculs à faire, donc il y a une limite à cette réduction.
- Lorsque la réduction du pas d'intégration devient trop importante pour obtenir une certaine précision, on peut passer plutôt à une méthode d'intégration non linéaire.

- Rappel : discrétisation par *quantum de temps* :
 - On fixe un quantum de temps Δt et à chaque itération on l'utilise comme pas d'intégration en se demandant : *Quelle sera la valeur de y_{i+1} au temps t_{i+1} étant donnée sa valeur y_i à t_i où $t_{i+1} = t_i + \Delta t$.*
- Dans plusieurs cas, on peut obtenir un meilleur contrôle de l'erreur pour un coût en calcul bien inférieur en utilisant un *quantum d'état*, technique publiée sous le nom *quantized state systems* ou QSS.
 - On fixe un quantum de variation de l'état $\pm \Delta y$ et on se demande à quel moment $t_{i+1} = t_i + \Delta_i$ la valeur y_{i+1} va atteindre $y_i \pm \Delta y$ (selon le signe de la dérivée), Δ_i devenant le pas d'intégration à cette étape de calcul.
 - C'est donc une technique à pas d'intégration *variable*.
- Pour prédire Δ_i , on utilise une approximation de la fonction y , comme dans les méthodes à quantum de temps. On trouve :
 - des prédictions linéaires, méthodes *QSS1* ;
 - des prédictions non linéaires, méthodes *QSS2*, *QSS3*, ...

Quand choisir la discrétisation de l'état ?

- Observation : plus la dérivée est faible, plus il faut de temps pour atteindre une variation de Δy fixe sur la fonction y .
- Ainsi, plus les dérivées sont faibles, plus les pas d'intégration s'allongent et plus la quantité de calcul diminue pour une durée de simulation fixée.
- On choisit donc les approches QSS lorsque les valeurs de $y'(t) = f(t, y)$ demeurent relativement faibles car, alors, elles permettent d'allonger les Δ_i et de diminuer la quantité de calcul pour la simulation.
- Une approche *mixte* consisterait à passer de l'une à l'autre (QSS versus quantum de temps) en fonction de la valeur instantanée de la dérivée.
- Les défenseurs des méthodes QSS ont montré que dans beaucoup de cas, fixer un quantum d'état permet d'obtenir une meilleure précision que de fixer des quantum de temps, à *quantité de calculs constant*.

Cas continu stochastique

- Deux principaux types de phénomènes stochastiques continus :
 - Variables aléatoires continues mais à *valeurs ponctuelles* réalisées par leurs fonctions de distribution de probabilité et dont les valeurs individuelles sont produites par des générateurs de nombres pseudo-aléatoires.
 - Trajectoires aléatoires (mouvements browniens) modélisées par des équations différentielles stochastiques qui peuvent être étudiées de deux façons :
 - 1 Production de trajectoires aléatoires : intégration de l'équation différentielle stochastique (par exemple, intégrale d'Itô numérique) produisant *graduellement* une trajectoire.
 - 2 Échantillonnage aléatoire de trajectoires déterministes : le côté stochastique est réalisé par un choix aléatoire entre différentes trajectoires déterministes qui sont donc *complètement connues* dès le début de la simulation.

Intégration numérique d'ED stochastique

- À la base d'une équation différentielle stochastique, l'intégrant suit une densité de probabilité connue.
Ex.: modèle de la bande passante WiFi avec

$$\dot{p}(t) = \sigma(p(t))d\mathcal{P}(t)$$

- Pour réaliser l'intégration numérique, il faut (encore ici) discrétiser cette ED stochastique pour passer en quelque sorte à une *équation aux différences*, c'est-à-dire que pour discrétiser $d\mathcal{P}(t)$ on va utiliser :

$$\Delta\mathcal{P}_i = \mathcal{P}(t_i) - \mathcal{P}(t_{i-1})$$

- Si on pose pour l'instant $\sigma(p(t)) = 1$ et $p(t_0) = P_0$, alors le calcul des valeurs discrétisées de $p(t)$, $t = 0, \dots$ devient pour chaque t_i :

$$p(t_i) = P_0 + \sum_{j=0}^i \Delta\mathcal{P}_j$$

Comment calculer une trajectoire ?

- Pour calculer la trajectoire, il faut calculer des différences $\Delta\mathcal{P}_i$ i.e., engendrer des valeurs aléatoires suivant une densité de probabilité donnée par le modèle avec des générateurs de nombres pseudo-aléatoires.
- Ex.: pour l'exemple du réseau WiFi, nous avons supposé que la valeur absolue de l'intégrant suivait une loi exponentielle. En discrétisant, il est logique de supposer que les différences successives croissent en fonction du délai $t_i - t_{i-1}$. Supposons donc que :

$$\Delta\mathcal{P}_i = \mathcal{P}(t_i) - \mathcal{P}(t_{i-1}) \sim \mathbf{Exp}[(t_i - t_{i-1}) * s]$$

où s est un facteur d'échelle contrôlant l'ampleur de variations de la bande passante par unité de temps.

- Une trajectoire sera alors obtenue grâce à l'équation du transparent précédent en y utilisant les nombres pseudo-aléatoires successifs générés selon les lois $\mathbf{Exp}[(t_i - t_{i-1}) * s]$.

Plan

- 1 Simulation des systèmes
- 2 Simulation par évènements discrets
- 3 Simulation continue
- 4 **Simulation hybride**

Vers la simulation des systèmes hybrides

- Principale technique : intégration pas-à-pas pour obtenir une trajectoire du système continue par morceaux interrompue pour traiter les deux types d'événements discrets :
 - ➊ ceux générés par leurs processus propres et traités comme dans la simulation par événements discrets mais en interrompant les calculs de la partie continue,
 - ➋ ceux générés par l'évolution continue par le franchissement de conditions aux frontières (comme ceux déclenchés par le modèle dans la simulation par événements discrets déjà vus, mais cette fois sur des variables continues).
- Principale difficulté : déterminer précisément le moment où les événements déclenchés par l'évolution continue se produisent.
 - Si on admet une tolérance ε , elle peut se répercuter par une modélisation trop infidèle du système.
 - Exemple : simulation hybride de la trajectoire d'une balle qui tombe sur une table et rebondit ; une tolérance ε pour détecter la collision implique d'admettre que la balle pénètre d'une profondeur ε dans la table...

Détermination des transitions en discrétisation du temps

- Avec un pas d'intégration Δt , le simulateur discrétisant le temps saute en réalité de t_i en $t_i + \Delta t$.
- Ceci produit une série d'instants qui n'ont pas de raison de correspondre aux instants d'occurrence des événements générés par l'évolution continue du système.
- Aucun pas d'intégration Δt ne peut garantir une précision ε donnée sur l'évolution des variables du modèle car les dérivées sont non-bornées.
 ⇒ *Il faut rechercher l'instant d'occurrence plus précisément !*
- Se fait généralement en deux temps :
 - détection du franchissement de frontière entre Δt_i et Δt_{i+1} puis
 - recherche (dichotomique, ...) du point de franchissement précis à ε près dans l'intervalle $]t_i, t_i + \Delta[$.

Détermination des transitions en discrétisation de l'état

- Constat : en discrétisation du temps, la recherche du moment d'occurrence de l'événement revient à se demander au temps t_i dans combien de temps le franchissement va se produire.
 $\Rightarrow$ *C'est exactement ce que fait la discrétisation de l'état !*
- Comment faire en discrétisation de l'état ?
 - Vérifier si le point de franchissement $\bar{y}$ se situe dans l'intervalle $y_i \pm \Delta y$.
 - Si c'est le cas, fixer temporairement le quantum $\Delta y' = |y - \bar{y}|$.
 - Déterminer dans combien de temps $\bar{\Delta} t$ le système atteindra $\bar{y}$.
 - Avancer le système à $t_i + \bar{\Delta} t$ en fixant la nouvelle valeur de y précisément à $\bar{y}$.
- Il demeure possible de faire une erreur sur le délai $\bar{\Delta} t$ mais cela est souvent moins dommageable qu'une erreur sur l'état (comme voir la balle pénétrer dans la table).

Intégration avec la simulation par événements discrets

- Les événements discrets déclenchés par le temps se produisent par définition à des instants prévisibles :
 - Pour les événements déterministes, leur instant d'occurrence t peut être pré-calculé et l'événement est planifié à cet instant.
 - Pour les événements stochastiques, la réalisation par un simulateur suppose de générer des valeurs de variables aléatoires pour leur instant d'occurrence et à partir de là, ces instants sont équivalents à des événements déterministes.
- Une fois l'instant d'occurrence d'un événement prévu, le simulateur gère sa liste des événements futurs à chaque pas d'intégration pour intercaler ces événements.
- Quand le prochain événement doit se produire avant la fin du pas courant ou avant l'atteinte du prochain quantum d'état, il faut
 - avancer à l'instant d'occurrence du prochain événement,
 - recalculer éventuellement les valeurs des variables continues,
 - exécuter l'événement puis
 - reprendre l'intégration à partir de cet instant.

Récapitulons...

- 1 Programmer une simulation n'est pas si difficile, mais plus difficile de *bien modéliser*. Ne pas abuser des simplifications, en particulier lorsque des activités ont des interactions restant implicites dans le modèle.
Ex. : modélisation de la communication par réseau par des événements émission et réception, sans égard aux autres communications et à la congestion éventuelle du réseau.
- 2 Dans le cas des modèles stochastiques, la disponibilité ou l'obtention des données réelles pour estimer des lois de probabilités justes est souvent l'une des plus grandes difficultés. La génération de valeurs aléatoires est aussi cruciale : évitez les mauvais générateurs des langages de programmation et préférez ceux des bibliothèques spécialisées (comme common-math).
- 3 Une seule exécution d'une simulation stochastique est insuffisante : il faut en faire beaucoup, selon un **plan d'expérimentation** bien défini et analyser scrupuleusement les résultats. L'aide d'un statisticien peut être utile.
- 4 L'intégration numérique selon l'approche de la **discrétisation de l'état** (*quantized state systems*) paraît la plus prometteuse pour produire des moteurs de **simulation hybride** précis et efficaces.
- 5 Parmi les techniques de simulation utilisables pour tester et valider les entités autonomiques, on trouve principalement la **simulation hybride modulaire répartie en temps réel** comme une extension des modèles DEVS de simulation par événements discrets modulaires.

Pour aller plus loin : sélection de lectures recommandées

- Simulation Modeling & Analysis*, A.M. Law, 4è éd., McGraw Hill, 2007.
- Numerical Analysis*, Richard Burden, Douglas Faires et Albert Reynolds, 2è édition, Prindle, Weber et Schmidt, 1981 (ou tout autre bon livre d'analyse numérique).
- An algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations*, Desmond Higham, SIAM Reviews, 43, 3, pp. 525-546, 2001.