DEFINITION D'UNE METHODOLOGIE DE CONCEPTION DES SYSTEMES MECATRONIQUES SURS DE FONCTIONNEMENT

Raphaël Schoenig

Institut **N**ational **P**olytechnique de **L**orraine
Centre de **R**echerche en **A**utomatique de **N**ancy

THESE

Présentée à

l'Institut National Polytechnique de Lorraine

en vue de l'obtention du titre de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Discipline : Automatique

par

Raphaël SCHOENIG

Le 26 octobre 2004

DEFINITION D'UNE METHODOLOGIE DE CONCEPTION DES SYSTEMES MECATRONIQUES SURS DE FONCTIONNEMENT

Membres du jury :

M. Jean-François AUBRY	Directeur de thèse
M. Yves DUTUIT	Rapporteur
M. Daniel NOYES	Rapporteur
M. Tony HUTINET	Examinateur
M. Pierre-Etienne LABEAU	Examinateur
M. Didier MAQUIN	Examinateur
M. Thierry CAMBOIS	Invité

A mes parents,

Avant-propos

Le travail présenté dans ce mémoire a été effectué dans le cadre d'une convention CIFRE entre le Centre de Recherche en Automatique de Nancy (CRAN) et GFI Consulting en partenariat avec PSA Peugeot Citroën.

Les travaux ont été dirigés par le professeur Jean-François AUBRY, à qui je tiens à exprimer ma plus profonde gratitude pour son soutien, ses conseils et pour sa disponibilité lors de mes passages au CRAN. Merci encore pour tout l'intérêt que vous avez porté sur mes travaux.

Au sein de PSA, mon encadrement a été assuré les deux premières années par Edwige GUILHEM dans le service Electronique et Informatique Embarqués Véhicule de la Direction des Systèmes d'Informations (DSIN). Merci Edwige pour tes remarques pertinentes et pour ta disponibilité.
Un grand merci à Thierry CAMBOIS, d'avoir repris l'encadrement de ma thèse pour la dernière année dans le service SMSD (Systèmes Mécatroniques Sûreté de fonctionnement Diagnostic et électronique). Je te suis reconnaissant pour l'intérêt que tu as porté sur mes travaux, pour toutes les discussions enrichissantes que nous avons eues ; tes connaissances techniques et ta connaissance du monde automobile ont permis d'accroître sensiblement la portée de mes travaux.

Je ne saurai oublier Monsieur Dominique CHARNY à qui j'exprime ma profonde reconnaissance, grâce à qui cette thèse a été possible.

Un grand merci à Tony HUTINET, responsable du pôle méthodes et outils à GFI Consulting, pour sa précieuse aide « informatique ». Merci Tony pour toute la confiance que tu m'as témoignée pendant ces trois années et pour m'avoir permis de rencontrer de nombreux interlocuteurs dans divers secteurs industriels.

Que tous les collègues que j'ai pu côtoyer durant ces années, reçoivent ici le témoignage de mon amitié. La bonne ambiance n'a rendu ces trois années que plus agréable.

Je tiens à remercier tous ceux qui, de près ou de loin, ont participé à la réalisation finale de ce manuscrit.

Enfin, je dédie ce paragraphe à tous ceux qui m'ont aidé et supporté dans cette épreuve. Je pense à mes amis et en particulier à ma famille grâce à qui j'ai pu en arriver jusque là.

Table des matières

Préambule

L'amélioration de la qualité et des performances a toujours été une préoccupation constante chez PSA Peugeot Citroën, qui propose à ses clients des véhicules toujours plus sûrs et bénéficiant des dernières innovations technologiques. Tous ces progrès ont été possibles grâce à l'introduction progressive de l'électronique et de l'informatique embarquées qui interviennent sur tous les fronts : en matière de *sécurité active*, de *sécurité passive*, d'*environnement*, de *performances* globales au niveau véhicule et, plus récemment, dans les domaines de la *télématique* et du *multimédia*. La voiture ne se limite plus uniquement à son rôle fonctionnel, mais contribue également au bien-être du conducteur et des passagers, et au plaisir de la conduite.

La part grandissante de l'électronique et de l'informatique n'est pas sans conséquences sur les méthodes de conception. La complexité croissante des systèmes embarqués nécessite d'adapter les méthodes et les outils existants par rapport aux *spécificités* de ces systèmes et au respect d'exigences toujours plus fortes en matière de *sûreté de fonctionnement*.
C'est dans ce contexte que de nombreux travaux de recherche ont été amorcés il y a plusieurs années à PSA Peugeot Citroën, en collaboration avec des laboratoires universitaires.

Le travail présenté dans ce manuscrit a pour objectif de définir une méthodologie de conception des systèmes *mécatroniques* intégrant dès les premières phases du cycle de développement, les aspects sûreté de fonctionnement. L'apport d'une telle méthodologie doit permettre de faire face à un certain nombre de contraintes propres au domaine automobile : la pression concurrentielle, les contraintes de temps, les exigences du client, le respect des normes législatives en vigueur, la coopération avec des équipementiers sont autant d'éléments justifiant la nécessité de définir un cadre méthodologique bien stabilisé.

Cependant, il ne s'agit pas de rajouter des méthodes et des outils à un processus de conception déjà établi sans tenir compte des aspects organisationnels et des habitudes des concepteurs. L'objectif de cette thèse consiste précisément à identifier les éléments méthodologiques et les outils permettant d'améliorer le processus de développement existant.

Nous cherchons tout d'abord à définir un formalisme de modélisation fonctionnelle et comportementale, support de la méthodologie, et en cohérence avec les spécificités des

systèmes mécatroniques. En particulier, les aspects *hybrides* et *temps réel* doivent pouvoir être représentés.

Ensuite, sur le plan de la vérification et de la validation, les analyses doivent garantir que le système est exempt d'erreur de conception et qu'il est conforme à ses spécifications. Outre les techniques habituellement utilisées, basées sur la simulation et les tests, nous préconisons l'utilisation de *méthodes formelles*, telles que le *model-checking*, en complément de ces techniques. Le bénéfice apporté par ces méthodes est déjà confirmé dans de nombreux secteurs industriels tels que l'aéronautique et le ferroviaire qui les exploitent dès les premières phases de la conception.

Enfin nous attachons une importance centrale à la sûreté de fonctionnement. A ce titre, une place considérable est accordée à la définition d'une méthode de représentation et d'analyse « dysfonctionnelles ». Il paraissait primordial de disposer d'un formalisme de description graphique, aussi commode que les arbres de défaillance, afin de faciliter la communication entre les divers protagonistes d'un projet et assurant une capitalisation de l'expérience sur les analyses effectuées. Cependant, les limites des méthodes qualifiées de « statiques » ne permettent pas d'évaluer la *fiabilité* des *systèmes dynamiques hybrides*. Nous nous sommes donc intéressés à l'approche *makovienne* que nous avons adaptée afin de contourner ses propres limites.

Le mémoire s'organise de la façon suivante :

Dans le chapitre introductif, nous faisons un tour d'horizon sur les systèmes mécatroniques du monde automobile et sur les différentes problématiques liées à la conception de ces systèmes. Un rappel de quelques définitions en sûreté de fonctionnement permettra ensuite d'éclairer le lecteur sur les principales notions utilisées et, en particulier, sur les problèmes liés à l'évaluation de la *fiabilité dynamique*. Nous illustrons également la nécessité de disposer d'une méthodologie de conception adaptée à ces systèmes et intégrant de façon rigoureuse et continue des aspects sûreté de fonctionnement. Nous montrons également le bénéfice apporté par les approches formelles dans les phases de vérification et de validation.

Le chapitre 2 est consacré à la description des systèmes de contrôle-commande et à l'identification de formalisme de représentation en adéquation avec leurs spécificités. Nous décrivons les principaux concepts de la technique de model-checking, et nous montrons comment celle-ci peut s'intégrer dans un cycle de conception. Nous accordons enfin une place importante à l'analyse markovienne.

Le chapitre 3 constitue la partie centrale de ce mémoire. Nous y présentons une méthode innovante permettant d'une part, de représenter graphiquement les combinaisons de défaillance amenant un système dynamique hybride dans les états redoutés identifiés et, d'autre part, de calculer les attributs de la sûreté de fonctionnement. Cette méthode est basée sur une technique d'agrégation des graphes de Markov. Les limites de l'approche markovienne et des autres méthodes d'évaluation de la sûreté de fonctionnement, ainsi que le bénéfice apporté par les outils graphiques, nous ont conduits à développer cette méthode. Les principes et les outils théoriques servant à la construction de tels graphes sont également précisés.

Enfin, le chapitre 4 illustrera l'efficacité de cette méthode à travers un cas d'application, en termes de représentation graphique et de performances. Elle sera notamment comparée à la

simulation de Monte-Carlo, et nous indiquons comment un graphe agrégé peut être construit à partir d'une représentation comportementale quelconque du système.

Nous finirons ce mémoire par une conclusion générale qui rappellera les principaux points développés dans ce travail, ainsi que notre contribution au regard des attentes des industriels dans l'amélioration de la conception des systèmes mécatroniques.

CHAPITRE 1

INTRODUCTION

Chapitre 1

Introduction

1. L'électronique dans les systèmes embarqués

1.1. Contexte

A propos de la direction assistée, on entend souvent dire : « il est difficile de s'en passer après y avoir goûté ». Difficile en effet de renoncer au confort qu'elle apporte, en particulier en ville ou lors de manœuvres délicates. A tel point que la plupart des constructeurs proposent aujourd'hui de petites voitures équipées de ce système. Après l'assistance hydraulique, la direction assistée électrique s'est imposée sur le marché pour des raisons de coûts de production, de simplicité, d'économie en énergie ou encore de la finesse du réglage de l'assistance. Cet exemple parmi tant d'autres est révélateur de la tendance technologique de ces dernières années : de plus en plus de fonctions réalisées mécaniquement ou hydrauliquement se dématérialisent au profit de l'électronique et du logiciel embarqué. De ces innovations est né le terme de « **mécatronique** » pour désigner la coopération intime de la mécanique, de l'hydraulique, de l'électronique et du logiciel dans un système. Ce concept est applicable à des fonctions aussi diverses que les commandes de portes, de l'éclairage ou encore aux fonctions dynamiques du véhicule.

Une part grandissante de l'électronique

Les premières fonctions de l'électronique se limitaient à l'amélioration du confort et des performances grâce à l'utilisation d'éléments semi-conducteurs (poste radio, allumage, injection…). L'avènement des microprocesseurs au début des années 80 a marqué une nouvelle révolution dans l'électronique embarquée par la possibilité d'intégrer du logiciel. Les premiers processeurs de traitement du signal étaient dotés d'instructions adaptées au calcul intensif en temps réel. Les équipements électroniques allaient devenir de plus en plus performants et miniaturisés, offrant la possibilité de réaliser des fonctionnalités programmables toujours plus complexes. On voit ainsi apparaître le système d'antiblocage des roues, la boîte de vitesse automatique…
Le mouvement n'est pas prêt à s'essouffler. L'électronique est devenue indispensable dans tous les domaines : le confort, la sécurité, la consommation d'énergie, la communication et la lutte contre la pollution. Il est reconnu qu'à 90%, les nouveautés apportées dans les nouveaux véhicules ne pourraient l'être sans l'électronique. Sa part représente à peu près 30% de la valeur d'un véhicule neuf moyen et pourra atteindre 40% en 2010. C'est PSA Peugeot Citroën qui avait le premier frappé les esprits lors du lancement de la 607, en soulignant (dans sa

publicité) que sa puissance de calcul est équivalente à celle d'un Airbus des années 80 ! [ELE01]

Des avantages nombreux

Cette puissance de calcul alliée à l'intégration de logiciel de plus en plus complexe a permis d'améliorer les performances et d'affiner les lois de contrôle-commande. L'électronique est ainsi devenue indispensable en matière d'environnement. Il est possible d'intervenir de façon extrêmement fine sur la gestion du moteur. Par exemple, le moteur HDI de la 307, équipé d'injecteurs à commande piézo-électriques, est capable de gérer plusieurs modes d'injection du carburant grâce à son dosage très précis et rapide. L'intégration d'un système de diagnostic embarqué, l'OBD (On Board Diagnostic), est encore un pas supplémentaire dans la lutte contre la pollution puisqu'il permet de contrôler les dépassements des seuils de rejets nocifs dans l'atmosphère.
En matière de sécurité passive et active, des progrès notables ont été réalisés dans l'automobile. Les systèmes relatifs à la sécurité active ont pour rôle d'agir sur la dynamique du véhicule afin d'augmenter ses performances et sa tenue de route. Les systèmes d'assistance à la conduite sont conçus pour permettre de corriger des erreurs de conduite mineures et d'empêcher qu'elles mènent à une situation génératrice d'accidents. On peut citer comme exemple l'ESP (programme électronique de stabilité) et l'ASR (système d'anti-patinage). En matière de freinage, outre l'ABS qui équipe la quasi-intégralité des gammes de véhicule à PSA, des systèmes plus récents ont vu le jour : le répartiteur électronique de freinage et l'assistance de freinage d'urgence (AFU). Ces systèmes équipent les modèles Citroën C3 et C5, ou encore Peugeot 307. Les trains roulants bénéficient également des avancées de l'électronique. Le système AMVAR (pour AMortissement VARiable) permet de piloter électriquement la suspension en générant plusieurs lois d'amortissement en fonction de la conduite. Certaines versions de la 607 sont dotées de ce système.
Concernant les systèmes liés à la sécurité passive, ils ont pour objet de diminuer les conséquences suite à un accident. Dans ce domaine, la gestion des ceintures de sécurité s'est elle aussi affinée : des capteurs de poids et de tension au niveau de la boucle permettent d'évaluer la morphologie du passager et adapter la réponse de la ceinture de sécurité en conséquence (ainsi que de l'airbag).
Enfin, la voiture devient communicante grâce aux possibilités de la télématique automobile. Le marché est évalué à plusieurs dizaines de milliards d'euros dans les dix ans à venir. Des systèmes embarqués permettent d'échanger de nombreuses informations avec l'extérieur. Outre le téléphone sans fil ou les systèmes d'aide à la navigation, la télématique crée de nouvelles applications telles que des services d'assistance gérés par centre d'appel (existant déjà aux Etats-Unis) en passant par des installations multimédias dédiées au divertissement des passagers. PSA commercialise sur la plupart des véhicules depuis quelques mois la fonctionnalité Appel d'Urgence destiné à mettre le conducteur en relation avec un plateau d'assistance spécialisé en cas d'accident. Un signal est émis manuellement ou automatiquement par liaison GPS à un opérateur (identification du véhicule, localisation géographique précise, …). Ce système a pour vocation d'améliorer et accélérer l'intervention des secours.

Une coopération subtile

Il y a encore quelques années, chaque fonction disposait de son propre boîtier. Ainsi, des systèmes tels que l'ABS étaient composés d'ensembles séparés calculateur/ capteurs/actionneurs. Mais la multiplication des boîtiers a rapidement engendré des problèmes

d'encombrement, de poids et de coût. Plusieurs solutions ont alors vu le jour. La première consistait à regrouper plusieurs fonctions au sein d'un même boîtier central, connu à PSA sous le nom de BSI (Boîte de Servitude Intelligente). La seconde solution consiste à partager les ressources, véritable révolution de la conception des systèmes mécatroniques. Ainsi, les informations d'un unique capteur de vitesse sont exploitées simultanément par plusieurs fonctions. Les avantages de cette approche sont indéniables : une fois que l'architecture matérielle est figée (calculateurs, capteurs), il suffit de rajouter des parties de logiciels pour offrir des fonctions supplémentaires. Par exemple, le capteur de pluie qui permet de commander la mise en route de l'essuie-glace pourrait tout aussi bien assurer la fermeture du toit ouvrant.

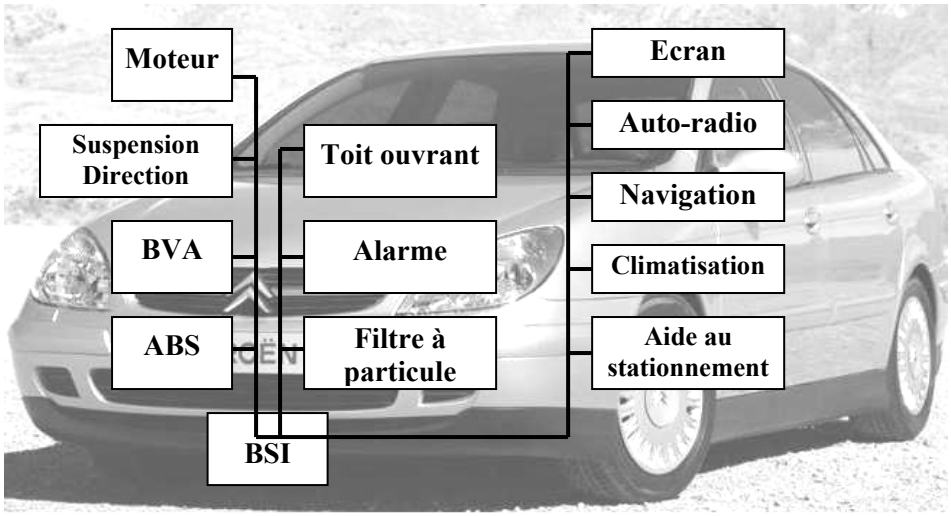


Figure 1 : Gestion des différentes fonctionnalités par multiplexage de la C5

Le domaine de la mécatronique entraîne ainsi une réduction globale du nombre de composants et des interconnections et permet un assemblage simplifié, la programmation de nouvelles fonctions ou une mise à jour logicielle des fonctions existantes. Ces avantages ont été acquis grâce au couplage de la mécatronique et d'une architecture distribuée ou centralisée par le biais des systèmes multiplexés (bus CAN, TTP,…) utilisés pour le contrôle des fonctions.

1.2. De nouvelles problématiques voient le jour

Ainsi, les systèmes mécatroniques peuvent se définir comme des systèmes complexes faisant intervenir des technologies différentes. Un système de contrôle-commande, composé d'une partie matérielle et logicielle, est en interaction constante avec un système physique (mécanique, hydraulique, électrique…). Ces deux parties communiquent par l'intermédiaire de capteurs et d'actionneurs.
L'augmentation de la complexité de ces systèmes n'est pas sans conséquences sur la *sûreté de fonctionnement*. En premier lieu, les nouvelles technologies électroniques intégrées massivement ne bénéficient pas encore d'un *retour d'expérience* sur les causes de dysfonctionnement des composants, à l'inverse des solutions mécaniques et hydrauliques. Ceux-ci doivent fonctionner dans un environnement hostile en termes de température, de

vibrations, de champs électro-magnétiques et sont fortement sollicités. Des études[(1)] ont montré que les pannes dues à des défauts électriques/électroniques ont augmenté de 23% entre 1998 et 2001, tous constructeurs confondus. Parmi ces pannes, celles liées à un problème électronique représentaient plus de 45%. Si l'on considère le rythme actuel d'augmentation de la part de l'électronique embarquée, ce chiffre ne fera que croître.

La technologie la plus problématique du point de vue de la sûreté de fonctionnement est certainement le logiciel. Outre les avantages procurés par l'enrichissement des fonctionnalités, le développement pose de sérieux problèmes. En effet, de par la taille et la complexité croissante du code embarqué, celui-ci est susceptible de contenir des défauts résultants d'erreurs *humaines* de conception. Ces erreurs sont alors révélées durant la vie opérationnelle du système et peuvent avoir des conséquences plus ou moins importantes selon le niveau de criticité de la fonction impliquée. Un dysfonctionnement de la composante logicielle peut avoir des origines différentes :

- un défaut logiciel (ou bogue) : il s'agit d'une erreur de programmation durant la phase de conception (faute de sémantique, typage des données, boucles infinies…) [VIL88].
- une spécification incomplète ou erronée : le comportement du logiciel n'est pas conforme aux exigences formulées dans le cahier des charges ou n'a pas été prévu (par exemple si certains paramètres d'entrée ou de commandes conducteurs prennent des valeurs différentes de celles prévues).
- inadéquation du matériel support vis-à-vis du logiciel : sa nature temps réel pousse les concepteurs à une utilisation proche des limites maximales, dans le but d'obtenir le maximum de performances à un moindre coût. Cela n'est pas sans conséquences sur la fiabilité ou la sécurité, si les composants impliqués dans la fonction ne sont pas correctement dimensionnés.

Le processus de développement d'un logiciel embarqué est représenté par un *cycle en « V »* normalisé (AFNOR). Celui-ci décrit les phases de conception allant du général au particulier par une démarche de raffinements successifs ainsi que les phases d'intégration et de validation associées à chaque phase de la conception.
L'expérience montre que des modifications tardives dans le cycle de développement d'un logiciel peuvent coûter très cher : entre 2 et 100 fois plus cher que ce qu'elles auraient coûté avant la phase d'industrialisation. De plus, une remise en question de l'architecture fonctionnelle est d'autant plus préjudiciable pour la sûreté de fonctionnement du système global que celle-ci est effectuée tardivement dans le cycle.

Bien qu'il soit utopique de prétendre pouvoir éradiquer complètement tout dysfonctionnement d'un système (matériel et logiciel), la « science » de la sûreté de fonctionnement a pour objectif d'aider le concepteur à comprendre les mécanismes des pannes, évaluer leurs fréquences d'apparition le plus tôt possible et leurs conséquences sur le système (*fiabilité prévisionnelle*) et, enfin, déterminer les actions pour réduire les risques. Durant ces dernières années, les fiabilistes ont développé un arsenal de méthodes et d'outils pour les aider dans leur quête. La maîtrise de la sûreté de fonctionnement consiste donc à faire descendre le risque en dessous d'un niveau acceptable [SIG96].
La démarche la plus naturelle et la plus utilisée dans l'industrie est basée sur le retour d'expérience (REX). Afin de ne pas reproduire des erreurs déjà commises, l'industrie s'est dotée d'instruments efficaces qui synthétisent l'expérience déjà acquise [SIG97]. Ainsi, la

(1) étude menée par l'ADAC (Automobile club d'Allemagne) et le CAR (Center Automotive Research)

confiance que l'on avait dans la sûreté d'un système nouveau était justifiée de par la réutilisation de solutions technologiques bien éprouvées. Avant la révolution mécatronique, l'exploitation d'une base de données suffisait à garantir que les nouveaux systèmes répondaient à des exigences de sécurité et de disponibilité.

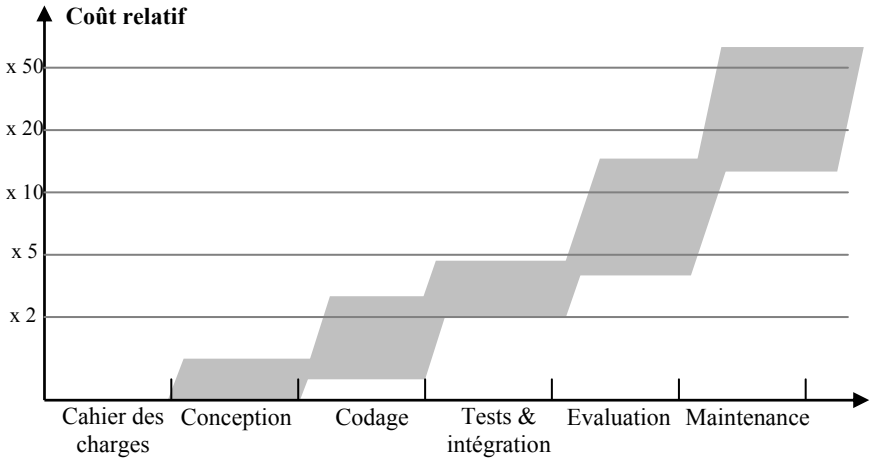


Figure 2 : Evolution du coût de correction d'une erreur durant les étapes du cycle de vie d'un logiciel [KOR92]

Remarquons que depuis plus de 20 ans, l'aéronautique et le spatial utilisent l'électronique (éventuellement programmé) pour commander des organes mécaniques dans des asservissements. La sûreté de fonctionnement y est assurée par des techniques de redondance massive (jusqu'à l'ordre 5, sur la navette spatiale américaine) et l'utilisation de composants fiabilisés (déverminage, tri, encapsulation spéciale) coûteux. Le retour d'expérience de ce domaine est difficilement exploitable en raison du coût prohibitif. L'automobile doit se contenter de composants grand public sans redondance massive.
La mécatronique a par la suite jeté un pavé dans la marre ; non seulement en raison de l'absence de retour d'expérience sur les nouvelles technologies employées (électronique, informatique,…), mais également en raison de la complexité accrue des systèmes à tous les niveaux : composants, organes, véhicule. Plusieurs facteurs contribuent à cette complexification ; tout d'abord, parce la conception d'un système mécatronique est largement tributaire de la réflexion et de l'action humaines. Il existe autant de solutions que de concepteurs. L'assurance de la qualité dans le processus de développement implique de maîtriser la sûreté de fonctionnement à chaque étape. Les activités de conception et de sûreté de fonctionnement sont indissociables [JAM01b]. L'avènement de la mécatronique étant relativement récent, cette constatation soulève de nombreux problèmes. En effet, il ne s'agit pas de rajouter des méthodes et des outils sans tenir compte des contraintes organisationnelles. Le manque de culture d'analyse de la sûreté, la faible maîtrise des méthodes et outils dans le cadre d'un processus de conception déjà établi, ne permettent pas l'intégration brutale de la dimension Sûreté dans les projets [HEN97]. Il faut donc progressivement changer les mentalités et donner aux acteurs les moyens de comprendre et d'intégrer les nouveaux outils employés.
A cela, s'ajoutent des problématiques propres à l'industrie automobile et aux systèmes embarqués. Comme il a été souligné, de nombreuses fonctionnalités réparties en sous-systèmes ou organes coopèrent entre elles. Il en résulte que la sûreté de fonctionnement doit s'étudier plutôt au niveau du système complet qu'au niveau composant. Les objectifs liés à des performances de sûreté de fonctionnement doivent être fixés au niveau du véhicule complet et non uniquement au niveau organe. *L'approche système* doit alors permettre

d'allouer des objectifs de sûreté localement. Cette démarche est également guidée par la considération des coopérations constructeur-équipementiers. Un certain nombre de sous-ensembles sont sous-traités par des équipementiers selon différents niveaux de délégation qui varient selon la nature des systèmes, allant de la pure sous-traitance au contrôle étroit des développements.

A partir d'exigences de sûreté de fonctionnement définies pour le système véhicule dans les spécifications, le constructeur alloue des exigences à des niveaux inférieurs de façon à respecter l'objectif global. Ces objectifs peuvent être qualitatifs (par exemple aucune panne simple ne doit conduire à un événement critique pour la sécurité), ou quantitatifs (en termes de probabilité d'apparition d'événements redoutés, de fiabilité, …). Pour les systèmes faisant appel fortement à la sous-traitance, le respect des exigences doit être justifié par le fournisseur. Certains systèmes étant très spécifiques au constructeur, celui-ci doit mettre en œuvre les démarches d'analyse adéquates dans le cadre d'un processus de conception. La maîtrise des risques est d'autant plus nécessaire que le système est critique vis-à-vis de la sécurité.

Enfin, la pression concurrentielle, les contraintes de temps, les exigences du client, le respect des normes législatives en vigueur sont autant de contraintes supplémentaires qui doivent encourager la mise en place d'une méthodologie bien stabilisée permettant d'améliorer l'assurance dans la qualité globale de la conception.

2. La sûreté de fonctionnement des systèmes embarqués

2.1. Introduction

La *sûreté de fonctionnement* d'un système peut se définir comme étant la propriété qui permet à l'utilisateur d'avoir une confiance justifiée dans le service qu'il délivre [LAP96].
Plus généralement, elle peut se définir comme la science des défaillances. Usuellement, la sûreté de fonctionnement d'un système s'exprime à l'aide de différents indicateurs ou attributs, mais complémentaires : la *Fiabilité*, la *Maintenabilité*, la *Disponibilité* et la *Sécurité* (FMDS). L'évaluation de ces différentes mesures fait appel à des calculs de probabilités. Ils reposent sur la connaissance des éléments pouvant défaillir (dans le cas d'éléments matériels) et de leur fréquence de panne dans le temps : les *taux de défaillance*. Tout composant matériel est susceptible de tomber en panne et d'affecter le fonctionnement nominal du système ou de la fonction. Les conséquences peuvent être bénignes ou critiques selon la nature du système et de l'implication du composant défaillant dans la fonction réalisée. Il est donc important d'évaluer le risque encouru par les utilisateurs de ce système et si nécessaire, déterminer les moyens qu'il faut mettre en œuvre pour *éliminer* ou *tolérer* ces défaillances.

La problématique à laquelle les ingénieurs sont confrontés dans la maîtrise de la sûreté de fonctionnement est toujours à peu près la même [SIG96] :

- Comprendre :
    - comment ça marche,
    - comment ça tombe en panne,
    - ce qui se passe en cas d'incident.

- Evaluer :
    - les probabilités des événements redoutés,
    - les dommages consécutifs aux incidents.

- Identifier et hiérarchiser :
    - les points faibles,
    - les scénarios de défaillance principaux,
    - les actions ayant le meilleur rapport efficacité/coût pour réduire les risques,
    - les parades pour limiter les conséquences.

- Fournir les éléments d'aide à la décision.

Le travail de l'ingénieur repose en grande partie sur une analyse des dysfonctionnements du système. Deux types de raisonnements logiques et complémentaires sont usuellement mis en œuvre :

- selon une approche *déductive* : l'ingénieur raisonne du général au particulier. A partir des événements redoutés identifiés, les causes sont recherchées à des niveaux inférieurs (composants),
- selon une approche *inductive* : l'ingénieur raisonne du particulier au général.

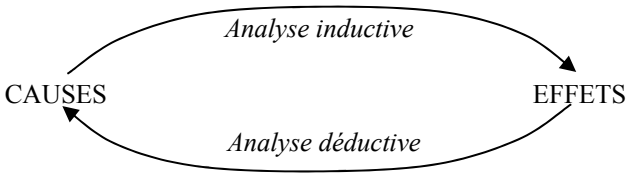


Figure 3 : Modes d'analyses des dysfonctionnements

Citons comme exemple d'approche déductive l'*arbre de défaillance*. Cette méthode, largement répandue dans l'industrie, consiste à représenter graphiquement au moyen d'une structure arborescente l'ensemble des combinaisons d'événements conduisant à l'occurrence d'un événement redouté, constituant la racine de l'arbre. Ces combinaisons s'expriment à l'aide de portes logiques (ET, OU, NON). Le succès de cette méthode résulte de sa simplicité conceptuelle et de la possibilité de mener simultanément une analyse qualitative (recherche des *coupes minimales*) et quantitative (calcul des probabilités d'occurrence des événements redoutés).
L'Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité (AMDEC) est un exemple d'approche inductive. Cette méthode prend la forme d'un tableau dans lequel sont répertoriés les composants susceptibles de défaillir, leurs modes de défaillance, les causes possibles de ces défaillances et les conséquences sur le système, la criticité (probabilité d'occurrence et gravité), et les actions à entreprendre. L'AMDEC peut être réalisée autant au niveau fonctionnel (décomposition en fonctions élémentaires), qu'au niveau structurel (décomposition en composants matériels ou logiciels).
De nombreuses méthodes et outils permettent de représenter, de façon simplifiée, les mécanismes et les interactions de défaillances d'un système. Les modèles abstraits présentent l'avantage d'être facilement compréhensibles par les différents acteurs du projet qui peuvent par la suite identifier les points faibles et apporter les modifications nécessaires pour renforcer la sûreté de fonctionnement. De plus, dans le cadre de l'élaboration d'une architecture matérielle d'un système, l'exploitation de modèles dysfonctionnels peut aider dans la

comparaison de plusieurs solutions potentielles. Généralement ces méthodes supportent une analyse quantitative (arbre de défaillance, diagramme de fiabilité, graphes de Markov,…).
Ces méthodes bien connues ont été pendant longtemps suffisantes. Néanmoins, face à la complexité et la spécificité des systèmes mécatroniques, ces méthodes se heurtent à leurs limites. Par exemple, les systèmes embarqués sont en interaction constante avec l'environnement (systèmes *réactifs*) et réagissent en temps réel à des événements par l'intermédiaire de capteurs en élaborant des opérations de contrôle-commande sur le processus. En particulier, les défaillances constituent une classe particulière d'événements qualifiés de stochastiques. Si des stratégies de tolérance aux fautes sont implémentées dans le logiciel embarqué, le système sera capable de se reconfigurer dynamiquement afin d'assurer une continuité du fonctionnement. Or, les approches classiques de la sûreté traditionnellement utilisées dans l'industrie sont statiques, c'est-à-dire qu'elles ne prennent pas en compte la dimension temporelle dans le modèle. Par exemple, les arbres de défaillance ne permettent de représenter que des combinaisons logiques et statiques de défaillances amenant le système dans un état redouté. Il n'est donc pas possible de différencier les scénarios où l'ordre d'apparition des défaillances intervient qualitativement ou quantitativement sur la nature ou la probabilité des événements redoutés. Une autre limitation de ces méthodes réside dans l'impossibilité de représenter l'influence réciproque des phénomènes stochastiques discrets liés aux défaillances et de la dynamique déterministe associée au processus physique. Cette discipline nouvelle est connue sous le nom de **fiabilité dynamique** [LAB02a]. Les méthodes d'évaluation dynamique telles que les graphes de Markov ou la simulation de Monte-Carlo sont plus aptes à traiter ce type de problématique, quitte à lever certaines difficultés intrinsèques à ces méthodes (explosion combinatoire du nombre d'états, hypothèses restrictives, volume de calcul important…). A ce titre, nous proposons dans le chapitre 3 une méthode de représentation du processus de défaillance à l'aide d'un graphe de Markov, tout en limitant le nombre d'états grâce à une méthode d'agrégation en prenant soin de conserver une interprétation physique de ces agrégats. Les interactions entre les événements stochastiques et les grandeurs physiques continues du processus seront également prises en compte et intégrées au modèle. Le concept de la fiabilité dynamique sera traité plus en détail dans les paragraphes suivants.

2.2. Quelques concepts et définitions

2.2.1. Choix des mesures de sûreté de fonctionnement

Lors de l'évaluation quantitative, un grand nombre de mesures de performance et de sûreté de fonctionnement permettent de caractériser la robustesse du système ou de valider des exigences de productivité. Les grandeurs les plus connues dans la littérature sont détaillées dans la section suivante.

Le choix des mesures dépend de la fonction et de l'utilisation du système. Deux catégories de mesures peuvent être distinguées [ZIE96] :

- La première catégorie concerne les systèmes non réparables. On suppose que le système étudié n'est soumis qu'au processus de défaillance, sans possibilité de réparation. Cette hypothèse est couramment posée pour évaluer la sûreté de fonctionnement de systèmes pour lesquelles la réparation n'est pas envisagée ou

envisageable. Le domaine de l'aérospatiale en est un exemple. Dans le domaine automobile, lorsqu'on cherche à évaluer les conséquences d'une défaillance catastrophique de certains systèmes critiques (fonctionnement intempestif de l'airbag, anti-blocage des roues, dysfonctionnement de la direction électrique…), la gravité de l'événement redouté est telle que la restauration du service n'est pas primordiale. Parmi ces mesures, on recense le MTTF, la fiabilité, la sécurité ou encore la disponibilité (ponctuelle) avant défaillance catastrophique. Tous les états redoutés sont alors considérés comme étant absorbants.

- La deuxième catégorie concerne les systèmes réparables. Le système étudié est soumis à la fois aux processus de défaillance et de réparation. L'hypothèse de réparabilité est faite pour les systèmes ayant un fonctionnement continu sur une longue période. Par exemple, l'étude d'une ligne de production, d'un réseau informatique industriel fait partie de ces systèmes. Les résultats d'une analyse quantitative se focaliseront plus sur la productivité, la disponibilité, les taux d'utilisation de stations ou de ressources, les niveaux moyens de stocks… Les mesures appartenant à cette catégorie sont la disponibilité (ponctuelle ou asymptotique), la maintenabilité, le MUT, le MDT, le MTTR et le MTBF.

Domaine de l'automobile :

Les systèmes embarqués dans l'automobile intègrent de plus en plus massivement de l'électronique et des composantes logicielles visant à remplacer des parties mécaniques stratégiques. Certaines fonctions critiques peuvent même être gérées par des systèmes de contrôle-commande. Les nouvelles solutions technologiques mises en place pour des raisons de coût, de performance ou de confort impliquent une remise en cause des méthodes d'analyse classiques et de reconsidérer les estimateurs de sûreté. En effet, les nouveaux systèmes embarqués sont des systèmes à plusieurs degrés de défaillance puisqu'il est possible de tolérer des défaillances grâce à des reconfigurations logicielles ou des redondances fonctionnelles. De ce fait, l'objectif des mesures portera soit sur la sécurité qui se traduira par une mesure de fiabilité ou de disponibilité avant défaillance critique, soit sur la délivrance d'un service correct non dégradé qui se traduira également par une mesure de disponibilité (disponibilité avant passage dans un état dégradé). Le type de défaillance est alors qualifié de bénigne par opposition à catastrophique.
Habituellement, les grandeurs représentant des temps moyens comme le MTTF sont également mesurées. Cependant, celles-ci sont moins riches en information que la fiabilité ou la disponibilité.

Mesures et unités utilisées dans l'automobile :

Les objectifs de sûreté de fonctionnement sont fixés dans les spécifications techniques. A PSA Peugeot Citroën, au niveau véhicule, ces objectifs sont de deux types :

- des objectifs de type *Fiabilité Perçue par le Client*, sous la forme d'un *TIC* (Taux d'Incidence Clientèle à un an). La notion de TIC recouvre tout défaut perceptible par le client. Il s'exprime en *ppm*.
- des objectifs de type *Sûreté de Fonctionnement*. Ils s'expriment en termes d'événement redouté de *sécurité* (s'ils peuvent engendrer un risque corporel ou des dégâts matériels irrémédiables), ou en termes de *panne* (indisponibilité du véhicule).

Les constructeurs automobiles utilisent souvent l'unité **ppm** ou **partie par million** dans les études de sûreté de fonctionnement. Cette unité correspond au nombre de systèmes ou

véhicules défaillants sur un an d'utilisation (soit en moyenne 500 heures effectives) rapportés au nombre de systèmes ou de véhicules produits. Ce chiffre est ensuite multiplié par 10^6 et s'exprime alors en ppm.

2.2.2. Définitions mathématiques des grandeurs de sûreté de fonctionnement

Nous rappelons ici quelques définitions principales normalisées [CEI90].

> *Définition : Sûreté de fonctionnement*
> *C'est l'ensemble des propriétés qui décrivent la disponibilité et les facteurs qui la conditionnent : fiabilité, maintenabilité et logistique de maintenance.*

La sûreté de fonctionnement est une notion très générale, qui ne se limite pas à un caractère quantitatif. Nous présentons néanmoins les principales mesures utilisées dans les analyses quantitatives.

> *Définition : Fiabilité*
> *La fiabilité est l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle donné.*

Cette notion est certainement l'une des plus anciennement définies avec le concept de la sécurité dans les années 50. Notons également que la fiabilité caractérise une probabilité de réussite d'une entité à accomplir une mission pendant une période donnée. Selon l'utilisation de cette grandeur, la notion de « période de temps » peut être remplacée par « nombre de cycles », « nombre de sollicitations », « nombre de kilomètres parcourus »,… Par exemple, on pourra parler de la fiabilité de 0,80 d'un type de pneu automobile déterminée sur une distance de parcours de 50 000 Kms en régime autoroutier.

La mesure de la fiabilité permet donc d'évaluer la capacité d'un système, sous-système ou d'une entité S à maintenir une fonction dans des conditions données continûment dans le temps. On la note R(t) (R comme Reliability) ; l'expression mathématique associée à la fiabilité s'écrit alors :

$$R(t) = \text{Probabilité}\ (\ S \text{ non défaillant sur } [0, t]\ )$$

R(t) est une fonction non croissante variant de 1 à 0 sur $[0, \infty[$.

> *Définition : Disponibilité*
> *La disponibilité est l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données ou pendant un intervalle de temps donné, en supposant que la fourniture des moyens nécessaires est assurée.*

La disponibilité est très certainement la notion la plus intuitive. L'automobiliste qui constate le matin que son véhicule refuse de démarrer, l'essuie-glace ne s'enclenchant pas en cas de pluie sont autant de cas où la perception du non-fonctionnement (ou « indisponibilité ») sont ressentis de manière instantanée.
Plus succinctement, la disponibilité est la probabilité pour que le système ou l'entité soit non défaillant à l'instant t. Cette grandeur est notée A(t) (Availability) et s'exprime mathématiquement de la façon suivante :

$$A(t) = \text{Probabilité (S non défaillant à l'instant t)}$$

L'indisponibilité $\overline{A}(t) = 1 - A(t)$ est également souvent employée.
Soulignons que pour des systèmes non réparables, la fiabilité et la disponibilité sont équivalentes. Sinon, $A(t) \geq R(t)$.

> *Définition : Maintenabilité*
> *Dans des conditions données, la maintenabilité est l'aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est accomplie dans des conditions données, avec des procédures et des moyens prescrits.*

La notion de maintenabilité est apparue dans les années 60 compte-tenu de l'importance extrême des dépenses entraînées par les travaux inévitables de maintenance, en particulier dans le domaine de l'aéronautique, des télécommunications ou de l'énergie atomique. Elle traduit l'aptitude d'un matériel à être remis en état de fonctionnement. La maintenabilité M(t) ne concerne que les systèmes réparables. D'un point de vue mathématique, elle se définit comme étant le complément à 1 de la probabilité pour que le système ne soit pas réparé sur l'intervalle [0, t] sachant qu'il est défaillant à l'instant t=0. Soit :

$$M(t) = 1 - \text{Probabilité (S non réparé sur [0, t])}$$

Cette fonction est non décroissante et varie de 0 à 1 sur $[0, \infty[$.
Dans le cas des systèmes de contrôle-commande traités dans ce manuscrit, l'hypothèse de non-réparabilité sera faite.

> *Définition : Sécurité* [VIL88]
> *C'est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques.*

En fait, le concept de sécurité est probablement le plus difficile à définir et à évaluer, car il englobe des aspects très divers. En effet, dans l'automobile, la sécurité est un terme générique qui définit souvent l'ensemble des dispositifs ajoutés aux fonctions élémentaires de conduite et contribuent à éviter ou diminuer les risques d'un accident potentiel (sécurité contre les accidents). Les dispositifs de *sécurité passive* (l'airbag, le prétentionneur de ceinture, les portes renforcées avec absorbeur, l'appuie-tête actif…) ont des actions de protection en cas d'accident, alors que les dispositifs de *sécurité active* ont des actions préventives (l'ABS ou Anti-Blocage de Roue, l'ESP ou contrôle de stabilité, l'ASR en anti-patinage des roues, l'AFU ou aide au freinage d'urgence…).

Dans le domaine de la sûreté de fonctionnement du logiciel, la terminologie de *sécurité-confidentialité* se rapporte à la non-occurrence de la divulgation non autorisée de l'information [LAP95], c'est-à-dire empêcher des utilisateurs d'accéder à des informations qu'ils ne sont pas autorisés à connaître.

Dans le cadre de l'étude quantitative présentée, le terme de sécurité sera relatif à la gravité des défaillances affectant le système étudié. On parlera plus précisément d'événement redouté de sécurité dans la mesure où l'occurrence d'un tel événement représente un risque pour l'usager (par opposition à événement redouté de disponibilité dont la criticité est plus faible). Ainsi, l'évaluation de la sécurité consistera à rechercher les conséquences de défaillances en termes

d'événements redoutés, à estimer leur gravité et leur fréquence d'apparition et envisager des actions de maîtrise (mise en place de barrières de sécurité) si nécessaires.

La sécurité se mesure donc par la probabilité que le système évite de faire apparaître des événements critiques ou catastrophiques. Ce concept peut devenir prépondérant dans une analyse de sûreté de fonctionnement, dans la mesure où une défaillance du système peut présenter un risque de dommage corporel à l'encontre des usagers.

2.2.3. Autres grandeurs

D'autres grandeurs peuvent être calculées à partir des mesures de probabilités. Contrairement aux précédentes, qui sont fonction du temps, les grandeurs suivantes caractérisent des durées moyennes [VIL88] :

- MTTF : Durée moyenne de fonctionnement d'une entité avant la première défaillance (Mean Time To Failure)
- MTTR : Durée moyenne de réparation (Mean Time To Repair)
- MUT : Durée moyenne de fonctionnement après réparation (Mean Up Time)
- MDT : Durée moyenne d'indisponibilité après défaillance (Mean Down Time)
- MTBF : Durée moyenne entre deux défaillances (Mean Time Between Failure)

Ces durées sont représentées sur le diagramme suivant :

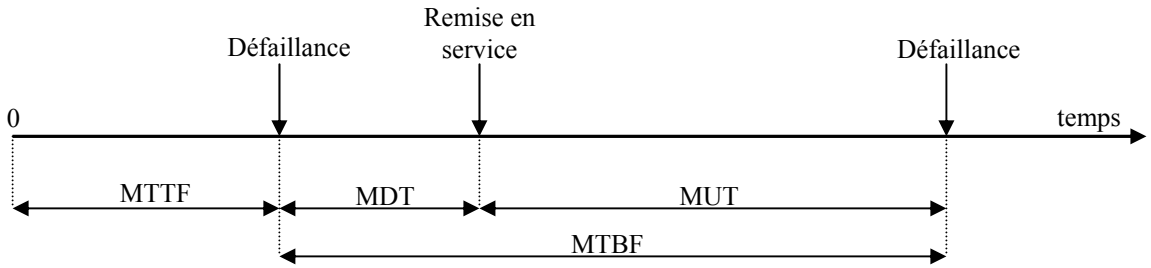


Figure 4 : Mesures moyennes de sûreté de fonctionnement

2.2.4. Taux de défaillance et de réparation

> *<u>Définition</u> : Taux de défaillance instantané [VIL88]*
> *C'est la limite, quand elle existe, du quotient de la probabilité conditionnelle pour que l'instant T d'une défaillance d'une entité E soit compris dans un intervalle de temps donné [t, t+Δt], par la durée de l'intervalle de temps, lorsque Δt tend vers 0 en supposant que l'entité n'a pas eu de défaillance sur [0, t].*

Ou encore en exprimant mathématiquement cette définition et en utilisant le théorème des probabilités totales :

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \frac{\text{Probabilité}\,[\text{E est défaillante entre t et t} + \Delta t, \text{ et E non défaillante sur } [0, t]\,]}{\text{Probabilité}\,[\text{E non défaillante sur } [0, t]\,]}$$

Soit, en utilisant la définition de la fiabilité R(t) :

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \frac{R(t) - R(t + \Delta t)}{R(t)}$$

$$\lambda(t) = \frac{-\frac{dR}{dt}(t)}{R(t)} \geq 0 \qquad [1]$$

Ainsi la connaissance de la fiabilité d'une entité après calcul ou simulation permet de connaître le taux de défaillance à chaque instant, et réciproquement.
La fiabilité est égale à :

$$R(t) = \exp(-\int_0^t \lambda(u).du) \qquad [2]$$

L'entité considérée peut être de natures diverses : du simple composant dont on cherche à évaluer la fiabilité ou le taux de défaillance jusqu'au niveau système dont on a calculé la fiabilité et dont on cherche à en déduire le taux de défaillance global.

Rappelons enfin que la fiabilité s'exprime très simplement en fonction de la répartition F de la loi associée à la variable aléatoire T mesurant la durée de bon fonctionnement de l'entité [PAG80] :

$$R(t)=1-F(t)$$

> *<u>Définition</u> : Taux de réparation instantanée*
> *C'est la limite, quand elle existe, du quotient de la probabilité conditionnelle pour que l'instant T d'achèvement de la réparation d'une entité soit compris dans un intervalle de temps donné [t, t+Δt], par la durée de l'intervalle de temps, lorsque Δt tend vers 0, en supposant que l'entité a été en panne sur [0, t].*

En procédant de la même façon, on montre que ce taux s'exprime en fonction de la maintenabilité :

$$\mu(t) = \frac{\frac{dM}{dt}(t)}{1 - M(t)} \geq 0$$

La maintenabilité s'écrit en fonction du taux de réparation :

$$M(t) = 1 - \exp(-\int_0^t \mu(du).du)$$

Les analyses dysfonctionnelles des systèmes automobiles embarqués seront réalisées en supposant les systèmes non-réparables. En effet, l'hypothèse de réparabilité et de maintenance (curative) n'est pertinente en général que pour l'étude des systèmes de production ; l'étude d'une installation sous-entend une possibilité de cycle fonctionnement-panne tout en assurant

une continuité du processus industriel. L'intervention d'un opérateur humain n'est pas envisageable pour les systèmes embarqués sur un temps de mission donné.

2.2.5. Evènements redoutés

Nous allons appeler Evènement Redouté (ER), les conséquences de l'occurrence d'une seule ou d'une séquence de défaillances amenant le système dans une situation de blocage en l'absence de réparation.

Ce terme restera volontairement assez général et pourra désigner une des situations suivantes :

- *événement acceptable* : pas de conséquence majeure portant atteinte à la sécurité de l'usager ou l'indisposant notablement.
- *événement indésirable* : c'est un événement ne devant pas se produire ou devant se produire avec une probabilité moins élevée au regard d'objectifs de sûreté de fonctionnement.
- *événement critique* : c'est un événement qui entraîne la perte d'une ou de plusieurs fonctions du système et pouvant provoquer des dommages importants au système ou à l'environnement, mais ne présente qu'un risque négligeable de mort ou de blessure.
- *événement catastrophique* : c'est un événement qui occasionne la perte d'une ou de plusieurs fonctions essentielles du système en causant des dommages importants au système ou à l'environnement et pouvant entraîner pour l'homme la mort ou des dommages corporels.

Soulignons que les événements qualifiés de catastrophiques sont surtout exploités dans l'aéronautique et dans le ferroviaire, domaines dans lesquelles les systèmes embarqués ont une forte criticité vis-à-vis de la sécurité. En effet, dans l'automobile, les pannes pouvant affecter les systèmes d'aide à la conduite, d'amélioration du comportement routier, ne sont en principe pas préjudiciables pour la sécurité. Un dysfonctionnement total induit une indisponibilité du système ou du véhicule. De plus, les contraintes de conception sont différentes entre les constructeurs de l'avionique et de l'automobile puisque les redondances matérielles massives ne sont pas envisageables pour des raisons de coût et d'encombrement pour l'automobile.
Néanmoins, cette vision des choses va certainement évoluer au cours des prochaines années par l'adoption des systèmes *x-by-wire*, étape ultime avant le pilotage automatique… PSA Peugeot Citroën est particulièrement actif dans ce domaine et travaille sur les concepts de *steer-by-wire (SbW)* et *brake-by-wire (BbW)*. Les fonctions traditionnellement assurées par la mécanique ou l'hydraulique sont converties en tout-électrique. Par exemple, pour la direction électrique (SbW), il n'y a plus de colonne de direction entre le volant et la crémaillère. Ce système, adopté depuis de nombreuses années par l'aéronautique, est qualifié de critique et nécessite le recours à des redondances matérielles et logicielles (actionneurs, capteurs, calculateurs, bus de communication, …).

Notons enfin qu'à PSA Peugeot Citroën, une grille de criticité permet d'attribuer un couple {gravité, probabilité d'occurrence} aux événements. La gravité se décline en 10 niveaux différents, allant de la défaillance minime (niveau 1) à la défaillance pouvant impliquer des problèmes de sécurité (niveau 10) [JAM01c].

2.3. Vers une architecture plus sûre

2.3.1. Généralités

Au cours du développement d'un système embarqué, le concepteur doit choisir entre plusieurs solutions d'architectures matérielles et logicielles répondant à des critères de performances et de sûreté de fonctionnement exprimés dans les spécifications. Tout système matériel/logiciel contient inévitablement des fautes qui se manifesteront potentiellement par l'apparition de défaillances au cours de la vie opérationnelle du système. Il est donc important d'évaluer les conséquences de ces défaillances grâce à des méthodes adaptées et déterminer l'architecture optimale répondant aux exigences de sécurité et de disponibilité. L'ingénieur dispose de moyens permettant *d'éviter les fautes* : choix de composants de qualité, déverminage, utilisation de composants autotestables, vérification du respect des normes de qualité après assemblage, utilisation de méthodes formelles,… [AUB87].
Malgré toutes ces précautions, l'évitement de fautes ne peut garantir la fiabilité absolue ou peut engendrer des coûts prohibitifs. Le concepteur peut alors introduire des mécanismes de *tolérance aux fautes* afin d'éviter que des erreurs ou des fautes entraînent une défaillance du système. Ces techniques sont basées sur le principe de la *redondance* ; ce dispositif est constitué de manière générale par : un élément primaire dont on peut tolérer les erreurs, un élément redondant pouvant réaliser tout ou une partie des fonctions de l'élément primaire, un élément de détection et un dispositif de réaction à cette erreur. Celle-ci peut alors être confinée, corrigée, ou bien l'élément défaillant peut être remplacé par un élément redondant suite à une *reconfiguration* du système.

Deux types de redondance peuvent être distingués [AUB87] :

- **La redondance statique** : le nombre d'éléments redondés peut être important (redondance *massive*) et chacun d'entre eux participe à la réalisation de la fonction. Il s'agit notamment des dispositifs de *vote majoritaire* pour lesquels le résultat final issu du vote résulte de la comparaison entre les différentes sorties des éléments redondés.
- **La redondance dynamique** : l'élément redondant ne participe à la fonction qu'après détection et réaction à l'erreur.

La référence [LAP96] détaille un certain nombre de stratégies de reconfiguration pouvant être mises en œuvre suite à la détection d'une erreur.

2.3.2. Le matériel

Les composants électroniques sont soumis à un stress important de la part de l'environnement qui constitue la principale cause de défaillance : humidité, vibration, température, champ électromagnétique,… L'évaluation prévisionnelle de la fiabilité de ces composants fait aujourd'hui principalement appel à des modèles analytiques [VIL88]. La connaissance de la fiabilité, ou bien encore du taux de défaillance des composants, est primordiale pour pouvoir évaluer la sûreté de fonctionnement d'une architecture matérielle à l'aide de méthodes d'analyse quantitative. Cependant, ces taux sont rarement constants et dépendent d'une multitude de paramètres liés aux conditions physico-chimiques. Il est donc nécessaire de prendre en compte les facteurs les plus influents dans les études. Il existe des modélisations de taux de défaillance [VIL88] qui considèrent, d'une part, un taux de défaillance générique et spécifique à la conception et, d'autre part, des coefficients qui tiennent compte des conditions

réelles d'utilisation. Ces modélisations sont surtout utilisées pour les composants électroniques. Le taux de défaillance λ s'exprime de la façon suivante :

$$\lambda = \lambda_b \pi_E \pi_A \pi_Q \pi_n$$

où les coefficients π sont appelés coefficients d'influence.

- λ_b est le taux de défaillance de base. Il est obtenu à partir d'essais de fiabilité effectués sur des composants dans des conditions normalisées. Généralement, il prend en compte les contraintes normalisées (niveau de tension, de puissance,…) et la température.
- π_E est le coefficient d'environnement. Il prend en compte les conditions opérationnelles autres que la température (vibrations, chocs…).
- π_A est le coefficient d'ajustement. Il prend en compte les contraintes secondaires et des conditions d'utilisation particulières.
- π_Q est le coefficient de qualité. Il prend en compte la qualité avec laquelle le composant a été conçu.
- π_n est le coefficient d'ajustement lié à d'autres facteurs pertinents vis-à-vis du taux de défaillance (par exemple le nombre de cycles de fonctionnement répétés).

Les conditions d'environnement et la température représentent des paramètres très importants dans le domaine de l'automobile. En effet, l'électronique embarquée est soumise à un ensemble de contraintes extrêmement sévères (température variant de –40° à +110°C, vibrations, humidité et hydrocarbures).

Les fournisseurs doivent justifier le niveau de fiabilité des composants en fonction de contraintes particulières spécifiées par le constructeur. Soulignons qu'il est également indispensable de connaître l'ensemble des modes de défaillance afin de mettre en place des stratégies spécifiques de détection et de correction des erreurs.
Des techniques de tolérance aux fautes matérielles peuvent être exploitées afin de renforcer la sûreté de l'architecture matérielle. Nous allons passer en revue les principaux éléments constituant un système mécatronique [ZIE96]:

a) Les capteurs : c'est la source principale de défaillance dans les systèmes embarqués, notamment en raison des contraintes d'environnement sévères. Les principaux modes de défaillance habituellement considérés sont la défaillance en valeur (les données sont erronées), la défaillance temporelle (l'information capteur est envoyée trop tard) et la défaillance par arrêt (le capteur n'envoie plus de données, ou reste figé à une valeur constante).
   Les méthodes de tolérance permettant d'assurer une continuité du fonctionnement en présence de ces modes sont :
   - La réplication du capteur : elle permet de tolérer des fautes intermittentes et permanentes qui sont dues à des défaillances en valeur ou par arrêt,
   - la relecture du capteur : elle permet surtout de tolérer des fautes transitoires dues à des défaillances en valeur ou temporelles,
   - la mise en place d'une diode de protection permet de tolérer la présence d'une surtension du capteur,
   - un filtre passe-bas peut être prévu pour tolérer les fautes transitoires dues à des défaillances en valeur,
   - etc…

b) Les actionneurs : ils constituent également une source importante de défaillance. Les modes de défaillance considérés sont : la défaillance en valeur (l'actionneur agit de façon trop forte ou trop faible), la défaillance temporelle (il agit avec un décalage temporel) et la défaillance par arrêt (l'actionneur ne fait plus aucune action, ou une action constante). La technique de tolérance la plus courante consiste à introduire une redondance :
   - deux actionneurs sont mis en série ; par exemple deux interrupteurs mis en série permettent de réduire la probabilité de gonflement intempestif de l'airbag,
   - deux actionneurs sont mis en parallèle : un seul actionneur est nécessaire pour effectuer l'action souhaitée ; cette technique permet de tolérer une défaillance par arrêt de l'un des actionneurs,
   - trois actionneurs forment un voteur : l'action effectuée correspond à la majorité du vote ou à la moyenne des trois actionneurs.

c) Les calculateurs : ces composants sont en général plus fiables que les capteurs. Néanmoins, leur complexité rend difficile la connaissance des nombreux modes de défaillance possibles. De nombreuses techniques de traitement d'erreur sont alors utilisées. Les calculateurs permettent de diagnostiquer la présence d'erreur par des contrôles temporels et d'exécution, par la duplication et la comparaison d'informations, puis un recouvrement d'erreur permet ensuite de substituer l'état erroné par un état exempt d'erreur. Nous ne rentrerons pas davantage dans les détails. Le lecteur intéressé pourra se référer à [LAP96] pour des compléments d'information.

2.3.3. Le logiciel

Les avantages du logiciel sont nombreux et contribueront à une utilisation de plus en plus massive dans les systèmes embarqués. Le logiciel permet :
- de multiplier les modes de fonctionnement sans réduire la sûreté du système,
- d'améliorer la disponibilité par les possibilités de diagnostic embarqué,
- d'assurer la modularité et la flexibilité de la conception.

De plus, contrairement au matériel, il n'y a pas de vieillissement et il bénéficie d'un coût peu élevé lors de son implémentation.

Cependant, comme nous l'avons déjà évoqué, c'est certainement la technologie la plus problématique. En effet, malgré ses avantages, le logiciel pose des problèmes de développement qui sont différents du matériel :
- il est très sensible aux erreurs aléatoires et temporaires,
- il est extrêmement difficile de quantifier les causes de défaillance,
- il peut être très complexe, engrangeant des procédures de validation beaucoup plus poussées.

Les erreurs du logiciel sont principalement introduites lors de la conception, un travail important de *vérification* et de *validation* du logiciel doit être mené tout au long du cycle de développement.
La vérification consiste à s'assurer qu'un système a été construit correctement (*bien* construire le système) alors que la validation consiste à s'assurer que le système est conforme aux spécifications (construire le *bon* système).

L'objectif principal de la vérification du logiciel est de révéler les fautes de conception. Ces fautes ont pu être introduites durant n'importe quelle phase du cycle de développement ; il est néanmoins important de les révéler le plus tôt possible, pour des raisons de coût et d'efficacité.

La validation recouvre deux activités principales :

- **l'élimination des fautes** : c'est-à-dire comment minimiser, *par vérification*, la présence de fautes ;
- **la prévision des fautes** : comment estimer, *par évaluation*, la présence, la création et les conséquences de fautes.

Vérification et Validation (V&V) sont donc deux activités intimement liées.

L'élimination des fautes comporte trois étapes : vérification, diagnostic et correction. La vérification consiste à déterminer si le système satisfait à des propriétés générales ou spécifiques, appelées conditions de vérification [ARL95]. Si des propriétés ne sont pas vérifiées, les deux autres étapes sont entreprises et une nouvelle phase de vérification doit être réalisée afin de s'assurer que l'élimination des fautes ne s'est pas accompagnée de la création de nouvelles fautes. Il existe deux classes de vérification, en fonction de l'exécution ou non du logiciel : statique ou dynamique [LAP96] (cf. figure 5).

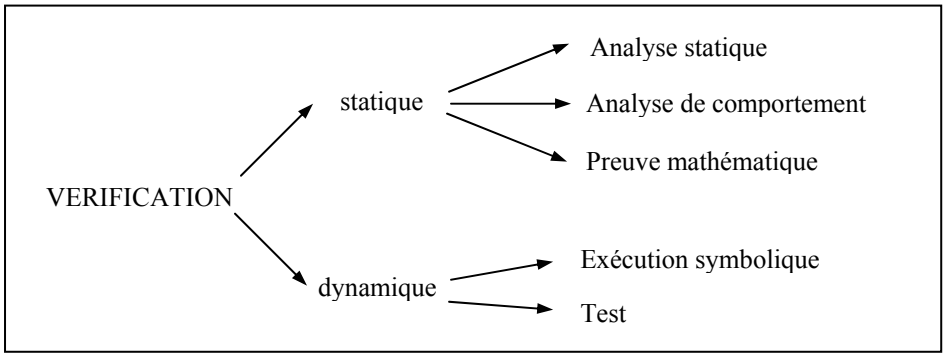


Figure 5 : Les techniques de vérification [LAP96]

Parmi ces techniques, la plus couramment utilisée à PSA peugeot Citroën est certainement le test. Il consiste à exécuter un programme en lui fournissant des valeurs numériques : les entrées de test. Le test exhaustif étant quasiment impossible, l'ingénieur est amené à sélectionner de manière pertinente un sous-ensemble des valeurs d'entrée possibles. Par exemple, celui constitué par les valeurs les plus probables et/ou les valeurs les plus critiques vis-à-vis de la fonction à assurer.

Classiquement, on distingue trois types de test pendant le développement du logiciel :

- **le test unitaire** : il permet de vérifier les fonctions élémentaires indépendamment de leur environnement,
- **le test d'intégration** : il permet de vérifier un ensemble de fonctions et de leur intégration,
- **le test système** : la vérification concerne l'ensemble du système matériel/logiciel et est effectué sur banc de test.

Evoquons enfin l'approche par *injection de fautes*. Celle-ci consiste à observer le comportement du système en présence de fautes, introduites volontairement dans le modèle. Cette technique permet de valider les stratégies de tolérance aux fautes mises en place dans le modèle (validation des redondances, reconfigurations logicielles ou matérielles, …). L'injection de fautes peut donc être vue comme une forme de test par rapport à une classe particulière d'entrées : les fautes.
Cette technique a notamment été exploitée pour vérifier le comportement d'un système de gestion automatique des portes d'un métro dans [SCH02]. Le principe consistait à coller certaines variables booléennes en entrée du modèle à la valeur 0 ou 1 puis à simuler le système afin d'observer son comportement. Bien qu'efficace (elle a mis en évidence certaines faiblesses dans la gestion des reconfigurations), cette méthode demeure assez lourde à mettre en place puisque les fautes ont été injectées manuellement. De ce fait, les tests se sont limités à l'injection de pannes simples.

Pour résumer, insistons sur le fait que les tests montrent la présence de défauts, mais jamais leur absence ! La vérification ne peut généralement pas démontrer la perfection du programme. Pour combler cette lacune, la catégorie des méthodes formelles apporte une solution. Ces méthodes font appel à des langages de spécification formelle, de preuve de théorème ou vérificateur de preuves. Pendant longtemps, celles-ci sont restées une affaire de spécialistes, en raison de la rigueur mathématique nécessaire quant à leur application. L'intégration de ces méthodes permet de démontrer « mathématiquement » qu'une propriété, préalablement formulée, est vérifiée. Nous évoquerons ultérieurement ce type de méthode plus en détail.

2.4. Fiabilité dynamique des systèmes hybrides

2.4.1. Concept

Introduisons cette notion par deux exemples.
Considérons un processus industriel devant être maintenu à une température constante à l'aide d'un système de refroidissement. L'évolution dynamique de l'ensemble des variables physiques associées au processus (température, débit, pression,…) est contrôlé à l'aide d'un dispositif de contrôle-commande en surveillant par exemple le dépassement de certains seuils caractéristiques. Ainsi des variations de certaines grandeurs physiques vont provoquer des changements d'état du dispositif de contrôle qui va agir en conséquence sur ces grandeurs pour les maintenir dans un intervalle de sécurité. Inversement, l'apparition d'une défaillance sur ce système de refroidissement va provoquer une augmentation de la température. Cette défaillance se répercute donc directement sur l'évolution dynamique du processus physique.

Le second exemple sera traité plus amplement dans le chapitre 4. Considérons un réservoir dont le niveau est maintenu entre deux seuils à l'aide de deux pompes et d'une vanne. La sollicitation de ces actionneurs dépend directement de l'évolution du processus physique représenté par le débit et le niveau de liquide. L'état du système est caractérisé par l'état du processus et l'état de la partie contrôle-commande assurant la régulation. Supposons à présent qu'une défaillance affecte la pompe principale. L'état du système est modifié par la perte de cette pompe. Une reconfiguration globale est alors enclenchée en activant la pompe de secours. Supposons maintenant que les modes de défaillance des actionneurs sont « blocage

dans la position courante ». Une défaillance aura une répercussion différente selon l'état du système à l'instant d'apparition de la défaillance (actionneur en position ouverte ou fermée).
Ces deux exemples montrent que le comportement du système résulte de l'interaction entre la composante dynamique et déterministe de l'évolution des grandeurs physiques et la composante discrète et stochastique associée au processus de défaillance.

L'évaluation des paramètres de la sûreté de fonctionnement nécessite de prendre en compte ce couplage entre processus déterministe (continu et/ou discret) et processus stochastique. C'est le domaine connu sous le nom de *fiabilité dynamique*.

[LAB02a] en donne une définition : *« c'est la partie des analyses probabilistes de sûreté qui étudie de manière intégrée le comportement des systèmes affectés par une évolution dynamique sous-jacente »*.

Initialement, la fiabilité dynamique a été qualifiée de *dynamique probabiliste* [LAB02]. La formulation proposée nous semble tout à fait intéressante dans le contexte de nos travaux. Le comportement dynamique d'un système peut être vu comme une succession de sections d'évolutions déterministes connectées entre elles par des événements aléatoires régis par des lois probabilistes. Un tel comportement est décrit par un *processus stochastique déterministe par partie*.

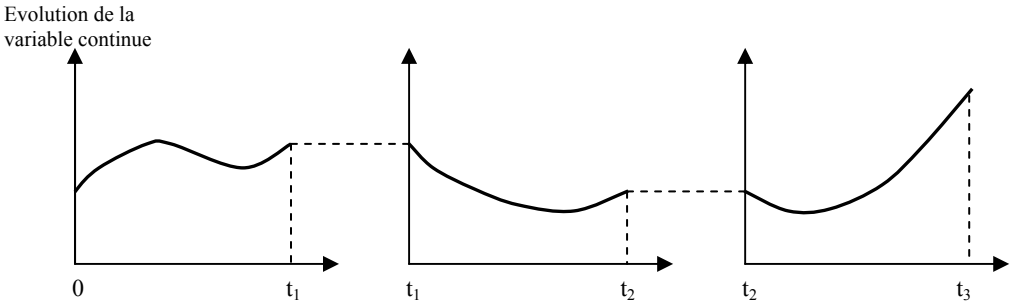


Figure 6 : Evolution d'un système soumis à des changements d'état

L'évolution dynamique des variables physiques du processus obéit à des lois déterministes (pouvant par exemple être modélisées par des équations algébro-différentielles [CHAM98a], [CHAMP98c]) jusqu'à ce que des événements affectent le système. Par exemple, le franchissement d'un seuil de sécurité, l'occurrence d'une défaillance modifie l'état actuel du système, c'est-à-dire l'évolution des variables physiques. Dans la nouvelle configuration, l'évolution du système est alors caractérisée par un nouvel ensemble d'équations différentielles jusqu'au prochain changement d'état. La figure 6 illustre l'interaction existante entre les variables physiques et les changements d'état du système.
L'analyse du comportement et de la sûreté d'un système caractérisé par ces différentes interactions a fait l'objet de nombreux travaux. Les auteurs ont notamment souligné l'intérêt de disposer d'une modélisation permettant de représenter d'une part l'évolution des variables continues et, d'autre part, l'ensemble des états discrets dans lesquels le système peut se trouver ainsi que les phénomènes stochastiques discrets liés aux défaillances. Cette modélisation des systèmes qualifiés *d'hybrides* est traitée dans ce mémoire.

2.4.2. Quelques approches existantes

Par nature, les réseaux de Petri conviennent bien pour modéliser des changements d'états discrets par le franchissement de transitions. Couplés à un système d'équations algébro-différentielles pour représenter l'évolution de grandeurs continues, les réseaux de Petri répondent alors aux besoins de modélisation des systèmes hybrides. Cette approche a été proposée dans [CHA98b]. L'outil MOCA-RP a servi pour modéliser le système en réseau de Petri stochastique temporisé. Le couplage avec le moteur de calcul continu repose sur l'utilisation de lois spéciales associées à certaines transitions du réseau de Petri, ces lois générant des délais issus d'un calcul par inversion des lois dynamiques de la partie continue.
L'analyse du comportement global est alors supportée par une simulation de Monte-Carlo.

Une seconde approche également proposée dans [CHA98b] et exploitant les réseaux de Petri, consiste à discrétiser les variables continues et en les modélisant par une certaine quantité de jetons dans des places du réseau. L'évolution des variables se traduit par l'ajout ou le retrait de jetons par le franchissement de transitions.

Bien qu'efficaces sur les cas d'application traités, ces méthodes de modélisation souffrent du manque de clarté dans la représentation de la partie continue. Cette limitation est également constatée dans le formalisme des réseaux de Petri hybrides [DAV89]. Au niveau industriel, ces approches sont donc difficilement adaptables à des systèmes beaucoup plus complexes.

2.4.3. Conclusion

Le domaine de la fiabilité dynamique doit permettre de combler certaines lacunes propres aux méthodes statiques. En effet, bien que rarement vérifiées, l'utilisation des méthodes statiques induit un certain nombre d'hypothèses :

- les conséquences d'un scénario de défaillance ne dépendent pas de l'ordre dans lequel apparaissent ces défaillances, autant qualitativement, que quantitativement,
- les instants d'apparition de ces défaillances n'ont pas d'impact non plus sur la nature et la probabilité des événements redoutés,
- le processus de défaillance est indépendant de l'évolution dynamique du système.

Néanmoins, la plupart des problèmes de fiabilité dynamique sont résolus en exploitant la simulation de Monte-Carlo sur un formalisme supportant la représentation hybride. Outre les difficultés relatives au temps de calcul nécessaire pour aboutir à des résultats précis, cette méthode de résolution n'offre pas de support de représentation dysfonctionnelle à l'instar des arbres de défaillance ou des graphes de Markov.
Cette nécessité de disposer d'un modèle décrivant l'ensemble des mécanismes de défaillance nous apparaît primordiale dans le contexte industriel, dans l'objectif de faciliter les communications et les échanges entre constructeur et équipementiers, mais également de capitaliser l'expérience acquise dans les projets en cours.

3. Nécessité d'une méthodologie de conception

3.1. Introduction

Pour concevoir un système, l'ingénieur dispose de techniques et de moyens variés. Quelle que soit la nature ou la complexité de ces systèmes, il se doit de respecter un certain nombre d'exigences [CAL92] :

- satisfaire les besoins du client : en d'autres termes, le système final doit être conforme aux exigences de fonctionnement établies dans le cahier des charges,
- respecter les délais et les coûts,
- satisfaire des critères de qualité, en particulier en termes de sûreté de fonctionnement.

Le respect de ces contraintes face à la complexité des systèmes mécatroniques passe nécessairement par la définition de « règles » permettant de passer rigoureusement de l'expression d'un besoin à la réalisation du produit. L'ensemble des règles et des méthodes appliquées constitue la méthodologie.

Des études montrent que les principaux défauts de conception résultent [LEN00] :

- de besoins mal spécifiés,
- d'une déformation des besoins dans le temps,
- d'une modification sauvage, parfois faite avec de bonnes intentions,
- de la perte de savoir-faire,
- du pari technologique (qui consiste à penser que si quelque chose marche pour 3m/s, elle devrait marcher pour 10m/s…),
- d'une mauvaise définition d'interfaces,
- de la pression de la concurrence,
- d'une modification fonctionnelle.

L'objectif d'une méthodologie consiste précisément à améliorer les méthodes de conception pour éviter que des défauts n'apparaissent, dans le meilleur des cas, avant la phase de production.

3.2. Etat des lieux : l'Approche Système

Avant l'ère mécatronique, un organe automobile était conçu de façon à remplir une fonction locale bien définie. Chaque fonction pouvait être étudiée et développée indépendamment des autres et l'implication de la sûreté de fonctionnement se résumait à la réutilisation de modèles génériques issus du retour d'expérience. Cependant, cette approche ne permet pas de considérer les risques inhérents à un nouveau système exploitant des technologies différentes et de complexité accrue. En effet, le choix d'une architecture fonctionnelle, matérielle, des lois de pilotage, ne doit plus se limiter à la tenue de performances locales mais doit être lié à des objectifs globaux de l'automobile. Cette approche est motivée par les interactions des différents sous-systèmes et par l'existence de coopérations constructeur/équipementiers. Les performances de sûreté de fonctionnement doivent être formulées au niveau le plus haut, c'est-à-dire au niveau du système complet, puis être déclinées à des niveaux inférieurs (sous-systèmes, organes, composants) suivant la structure arborescente du véhicule. Il n'est pas nécessaire de descendre tout en bas de l'arborescence, si les exigences sont justifiées à des

niveaux intermédiaires ou si les analyses de sûreté de fonctionnement sont traitées par les fournisseurs. Par conséquent, la démarche de sûreté de fonctionnement au niveau global se résume en trois points :

1. *l'Analyse des Risques* visant à identifier les événements redoutés critiques pour la sécurité et la disponibilité,
2. l'allocation d'objectifs de sûreté,
3. l'étude de la sûreté par les méthodes et outils adaptés.

Cette *Approche Système* retenue par PSA Peugeot Citroën permet d'assurer une cohérence globale de l'ensemble des études de sûreté de fonctionnement à chaque niveau intermédiaire tout en respectant l'objectif global à tenir au niveau véhicule et en prenant en compte les contraintes organisationnelles liées à la sous-traitance [HEN97].

Soulignons également l'importance de disposer de méthodes et outils de sûreté de fonctionnement. Pendant longtemps, les constructeurs automobiles étaient assimilés à des *intégrateurs de systèmes* en raison de la forte sous-traitance des équipements à des fournisseurs, les analyses de sûreté étant alors confiées à ces derniers. Cependant, pour certains systèmes spécifiques aux constructeurs, les analyses doivent être menées au sein de l'équipe projet.

3.3. Vers une méthodologie de conception intégrant la sûreté de fonctionnement

3.3.1. Cycle de développement

Le cycle de vie généralement utilisé pour décrire le développement des systèmes embarqués est le « cycle en V ». Celui-ci présente un découpage du processus de développement en deux branches. La branche descendante du « V » décrit les phases de conception en allant du général, l'expression des besoins, au particulier selon une démarche de raffinements successifs. La branche ascendante détaille les phases d'intégration et de validation correspondant à chaque phase de conception.

Le principal objectif de ce type de démarche en V est de retarder le choix de la technologie de réalisation. Cela permet de se concentrer sur la spécification et la conception, en faisant abstraction de la solution technologique dans les premières phases du développement. Cette démarche est issue de la méthodologie MCSE (Méthodologie de Conception des Systèmes Electroniques) [CAL90].

Ainsi, les phases de spécification et de conception conduisent à définir des niveaux de description de plus en plus détaillés. Les phases d'intégration, de test et de vérification permettent d'évaluer la conformité de la réalisation pour chaque niveau de la conception en commençant par les parties les plus élémentaires puis en remontant jusqu'au système complet. Le cycle en V le plus général correspond au système véhicule. Celui se décompose alors en plusieurs niveaux hiérarchiques par raffinements successifs du niveau véhicule complet jusqu'à la définition des sous-systèmes organes. Cette structure hiérarchique suggère un certain nombre de remarques :

- toute réalisation est un constituant développé comme un tout mais intégrable dans un ensemble plus complexe,

- pour chaque constituant (système, sous-système, organe, composant,…) on retrouve les phases de spécification, de conception, de réalisation, d'assemblage et de test. Ces terminologies ne sont pas liées au niveau de description.
- un système complet (par exemple au niveau véhicule) ne devient opérationnel que lorsque tous ses constituants sont réalisés et testés.

Nous nous contentons de présenter le cycle en V correspondant au développement d'un système « organe » associé à la réalisation d'une fonction donnée. Le caractère multi-technologique du système se traduit par le développement simultané de la composante matérielle et de la composante logicielle (cf. figure 7). Toujours en passant par une spécification puis une conception, la partie matérielle est décomposée en un ensemble de fonctions ou de constituants. Ces constituants, s'ils existent, sont directement utilisables, sinon il s'agit de les concevoir. La hiérarchie s'arrête lorsque tous les constituants retenus existent. La réalisation est alors possible, puis progressivement l'assemblage et le test en remontant la hiérarchie. De même la partie logicielle est décomposée en constituants (tâches ou modules). La phase de réalisation correspond au codage. Les modules codés sont testés puis assemblés au niveau hiérarchique supérieur (organe).

Chaque étape de la conception a recours à une modélisation plus ou moins abstraite. L'expression des besoins exprime l'ensemble des fonctionnalités que doit accomplir le système ainsi que l'ensemble des exigences techniques (performances, fiabilités, …). Cette description de haut niveau permet ensuite d'identifier et de structurer l'ensemble des fonctions sous la forme d'une architecture fonctionnelle. Cette description fonctionnelle permet également de spécifier les transferts d'informations entre les fonctions et l'environnement. Pour les systèmes complexes, en particulier les systèmes réactifs, la spécification fonctionnelle est complétée par des modèles de comportements susceptibles d'être validés dynamiquement.

La définition d'une méthodologie orientée sûreté de fonctionnement permet d'identifier au plus tôt les risques potentiels, de les classer et d'envisager des actions correctives avant que les choix de conception ne soient figés. Une telle démarche permet également de justifier au plus tôt le choix d'une architecture fonctionnelle répondant aux critères de sûreté avant que la complexité du modèle, encombré par un niveau de détail trop fin, ne nuise à la compréhension globale du système et ne complique inutilement l'analyse dysfonctionnelle.

De plus, l'emploi de règles de modélisation et de structuration, d'analyse formelle de vérification et de validation pendant la phase de description comportementale est encouragé. En effet, l'effort apporté par un contrôle et une modélisation rigoureuse du modèle permet de vérifier, quel que soit le niveau d'abstraction, la cohérence, la complétude des variables introduites et que le système vérifie un certain nombre de propriétés générales relatives à une bonne spécification (absence de *blocage*, *atteignabilité*, …). L'ingénieur aura également la garantie que des erreurs de conception ne soient pas introduites par inadvertance.
Ainsi, durant le cycle de développement, un effort plus important doit être apporté au niveau de la spécification et de la conception (cf. figure 8).

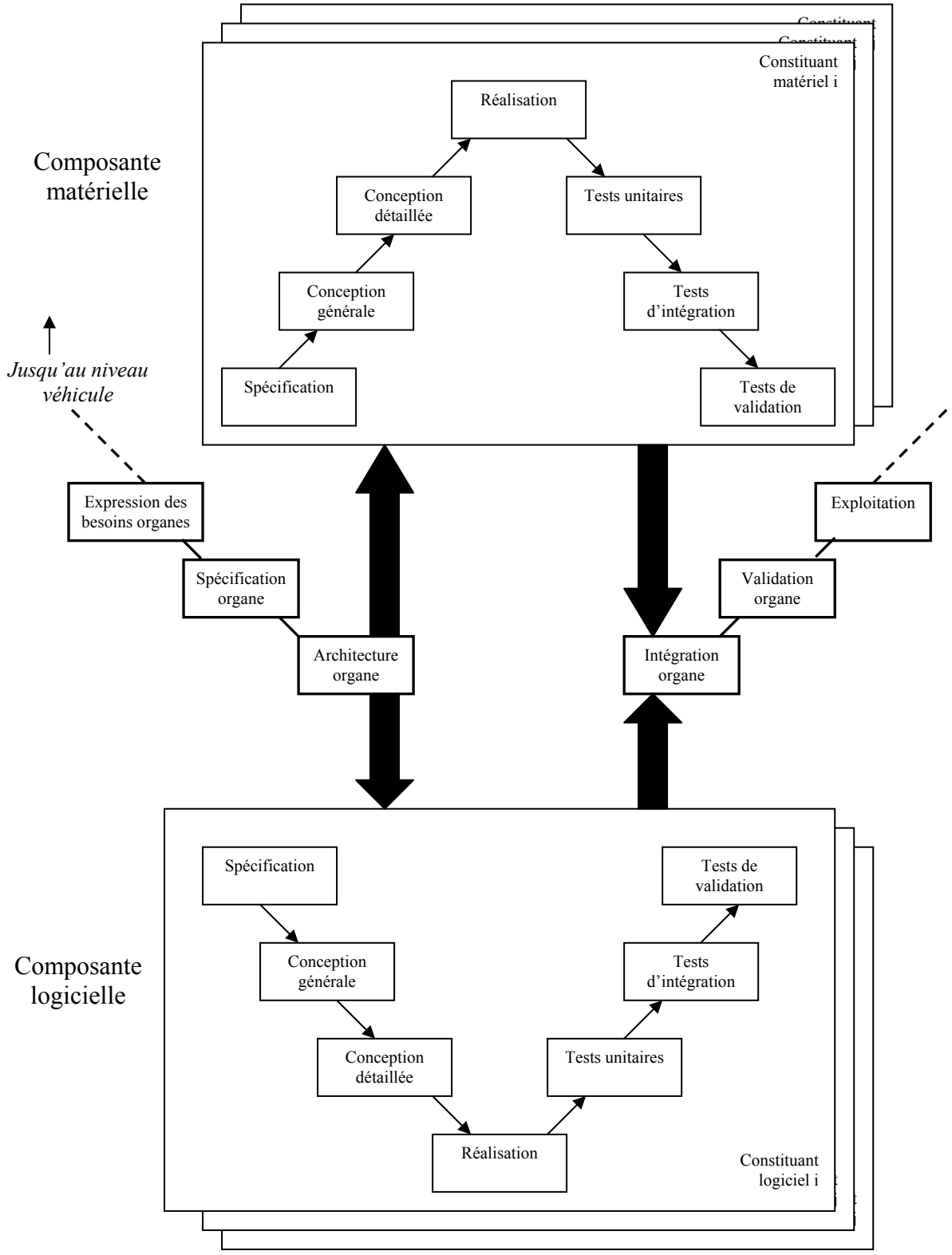


Figure 7 : Cycle en V d'un système mécatronique

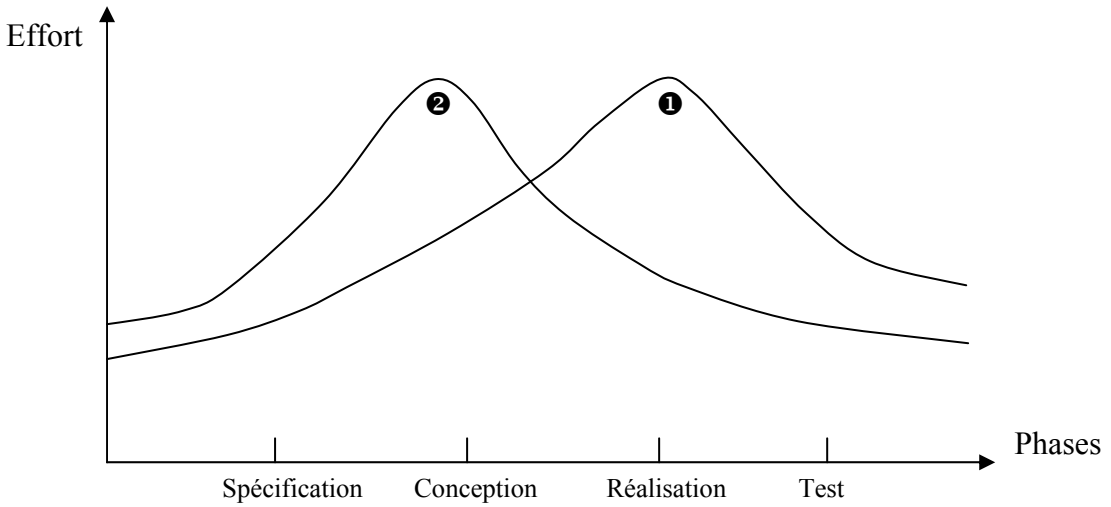


Figure 8 : Répartition de l'effort [CAL90]

La courbe ❶ correspond à une démarche traditionnelle pour laquelle un effort important est investi durant la phase de réalisation. Ceci correspond au fait qu'une grande partie du travail d'analyse finit par être entrepris à ce stade. La courbe ❷ stipule la nécessité d'un effort important en spécification puis en conception. L'investissement supérieur en ressources dès les premières phases pour la courbe ❷ est ensuite largement compensé par une réduction des coûts pour les phases de réalisation et de test.

Concernant le logiciel, l'accentuation de l'effort de spécification/conception prend une importance particulière. Le concept de logiciel intrinsèquement exempt d'erreur étant inconcevable et son développement demeurant une activité humaine, la définition de méthodes, d'outils d'analyse, le choix d'un formalisme de description adéquat au sein d'une méthodologie constitue une garantie de la qualité du système.

3.3.2. L'intégration des méthodes formelles dans une méthodologie

Les composants matériels ont connu de nombreux progrès depuis l'invention des transistors et de l'arrivée des circuits intégrés en 1959. Le prix des circuits intégrés a chuté et le nombre de circuits par puce a augmenté de façon considérable d'année en année tout en étant de plus en plus fiables. De tels progrès ne se sont pas produits pour le logiciel qui est devenu, lui aussi, plus complexe mais moins fiable [BOW93]. La validation et la vérification sont devenues incontournables et ont pris une importance considérable dans le cycle de développement des fonctions de contrôle dans les systèmes embarqués. Ainsi, à PSA, les étapes de tests unitaires et de validation constituent des maillons essentiels dans la branche ascendante du cycle en V. Les premiers permettent de valider les modules logiciels de façon autonome pour s'assurer que les constituants de base sont sains. Les seconds sont effectués sur des bancs de test et par des essais en réel et permettent de garantir que la réalisation est conforme aux spécifications du système.
En complément de ces vérifications de nature empirique, l'utilisation de méthodes formelles constitue un moyen supplémentaire et efficace d'accroître la sûreté de fonctionnement du logiciel en éliminant considérablement le nombre d'erreurs de conception. En effet, elles peuvent être utilisées dans de multiples contextes ; en particulier dans les étapes cruciales de spécifications, de conceptions, c'est-à-dire très tôt lorsque le système est encore relativement

abstrait et avant que la complexité des modèles analysés devienne inextricable (cf. figure 9). La conformité de ces modèles est assurée soit par l'utilisation de langages de spécification formels qui garantissent par construction que le système vérifie certaines propriétés (par exemple les réseaux de Petri [JAM01c], la méthode B), soit par l'application de « prouveurs » démontrant mathématiquement que les propriétés formulées dans un langage approprié sont vérifiées (par exemple les outils de Model-Checking, cf. [SCH99]).
L'introduction de méthodes formelles ne doit pas éliminer les tests puisque ceux-ci sont nécessaires pour valider le système complet matériel/logiciel après intégration mais peut simplifier les procédures en améliorant considérablement le rendement de l'effort de test pour les parties critiques.

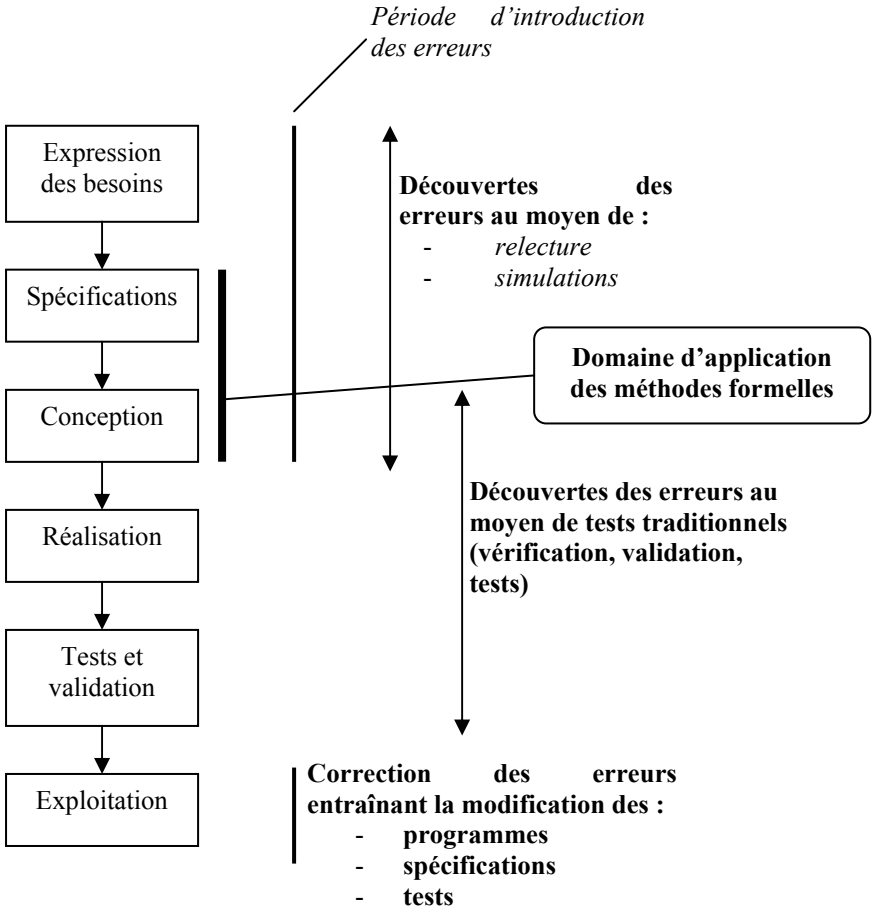


Figure 9 : Place des méthodes formelles dans le cycle de développement [OFT97]

La vérification formelle dans le cadre d'une analyse fonctionnelle est restée pendant longtemps une spécialité de la recherche informatique, peu exploitée dans l'industrie [BOW83], ceci pour plusieurs raisons plus ou moins légitimes. Premièrement en raison des limites technologiques de l'informatique. Les techniques telles que le model-checking consistent à explorer le graphe des états accessibles d'un système, dont le nombre croît exponentiellement avec sa taille. Les systèmes étudiés se limitaient à des cas d'école en raison du risque d'explosion combinatoire du nombre d'états générés. Un arbre de 10 millions d'états en mémoire centrale nécessite entre 40 et 80 millions d'octets selon la taille des cellules, la taille des systèmes industriels pouvant allègrement dépasser le milliard d'états. Néanmoins, les progrès que connaît l'informatique repoussent continuellement les limites, autorisant ainsi le traitement de systèmes de plus en plus conséquents. De plus, la technique

des méthodes formelles est souvent perçue comme une collection de notations et d'outils difficiles à utiliser et à généraliser. En effet, l'expression de propriétés dans des formalismes spécifiques CTL*, PLTL,…(langages de logique temporelle) [SCH99] requiert une maîtrise par les praticiens du model-checking qui n'existe pas actuellement dans l'industrie. Toutefois l'évolution rapide de cette spécialité ces dernières années a vu naître de nouveau outils, exploités avec succès dans le domaine de l'aéronautique et du ferroviaire : méthode B, Esterel, Statemate, Lustre, la liste est assez longue. La référence [OFT97] fournit quelques applications des méthodes formelles dans de nombreux domaines. L'industrie s'intéresse d'autant plus à ces outils qu'ils bénéficient d'améliorations continues concernant la formulation des propriétés, l'interface homme-machine, l'intégration dans des langages de spécification courants… Citons comme exemple l'outil Safety Checker Blocket développé par la société TNI-Valiosys et commercialisé depuis peu. Cet outil se présente sous la forme d'une toolbox intégrée dans l'environnement Simulink/Stateflow et permet de vérifier des propriétés de modèles exprimées à l'aide de blocks Simulink/Stateflow. Le model-checker explore le graphe des états du système et vérifie si la propriété est vraie. Dans le cas contraire, celui-ci fournit un scénario (ou contre-exemple), c'est-à-dire la succession ordonnée des entrées du modèle qui amène le système dans l'état identifié. L'arrivée de ce type d'outil sur le marché est très prometteuse pour voir se populariser l'application des méthodes formelles dans les secteurs industriels encore réticents.
Enfin, évoquons les coûts engendrés par l'introduction du formel dans le développement. L'adoption de ces techniques requiert l'affectation précoce de ressources aux projets, pratique qui s'oppose à l'habitude industrielle selon laquelle, dans la plupart des projets, l'essentiel des ressources (effort et coûts) est attribué vers la fin du développement (test et mise au point).

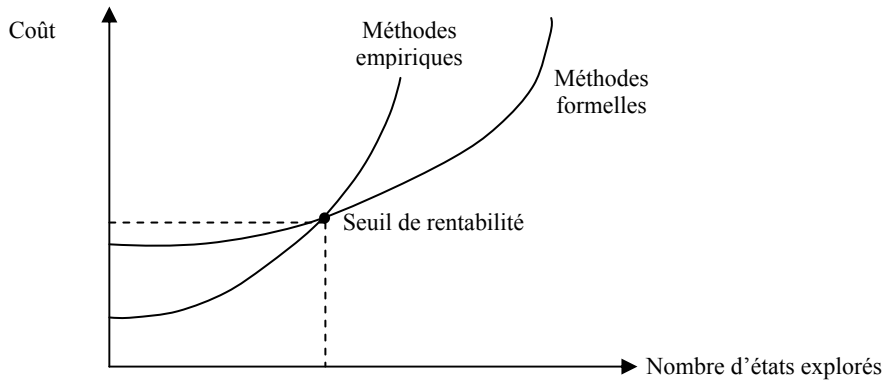


Figure 10 : Comparaison des coûts

Le coût d'introduction des méthodes formelles est généralement élevé, car elles requièrent dans le cas général des ingénieurs expérimentés, bien formés à la méthode, d'où une ordonnée à l'origine élevée pour la courbe des méthodes formelles (figure 10), alors que le coût des méthodes empiriques (tests, simulation) est plus faible. Néanmoins, les coûts de formation peuvent s'amortir sur plusieurs projets si les méthodes formelles sont utilisées de façon continue.

Concluons cette partie en soulignant l'intérêt des méthodes formelles. C'est un moyen efficace pour éviter d'introduire des erreurs de conception, pouvant coûter cher si elles sont découvertes tardivement. De plus, le bénéfice apporté est réel pour la plupart des parties critiques du point de vue de la fiabilité d'un système embarqué. Une spécification formelle et/ou l'utilisation de règles rigoureuses de construction peuvent être mieux appropriées pour le reste du système.

3.3.3. Conditions d'applicabilité d'une méthodologie de conception sûre

Il s'agit de définir les méthodes appropriées à utiliser, et aussi la manière de les appliquer dans un cadre donné : spécificité des systèmes, méthode de travail, culture d'entreprise. Ces méthodes doivent contribuer à la qualité de la conception tant au niveau fonctionnel (méthode de vérification, validation), que sur le plan de la sûreté de fonctionnement (choix d'une modélisation dysfonctionnelle « comportementale », calcul des attributs, renforcement de la fiabilité, …).
Néanmoins, comme il a été souligné au début de ce chapitre, les méthodes classiques de conception ne doivent pas être brutalement bouleversées par l'introduction d'outils radicalement différents de ceux habituellement manipulés dans une équipe projet. Cette constatation est particulièrement vraie dans le cas des méthodes formelles qui ont pendant longtemps connu une certaine opposition dans le milieu industriel : bouleversement des acquis et des modes de travail, complexité des techniques, manque de formation des utilisateurs. Concernant la sûreté de fonctionnement, celle-ci est souvent apparue comme une démarche trop lourde et trop complexe vis-à-vis des systèmes développés. En particulier, les méthodes analytiques de type graphe de Markov ont été longtemps considérées comme trop complexes à mettre en œuvre ou trop abstraites dans leur principe face à la simplicité des méthodes empiriques comme la simulation de Monte-Carlo. Toutes ces méthodes paraissaient s'opposer aux méthodes conventionnelles de conception. Néanmoins, cet état des lieux est à relativiser. En effet, les bouleversements sont surtout dus à l'évolution des besoins, à la complexité et à la taille croissante des systèmes développés.
L'appréhension dans l'utilisation des méthodes formelles et analytiques résultait également du manque de maturité des outils et de leur dissociation par rapport aux outils de spécification et de modélisation habituellement utilisés. La mise en œuvre de méthodes formelles et de sûreté de fonctionnement impliquait une charge de travail supplémentaire pour l'ingénieur de par la multiplication des différents modèles associés. De cette constatation est né il y a quelques années un certain nombre d'ateliers logiciels [SIG96]. On peut citer par exemple Statemate développé par la société I-Logic et utilisé à PSA pour valider dynamiquement le comportement de certaines parties critiques d'un système. La particularité de cet outil tient à son modèle dynamique qui repose sur le formalisme des *Statecharts*. Ce formalisme à états/transitions, d'un haut niveau d'expression, décrit de façon détaillée le comportement d'un système. Le caractère formel de cet outil résulte de la possibilité d'explorer le graphe des états du système modélisé pour vérifier certaines propriétés.
Concernant les outils de sûreté de fonctionnement classiquement utilisés comme les arbres de défaillances, il convient de souligner l'importance de l'aptitude à la synthèse qui est requise pour construire un modèle. En effet, l'ingénieur chargé de l'étude doit acquérir une connaissance en profondeur du fonctionnement de son système dans tous les modes (nominaux, dégradés), et être capable d'identifier tous les scénarios (coupes minimales) aboutissant aux événements redoutés. Face à la complexité croissante des modèles dans le milieu industriel, plusieurs ateliers logiciels sont nés. Le principe général consiste à représenter le comportement (dys)fonctionnel du système dans un langage de plus haut niveau que les arbres de défaillance. En effet, il apparaît plus naturel pour un ingénieur de formaliser un comportement « fonctionnel » plutôt que l'inverse. A partir de cette représentation, ces ateliers permettent de générer automatiquement les arbres de défaillance correspondant. Citons comme exemple SIMFIA de la société Sofreten, actuellement utilisé à PSA Peugeot Citroën, et OCAS de la société Dassault Aviation et distribué par GFI. Ces deux ateliers seront détaillés dans le chapitre suivant.

Ainsi, la définition d'une méthodologie permettant de concevoir des produits complexes prend une importance significative, en particulier si celle-ci s'adapte aisément aux habitudes du concepteur. L'intérêt majeur réside dans la possibilité d'intégrer facilement des outils améliorant la qualité et la sûreté, en rendant les principes théoriques de ces outils les plus transparents possible aux yeux du concepteur.

Face à l'ensemble des difficultés pour intégrer dans un même processus de développement les activités de conception et de sûreté de fonctionnement, et dans l'objectif d'améliorer continûment la qualité de ses produits, PSA poursuit depuis de nombreuses années des travaux de recherche dans le cadre de thèses CIFRE en collaboration avec des laboratoires universitaires.

3.3.4. Thèse de Didier Jampi [JAM01c]

Les travaux présentés dans ce mémoire font suite à ceux de Didier Jampi. Sa thèse [JAM01c], réalisée dans le cadre d'une convention CIFRE entre le CRAN et PSA Peugeot Citroën et soutenue en 2001, propose une méthodologie de conception des systèmes de contrôle-commande basée sur le formalisme des réseaux de Petri interprétés.
Le formalisme des réseaux de Petri interprétés de commande s'avère tout à fait adapté pour modéliser le comportement des systèmes de contrôle-commande de par sa nature discrète et a fait l'objet de plusieurs travaux antérieurs au CRAN [AUB87], [ZAN95].
La méthodologie de conception décrite dans cette thèse exploite le caractère formel des réseaux de Petri pour s'assurer que les fonctions spécifiées possèdent un certain nombre de propriétés génériques (vivacité, réinitiabilité, …). Contrairement à l'approche classique qui consiste à vérifier ces propriétés a posteriori, la démarche proposée est fondée sur une construction modulaire par raffinement à base de *primitives de constructions* ; ces primitives sont au nombre de quatre : *Embryon, Séquence, Alternative et Répétitive* [JAM01a], [JAM01b], [JAM01c]. Il est montré que toute fonction construite à l'aide de ces primitives est vivante et, réciproquement, toute fonction vivante est décomposable en primitives élémentaires.
Précisons néanmoins que le caractère vivant se limite à la structure autonome des réseaux de Petri modélisés. L'ajout de l'interprétation, c'est-à-dire d'un couple {événement/condition} sur les transitions ne permet plus de garantir l'absence de blocage ou d'interblocage du système complet. Cependant, la vivacité structurelle est une condition nécessaire de la vivacité « comportementale » du système.
Un autre aspect méthodologique développé concerne l'intégration des analyses de sûreté de fonctionnement dans la conception. Tout d'abord sur le plan qualitatif : la méthode proposée s'inspire de l'AMDEC. Une réflexion inductive, basée sur la structure du réseau de Petri, consiste à rechercher les causes potentielles d'un dysfonctionnement des fonctions. En effet, d'un point de vue de la modélisation, un dysfonctionnement se manifeste par une évolution anormale du marquage du réseau. Sachant qu'aux places sont associées des actions (élaboration de consignes actionneurs, …) et aux transitions un couple {événement/condition} (lecture d'informations capteurs,…), les causes matérielles et/ou logicielles sont répertoriées pour chaque nœud du réseau. L'étude qualitative repose donc sur la construction d'un tableau dans lequel tous les nœuds sont répertoriés et analysés : causes et conséquences d'un franchissement prématuré ou d'un blocage d'une transition, erreur d'évaluation des actions associées aux places, …
Ensuite, une évaluation quantitative de la sûreté de fonctionnement est mise en place à partir de la représentation précédente, afin de valider la pertinence de certaines stratégies de

reconfigurations (logicielles, matérielles) issues de recommandations émises suites à l'analyse qualitative. Cette étude est basée sur l'utilisation des graphes de Markov.

4. Contributions des travaux

Les travaux de thèse présentés dans ce manuscrit s'inscrivent dans la lignée des travaux précédents, à savoir définir une méthodologie de conception des systèmes mécatroniques intégrant la sûreté de fonctionnement.
L'expérience acquise dans le milieu industriel montre qu'il n'est pas envisageable de bouleverser radicalement les habitudes de conception mais, par contre, que les méthodes actuellement utilisées peuvent être facilement améliorées en apportant des éléments complémentaires. L'objectif d'une méthodologie est d'apporter une garantie sur la qualité des systèmes développés, tout en considérant les contraintes de coût et de temps.

Il est manifeste que la complexité croissante et la spécificité des systèmes embarqués justifieront de plus en plus le besoin de structurer l'ensemble des activités de conception. Vérification, validation, spécification comportementale et bien entendu sûreté de fonctionnement font partie intégrante d'un même processus.

Suite aux travaux précédents, nous avons tout d'abord souhaité améliorer la méthodologie basée sur les réseaux de Petri et apporter quelques compléments indispensables. L'étude des méthodes formelles a démontré un intérêt certain dans la phase de vérification/validation : la capacité d'élimination d'un maximum d'erreurs permet d'accélérer les phases de tests ; de plus, les méthodes formelles sont applicables à chaque étape du raffinement d'un modèle, évitant ainsi l'introduction de fautes au cours de la conception. Enfin précisons que cette approche répond au problème de vérification de la vivacité des fonctions modélisées en réseaux de Petri après ajout de l'interprétation ou, plus généralement, après l'intégration de la composante continue.
Un travail important a été réalisé sur l'analyse quantitative. En effet, comme il a été souligné, il semble important de disposer d'une modélisation dysfonctionnelle du système afin d'améliorer la compréhension des mécanismes de défaillance et de permettre des échanges entre les différents protagonistes participant à un projet véhicule. Cette modélisation est également nécessaire en vue d'une capitalisation des expériences réalisées tout au long des différents projets. Nous nous sommes donc orientés vers une méthode graphique et analytique : les graphes de Markov. De plus, la construction d'un tel modèle est facilitée de par son analogie avec les réseaux de Petri. Néanmoins, il est reconnu que la démarche traditionnelle souffre de certaines limitations liées à la génération de l'explosion combinatoire du nombre d'états et d'hypothèses restrictives. Les méthodes de simplification trouvées dans la littérature supposent une construction initiale exhaustive du graphe, la réduction étant opérée a posteriori.
Il a également été souhaitable de développer une approche quantitative apportant une solution à la problématique de la fiabilité dynamique.
Le principe de l'analyse markovienne développée repose sur la construction d'un graphe de Markov réduit en agrégeant les états ayant une interprétation physique commune : les modes de fonctionnement. Le graphe agrégé est composé d'un nombre de « macro-états » égal au nombre de modes de fonctionnement. Les taux de transition sont alors évalués en tenant compte du processus d'agrégation. On montre qu'ils dépendent notamment de la dynamique du système, illustrant l'existence du couplage entre la composante stochastique et la

composante déterministe (hybride), aspect qui a été relevé précédemment dans le paragraphe de la fiabilité dynamique. En reprenant le schéma déjà évoqué :

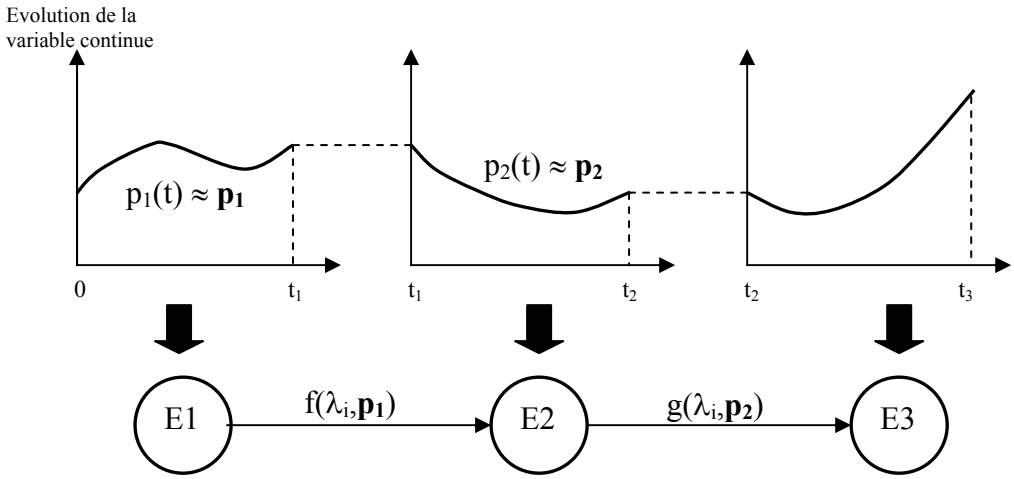


Figure 11 : Principe général de la représentation par un graphe de Markov agrégé

Si pour chaque configuration, il est possible de synthétiser l'évolution dynamique des variables continues en une simple variable statique puis de l'intégrer dans le modèle markovien, alors le calcul des paramètres de sûreté de fonctionnement tiendra compte des interactions précédemment définies.
Nous montrons comment il est possible, et sous quelles conditions, d'opérer une telle réduction.

Un autre aspect développé est la possibilité de générer un graphe de Markov réduit sans passer par la construction d'un graphe exhaustif. Une bonne connaissance des modes de fonctionnement et du comportement du système est suffisante.

Enfin, précisons que la méthodologie est généralisable à d'autres formalismes que les réseaux de Petri, plus familiers aux yeux du concepteur, en particulier les Statecharts. Cette généralisation est indispensable pour pérenniser la méthodologie.

Le fil directeur repose sur une modélisation de haut-niveau, riche en détails, simulable, puis sur une déclinaison de cette modélisation en des modèles plus abstraits permettant d'appliquer des outils de vérification formelle existants, comme le Model-Checking et, bien entendu, la méthode markovienne. Celle-ci s'apparente également à une abstraction du modèle initial.

Dans le chapitre 2, nous rappelons quelques concepts sur les systèmes de contrôle-commande, et le caractère hybride de ces systèmes. Nous faisons quelques rappels sur les formalismes répondant à nos besoins de modélisation des systèmes dynamiques hybrides. La technique de model-checking est également évoquée, car il apparaît que de nombreux outils commencent à voir le jour et sont exploitables industriellement. Nous finissons ce chapitre par un rappel sur la simulation de Monte-Carlo et les processus de Markov.
Le chapitre 3 sera consacré à la méthode de réduction des graphes de Markov précédemment évoquée ; une analyse des performances de cette méthode est effectuée.
Finalement nous appliquons cette méthode de simplification à un cas test. Nous montrons qu'une analyse markovienne est possible en exploitant des modélisations différentes. Une comparaison est réalisée par rapport à la simulation de Monte-Carlo.

CHAPITRE 2

MODELISATION ET ANALYSE DES SYSTEMES DE CONTROLE-COMMANDE

Chapitre 2

Modélisation et Analyse des systèmes de contrôle-commande

1. Introduction

Modéliser, c'est chercher à avoir une vue *suffisamment abstraite* d'un système physique, pour pouvoir le représenter de façon compréhensible par l'homme.
En fonction des objectifs recherchés, le niveau d'abstraction sera différent. En effet, il est important ou cours d'une modélisation de pouvoir s'exprimer à grand niveau de généralité en oubliant les détails non significatifs ou, au contraire, être capable d'enrichir suffisamment la description pour représenter des comportements très précis.
Le choix d'un formalisme de spécification et de conception intégré dans une méthodologie dépendra de plusieurs critères :

- le niveau d'abstraction dans le mode de représentation,
- la capacité d'adaptation aux spécificités des systèmes étudiés,
- le nombre et la nature des méthodes d'analyse (dys)fonctionnelle exploitant ce formalisme.

Comme nous l'avons vu précédemment, les systèmes mécatroniques sont constitués d'éléments matériels et logiciels et comprennent à la fois des aspects continus (variables physiques et variables de la partie opérative) et des aspects discrets (variables échantillonnées du système de contrôle-commande, occurrence de défaillance, …).
Le système est en relation permanente avec l'environnement de façon à en acquérir des informations et à interagir sur lui, selon des délais cohérents avec la dynamique du processus physique à commander. Ces systèmes sont dits *réactifs* et sont également soumis à des contraintes *temps-réels*.

Le choix d'un formalisme de modélisation va donc reposer sur leur capacité à intégrer tous ces aspects.

Néanmoins, l'exploitation d'outils d'analyse et la mise en œuvre des activités de sûreté de fonctionnement requièrent des niveaux d'abstraction ou de raffinement différents. Par exemple, la construction d'un arbre de défaillance ou d'un graphe de Markov pour ne citer qu'eux, fournit une représentation du système se restreignant à des scénarios de défaillance conduisant à un état redouté.
Il est donc nécessaire d'avoir recours à des descriptions différentes pour chaque activité :

- **de nature déterministe** pour une représentation comportementale fonctionnelle du système. Celui-ci peut ensuite être éventuellement décliné en plusieurs modèles plus ou moins abstraits, par exemple, la mesure de performances temporelles par des simulations (validation dynamique du système), ou la vérification/validation par des méthodes formelles comme le model-checking exploiteront la modélisation différemment (cf. figure 12),
- **de nature probabiliste** pour une représentation des phénomènes stochastiques liés aux défaillances et pour mesurer les différents indicateurs de sûreté de fonctionnement.

Cependant, pour l'applicabilité industrielle d'une méthodologie, il est important que chaque analyse n'implique pas une re-formalisation du système complet dans le langage associé, pour des raisons évidentes de coût, de temps, et d'erreurs de traduction d'un modèle à l'autre. Le fil conducteur de la méthodologie de conception consiste à opter pour un formalisme de modélisation comportementale, compatible d'une part, avec les spécificités des systèmes de contrôle-commande et, d'autre part, avec l'ensemble des analyses précitées. A partir d'une modélisation initiale de haut niveau, autorisant une description raffinée des différentes tâches à réaliser par le système, plusieurs modélisations plus abstraites sont déduites pour les analyses effectuées en parallèle de la modélisation. En fonction des outils existant sur le marché, ces déclinaisons peuvent être réalisées automatiquement ou, le cas échéant, manuellement par élimination ou agrégation des détails superflus.
La figure suivante résume ce concept :

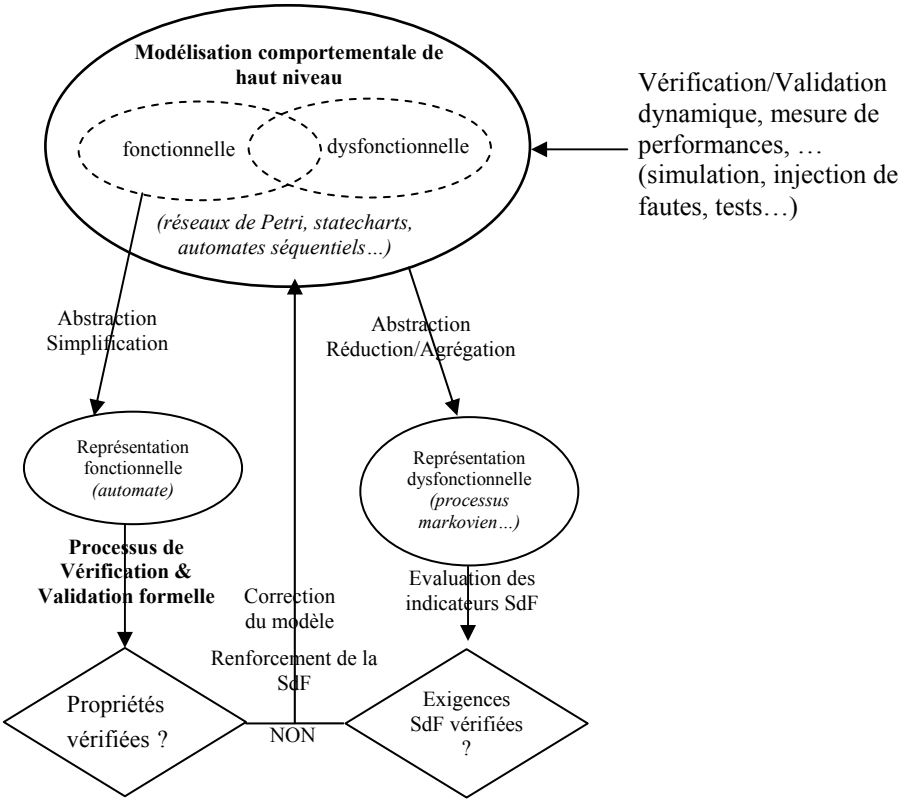


Figure 12 : Traitement des différentes modélisations

Il apparaît que les formalismes basés sur les automates à états finis correspondent le mieux aux besoins du secteur automobile. D'une part, leur nature discrète est en adéquation avec le

caractère discret des systèmes de contrôle-commande ; d'autre part, cette représentation, couplée à un moteur de preuve et une spécification de propriété, permet d'exploiter les méthodes formelles telles que le model-checking. Enfin, la représentation du comportement sous forme d'automates facilite la construction du graphe de Markov sous-jacent.
Après avoir rappelé les principales caractéristiques des systèmes de contrôle-commande, nous allons faire un état des lieux sur les formalismes de modélisation « fonctionnelle » jugés intéressants comme base d'une méthodologie de conception. Les formalismes retenus reposent sur une description sous forme d'automates à états finis. Nous examinons également quelques ateliers logiciels existant permettant à la fois de modéliser les aspects fonctionnel et dysfonctionnel et de supporter des analyses de sûreté de fonctionnement. Nous finissons enfin ce chapitre par un rappel de la technique de model-checking ainsi que les techniques d'évaluation de la sûreté de fonctionnement permettant de prendre en compte des aspects dynamiques de la fiabilité : la simulation de Monte-Carlo et l'approche markovienne.

2. Les systèmes de contrôle-commande

Les applications actuelles des systèmes de contrôle-commande deviennent de plus en plus complexes, en raison du nombre de fonctions à assurer et à spécifier et des multiples interactions entre les différents systèmes embarqués. Les principales missions de ces systèmes, conçus dans la technologie informatique, sont [AUB91] :

- assurer la commande du processus physique,
- assurer la performance de la commande vis-à-vis des exigences formulées,
- assurer l'aide au diagnostic des défaillances du système,
- assurer les stratégies de tolérance aux fautes implémentées dans l'algorithme de contrôle-commande.

A titre informatif, précisons que ce dernier point représente une source d'erreur assez importante dans le processus de conception de l'algorithme en raison de la complexité induite par la multiplication des modes de fonctionnement [ARL95].

Les caractéristiques de ces systèmes sont bien connues [MOI91] :

- ils mélangent des phénomènes continus et des événements discrets (systèmes hybrides),
- ils possèdent plusieurs modes de fonctionnement,
- ils ont un nombre de variables limité,
- ils nécessitent des fréquences d'échantillonnage élevées,
- ils nécessitent un retard faible entre la détection d'un événement et l'action associée,
- ils doivent assurer une grande variété de missions : régulation, gestion d'événements, sûreté de fonctionnement, diagnostic, …

Ces systèmes sont qualifiés de *réactifs*, c'est-à-dire qu'ils réagissent de façon permanente aux sollicitations de leur environnement. Le système réactif contrôle le comportement de l'environnement en observant son évolution dans le temps par des capteurs et en pilotant au moyen d'actionneurs (cf. figure 13). En général, ces systèmes sont soumis à des contraintes temporelles fortes et doivent répondre dans des délais très stricts à des stimuli de l'environnement. La dynamique est imposée par le processus commandé. Ces systèmes sont alors qualifiés de *temps réel*. [PAL98] fournit une définition d'un système temps réel : « *c'est un système qui doit satisfaire des contraintes de temps de réponse explicites et bornées. Si les*

bornes temporelles ne sont pas respectés, des dégradations des performances et des mauvais fonctionnements apparaissent ».

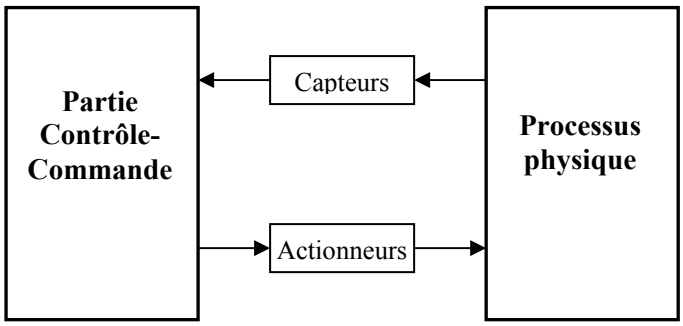


Figure 13 : Interactions Système de Contrôle-Commande/Processus physique

De plus, les systèmes de contrôle-commande peuvent être composés de différentes parties actives opérant au même moment. Ce sont des systèmes à composantes *parallèles*. Cette caractéristique engendre des complications qui n'existent pas dans les systèmes où une seule tâche s'exécute. Par exemple, il est nécessaire de résoudre le problème de l'accès concurrent à des ressources communes.

Enfin, comme nous l'avons déjà évoqué, la conception des systèmes de contrôle-commande nécessite de prendre en compte les interactions entre les phénomènes discrets et continus. On décompose habituellement le système hybride en plusieurs sous-systèmes discrets et continus comme illustré sur la figure 14 [AUB87], [ZAN95]. La *partie opérative* contient l'ensemble des variables continues représentant le processus physique à commander (régulation en vitesse, pression, débit, …). La *partie commande* représente l'automatisme à concevoir et élabore l'ensemble des actions/opérations en fonction de l'état de la partie opérative (élaboration des consignes actionneurs, gestion des modes de fonctionnement, des reconfigurations,…). Lorsque cette partie commande est complexe, celle-ci peut se décomposer en une partie commande (PC) et opérative (PO), cette dernière contenant l'ensemble des variables et des opérateurs servant à effectuer les calculs.

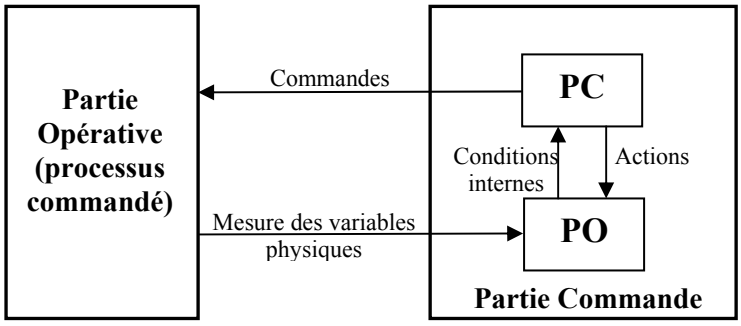


Figure 14 : Décomposition du système en PC et PO

Au regard de la spécificité et de la complexité des systèmes de contrôle-commande, un formalisme de modélisation doit être judicieusement choisi pour représenter le comportement du système par rapport à la dynamique de l'environnement. Celui-ci doit être suffisamment expressif pour permettre une analyse de performance tant au niveau fonctionnel

(performances temporelles, gestion des modes de fonctionnement) qu'au niveau dysfonctionnel (possibilité de représentation de défaillance). Un critère d'importance porte également sur la possibilité de mettre en œuvre des analyses formelles à des fins de vérification/validation, quitte à adapter le modèle en conséquence par un conditionnement de celui-ci et de supporter les techniques de simulation hybride, approche complémentaire des techniques formelles. Nous allons à présent passer en revue quelques formalismes de modélisation répondant à nos besoins.

3. Formalismes de modélisation

3.1. Préambule

La méthodologie proposée nécessite de disposer le plus en amont possible d'un modèle complet du système à développer afin d'être capable de maîtriser au plus tôt l'ensemble des erreurs potentielles de conception et de mettre en œuvre les évaluations et les stratégies de renforcement de la sûreté de fonctionnement.
Il est nécessaire de modéliser :

- le système lui-même (PO+PC) dans les modes nominaux, dégradés et au cours des transitions de reconfiguration,
- son environnement (générant les entrées du système, utilisant les sorties et produisant les événements externes agissant sur le système),
- les défaillances d'éléments devant être prises en compte dans les analyses de sûreté de fonctionnement.

De nombreux formalismes sont disponibles actuellement, que nous pouvons classer selon trois critères :

- Approche de type « états-transitions » (Statecharts, Esterel, Réseaux de Petri, …) ou de type « logique » (logique temporelle, …) ,
- Approche synchrone ou asynchrone,
- Approche textuelle ou graphique.

Concernant le premier point, précisons que les systèmes de transitions se focalisent sur la description de la dynamique du système à commander tandis que les approches logiques insistent plus une description statique et abstraite du système.
Le deuxième point caractérise les formalismes selon leur façon de prendre en compte le temps et les événements.

Nous privilégions le formalisme de type état-transition, car il apparaît comme étant le mieux adapté à la modélisation de systèmes de contrôle-commande et à la mise en œuvre des différentes analyses précédemment évoquées.

3.2. Modélisation hybride

Les systèmes considérés nécessitent de prendre en compte à la fois l'aspect discret et continu. Or, quand les deux aspects sont étroitement liés et que les degrés d'intégration et de complexité sont élevés, la modélisation du système par une seule de ces composantes n'est

pas possible ; il est difficilement envisageable de se limiter à une représentation intégralement discrète ou entièrement continue.
De nombreux travaux ont fait l'objet du problème de la coopération des deux composantes au sein d'un même modèle. La modélisation de l'aspect hybride est alors abordée selon deux tendances [AND96] : d'une part, la tendance reposant sur l'intégration des aspects continu et discret au sein d'un même formalisme, et d'autre part, celle associant un formalisme différent pour chaque aspect.

3.2.1. L'approche intégrée

Dans la démarche d'intégration des deux aspects au sein d'un même modèle, deux approches intégrées sont possibles. Elles dépendent du modèle dominant, c'est-à-dire du modèle à partir duquel est effectuée l'extension : on distingue *l'approche continue* et *l'approche discrète*.

- **l'approche continue** : celle-ci consiste à intégrer l'aspect discret au sein de la représentation continue. Le comportement du modèle continu est étudié en présence de discontinuités : non-linéarité (seuil, saturation, relais, …), commutation, etc…
- **l'approche discrète** : l'aspect continu est intégré au sein d'un modèle à événements discrets comme les réseaux de Petri. [CHA98b] propose un exemple basé sur les réseaux de Petri. Les variables continues du processus physique à commander sont discrétisées et représentées par un certain nombre de jetons dans des places. L'évolution des variables se traduit par l'apparition ou la disparition de jetons dans ces places. Le couplage entre l'aspect discret et l'aspect continu est réalisé à l'aide de transition commune entre places « continues » et places « discrètes ». Cet exemple sera notamment repris ultérieurement dans le chapitre 4 et est détaillé en annexe 4.

3.2.2. L'approche séparée

Celle-ci consiste à séparer l'aspect discret et l'aspect continu. On distingue également deux types d'approche :

- **Le « moniteur » de systèmes d'équations** : dans un certain sens, le modèle à événements discrets peut être considéré comme un moniteur des équations formant le modèle continu. A un instant donné, seul l'un des modèles est actif. Soit il y a modification de la structure des équations formant le modèle continu et le temps n'évolue pas, soit le temps évolue mais la structure des équations ne change pas. Certains automates hybrides se basent sur cette approche : ils associent un automate classique et un ensemble de variables continues dont les conditions portant sur leurs valeurs sont exprimées sur les transitions de l'automate. L'évolution dynamique des variables continues est modélisée par des ensembles d'équations différentielles associés aux états discrets de l'automate.
- **L'approche superviseur** : dans cette approche, deux blocs différents correspondant aux aspects discret et continu sont couplés par une interface. Chaque modèle décrit un sous-système, qui peut éventuellement évoluer de façon autonome. Il ne s'agit pas de deux vues complémentaires d'un même système, mais plutôt de deux sous-systèmes représentés par deux modèles. L'interface traduit les signaux échangés entre les deux parties. Les variables de la partie continue évoluent de façon autonome dans le temps. Le système superviseur, représentant un système à événement discret, regarde

l'évolution de ces variables continues en s'intéressant uniquement aux changements d'états discrets. Quand un état spécifique est observé, une action correspondante du système superviseur est lancée.

3.3. Les réseaux de Petri

3.3.1. Généralités

Les réseaux de Petri sont basés sur la théorie des automates. Leur intérêt est qu'ils permettent d'exprimer de manière aisée les mécanismes de parallélisme et de synchronisation. Ils peuvent être utilisés à la fois [DAV89] :

- pour des analyses de type qualitatif (vérification de propriétés),
- pour des analyses de type quantitatif, comme l'évaluation de performances fonctionnelles ou de sûreté de fonctionnement.

Au niveau théorique, les réseaux de Petri sont plus expressifs que les machines à état qui en constituent une sous-classe.

Les réseaux de Petri sont utilisés pour modéliser des systèmes évoluant dans le temps. Ces évolutions sont mises en évidence grâce aux marques. L'ensemble des marques forme le marquage (cf. annexe 2). Le marquage définit à un instant donné l'état du système. L'ensemble des transitions représente quant à lui l'ensemble des événements dont les occurrences provoquent des modifications de l'état du système. On peut par exemple associer aux transitions un couple <événement/action>. Cette vision des choses est typiquement celle des automates à états, puisque l'action est effectuée sur la transition et fait passer dans ce cas le système d'un état à un autre. Cependant, cette sémantique n'est pas fixe et il est aussi possible de considérer la représentation plus proche du temps réel dans laquelle les actions sont associées aux places et les conditions sont associées aux transitions. Dans le cas des systèmes de contrôle-commande, la catégorie des réseaux de Petri qui semble la plus appropriée à la modélisation est la classe des *réseaux de Petri interprétés de commande* [AUB87], [JAM01c], [DAV89].

Tout comme les automates, les seules propriétés qu'il est possible d'obtenir sur les réseaux de Petri « autonomes » sont les caractéristiques fonctionnelles. Ces réseaux ont été initialement conçus pour modéliser des systèmes parallèles. Des extensions ont été développées afin d'insérer le temps pour modéliser les systèmes temps réel [MES94]. Ces extensions permettent de prendre en compte les durées de « passage » d'un état à autre (temporisation) et l'influence de l'environnement sur le système (synchronisation). Pour ces extensions, certaines propriétés fonctionnelles ne peuvent plus être appliquées.

On distingue deux types de propriétés fonctionnelles :

- **Les propriétés structurelles** : elles sont basées sur la théorie des graphes car le marquage n'est pas pris en compte. Ces propriétés ne font pas intervenir le temps, mais uniquement la structure définie par l'ensemble des transitions et des places.
- **Les propriétés issues de l'analyse des évolutions possibles** : ces propriétés font intervenir le marquage du réseau. Les propriétés sont vérifiées en analysant le graphe d'accessibilité. Par exemple, la vivacité, l'existence de blocage, de conflits, la bornitude, … en sont quelques exemples. Ces propriétés expriment typiquement le fait qu'un marquage donné puisse ou non être atteint.

Certaines techniques d'analyse utilisent l'algèbre linéaire pour rechercher des invariants dans l'évolution du réseau [DAV89]. Chaque marquage peut être représenté par un vecteur et le réseau peut être modélisé comme un ensemble d'équations algébriques.

Néanmoins, la plupart des propriétés ne sont pas vérifiables pour les extensions de réseaux de Petri (synchronisés ou interprétés), notamment lorsque l'évolution du réseau dépend du franchissement de transition sur occurrence d'événements issus d'une partie opérative modélisée par exemple dans un autre formalisme plus adapté (par exemple dans le cas d'une modélisation hybride par approche séparée). Cependant, certaines propriétés pouvant être vérifiées sur le réseau autonome sous-jacent constituent alors des conditions nécessaires de ces propriétés [JAM01c].

Pour finir, rappelons que les réseaux de Petri sont tout à fait adaptés pour modéliser des *systèmes à événements discrets* qui correspondent par abus de langage à des modèles à événements discrets de systèmes physiques complexes. Dans ces modèles, le temps est représenté par une suite d'événements, a priori non répartis également. Un événement est un instant où au moins une variable passe d'une valeur discrète à une autre. Ainsi, le franchissement d'une transition n'est plus directement fonction du temps mais dépend de l'occurrence asynchrone d'événements [CHAM99].

L'intégration du formalisme des réseaux de Petri dans la méthodologie de conception proposée implique de choisir la classe des réseaux la plus en adéquation avec les systèmes de contrôle-commande. Nous allons donc à présent rappeler brièvement la classe des *réseaux de Petri interprétés de commande*.

3.3.2. Les réseaux de Petri interprétés de commande

Les réseaux de Petri interprétés de commande sont adaptés à la modélisation des systèmes de contrôle-commande. Ils permettent de définir un automate de commande interfacé et synchronisé par rapport à l'environnement. L'évolution de cet automate dépend de l'occurrence d'événements internes à la commande ou issus du processus physique à commander.
La figure 15 [DAV89], [JAM01c] illustre les échanges d'informations entre la commande et le processus commandé :

Un réseau de Petri interprété de commande possède les caractéristiques suivantes :

- Il est synchronisé
- Il comprend une partie opérative dont l'état est défini par un ensemble de variables $V=\{V_1, V_2, \ldots\}$. Cet état est modifié par des opérations $O=\{O_1, O_2,\ldots\}$ qui sont associées aux places. Il détermine la valeur des conditions $C=\{C_1, C_2,\ldots\}$ qui sont associées aux transitions.
- Il est sauf, c'est-à-dire que chaque place contient au plus une marque,
- Il est non temporisé (contrairement aux réseaux de Petri interprétés),
- Il est déterministe
- Ses possibilités d'entrées et de sorties sont étendues par rapport aux réseaux de Petri interprétés. En d'autres termes, l'évolution du réseau peut être directement dépendante de la dynamique du processus physique par l'intermédiaire d'événements (externes) et de conditions sur les variables continues.

En fonction de l'état et l'évolution du processus physique (définis par les conditions C_j et l'occurrence d'événements E_j), la commande agit sur celui-ci par l'intermédiaire d'actionneurs en élaborant des actions A_i ou des opérations O_i adéquates (élaboration de consignes actionneurs, …).

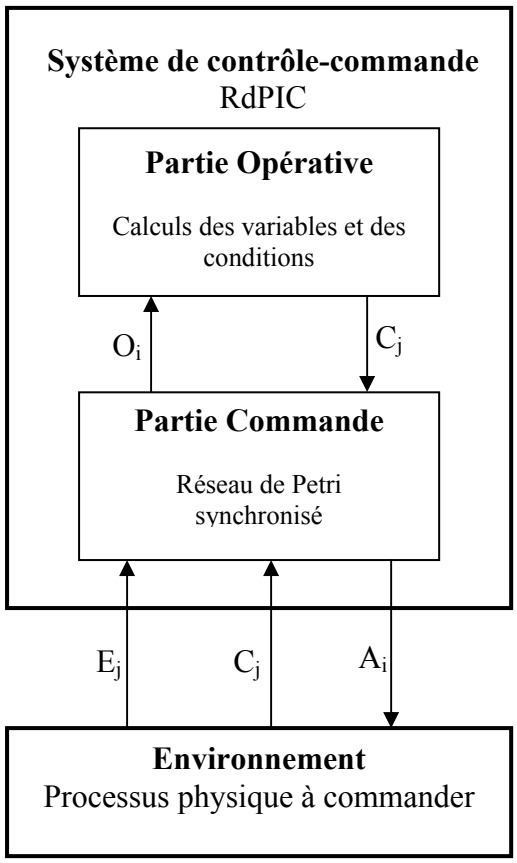


Figure 15 : Réseau de Petri Interprété de Commande

Ainsi, le franchissement d'une transition T_j s'effectuera (cf. figure 16) :

- si la transition T_j est validée,
- si la condition C_j est vraie,
- quand l'événement E_j se produira.

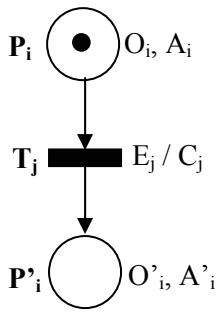


Figure 16 : Association des actions/opérations/conditions/événements aux places/transitions

Ce formalisme graphique se prête bien à la modélisation des stratégies de reconfiguration logicielle et assure une bonne compréhension globale du comportement en présence de fautes. Sur occurrence d'une défaillance prise en compte dans les mécanismes de tolérance aux fautes, un événement ou une condition particulièr(e) généré(e) par l'environnement ou la partie opérative inhibe le fonctionnement en mode nominal et reconfigure le système en mode dégradé (cf. figure 17).

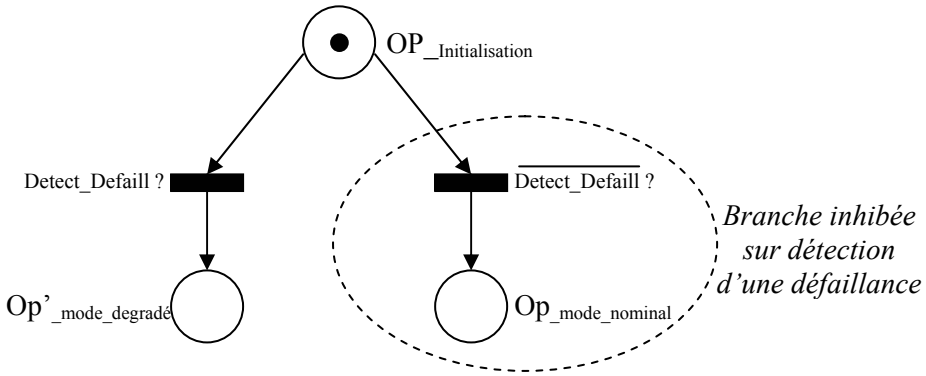


Figure 17 : Modélisation des stratégies de reconfiguration

Un intérêt méthodologique supplémentaire des réseaux de Petri est la possibilité d'ajouter à ce modèle des transitions stochastiques pour analyser, par exemple, le système en présence de fautes par une simulation classique. Le formalisme des réseaux de Petri stochastique est plus largement décrit en annexe 2.

Le chapitre suivant propose à titre d'information un exemple élémentaire d'une architecture fonctionnelle couplée à une architecture matérielle sur lequel ont été ajoutées des transitions stochastiques permettant de représenter les défaillances des composants matériels de l'architecture (calculateurs, capteurs).

Nous allons à présent voir quelques outils de modélisation et d'analyse des réseaux de Petri disponibles sur le marché.

3.3.3. Quelques outils supportant le formalisme des réseaux de Petri

Nous présentons quelques exemples d'outils que nous avons pu évaluer.

- **MOCA-RP**:
  Le logiciel MOCA-RP développé par la société Total et distribué antérieurement par GFI Consulting est destiné aux traitements probabilistes des systèmes dynamiques pour lequel il réalise des simulations de Monte-Carlo à partir de comportement décrit à l'aide de réseaux de Petri interprétés [MOCA]. Il autorise la notion d'arcs inhibiteurs et d'arcs pondérés. De plus, ce logiciel implémente la notion de messages émis et reçus par les transitions, permettant ainsi de synchroniser des sous-réseaux indépendants. La réception d'un message au niveau d'une transition permet de valider celle-ci et de faire évoluer le marquage en conséquence. Ces transitions peuvent également être temporisées à l'aide d'une trentaine de lois déterministes ou stochastiques (Exponentielle, Dirac, Weibull, Erlang,…) afin de modéliser par

exemple l'occurrence aléatoire de défaillances. La gestion des temporisations peut être définie par l'utilisateur qui a le choix entre deux politiques : avec et sans mémoire. Un certain nombre de résultats statistiques est disponible via la simulation de Monte-Carlo (cf. annexe 4).

- **Miss RdP** :
  Cet outil permet de modéliser des réseaux de Petri colorés, temporisés, stochastiques et hybrides et évaluer les performances par la simulation de Monte-Carlo. Outre les résultats obtenus en régime stationnaire, il permet également de visualiser le transitoire, utile pour les évaluations de disponibilité des systèmes non réparables et de la fiabilité. Il intègre également la notion de message sur les transitions permettant de synchroniser plusieurs réseaux entre eux.

- **SURF2** :
  Cet outil développé par le LAAS permet de construire des réseaux de Petri stochastiques à transitions exponentielles (hypothèse markovienne). L'originalité tient dans sa possibilité de générer automatiquement le graphe d'accessibilité sous-jacent et donc d'en déduire le graphe de Markov correspondant. De ce fait, les indicateurs de sûreté de fonctionnement sont évalués analytiquement en résolvant le système d'équations différentielles associé (Chapman-Kolmogorov). Néanmoins, les hypothèses restrictives permettent difficilement d'exploiter un réseau de Petri uniquement stochastique pour spécifier un système de contrôle-commande et les composantes hybrides.

3.4. Les Statecharts

3.4.1. Généralités

Le formalisme des Statecharts est un formalisme visuel créé par Harel pour la spécification des parties structurelles et comportementales des systèmes en cours de développement.
Il permet de décrire de façon hiérarchique le comportement dynamique d'un système à l'aide d'états et d'événements. Ainsi, suivant ce modèle, le système peut se trouver dans différents états et subir des transitions d'un état à un autre suite à l'apparition de certains événements. Ces événements sont diffusés et reçus instantanément, qualifiant ainsi ce modèle de synchrone. De ce fait, l'hypothèse synchrone permet de tirer un certain nombre de transitions au même instant (durées des transitoires nulles) afin d'atteindre un état atomique de la description.

Les Statecharts permettent de représenter les machines à états complexes sous des formes plus abstraites que les machines à états finis (MEF) conventionnelles. Ainsi des opérateurs de parallélisme et d'abstraction ont été définis comme, par exemple, les opérateurs XOR et AND.

Le symbolisme graphique de ce formalisme est principalement composé de [MES94] :

- **rectangles arrondis pouvant être emboîtés** : ce sont les états, éventuellement hiérarchiques. Des sous-groupes d'états d'un sous-groupe d'états sont en disjonction

(par exemple C et D du groupe B). Une telle décomposition en états est un « ou-exclusif » : le système est soit dans C, soit dans D.

- **flèches étiquetés** : elles représentent les transitions, dont les étiquettes sont de la forme suivante : « événement(condition) / actions ». Les actions sont exécutées lorsque les conditions sont vraies et sur occurrence des événements associés. Une flèche sans état d'origine pointe sur l'état par défaut.
- **connecteur H** : ceci permet de conserver l'historique des derniers états visités.
- **lignes pointillées** : elles permettent de séparer un groupe d'états en deux sous-groupes. Elles correspondent à la conjonction entre les deux. Le groupe d'états est alors le produit orthogonal des sous-groupes. Une telle décomposition en états est un « et », le système est dans deux états à la fois.

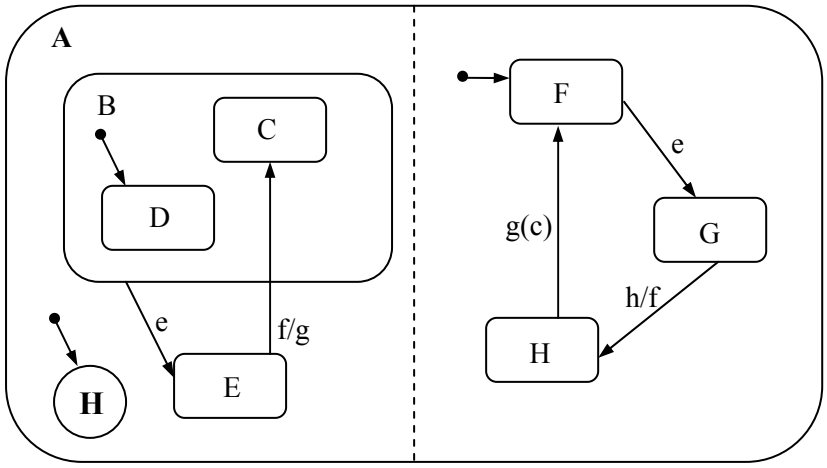


Figure 18 : Exemple de Statecharts

3.4.2. Quelques outils

Plusieurs outils supportent ce formalisme. Nous pouvons citer Statemate de la société I-Logic qui permet de décrire les aspects comportementaux, fonctionnels et structurels. Des contrôles de complétude et de cohérence sont effectués à l'aide de tests exhaustifs. La vérification de l'accessibilité d'un état, de la présence de non-déterminisme et de blocages est effectuée à l'aide de ces tests.

Un autre outil est Stateflow de la société The Mathworks qui se présente sous la forme d'une toolbox dans l'environnement Matlab / Simulink. Ceci permet notamment de partager des variables globales entre les deux environnements, facilitant la représentation hybride d'un système.

A titre de rappel [COS01], Matlab est un environnement d'évaluation de modèles numériques. Il désigne à la fois un langage et l'interprète de ce dernier. Des bibliothèques de fonctions pour plusieurs domaines en mathématiques numériques sont proposées. Matlab inclut une interface avec le langage C permettant une ouverture sur d'autres types de langages et autorise de ce fait la compilation de modèles en vue d'une implémentation sur calculateur. L'outil Simulink est une interface graphique pour la modélisation des systèmes dynamiques sous forme de schémas-blocs. Grâce à nombreux blocs de base fournis, il est possible de créer des

modèles rapidement et clairement sans écrire une seule ligne de code. A partir des modèles Simulink, des fonctions sophistiquées de simulation et d'analyse permettent d'obtenir des résultats rapides et précis : méthodes d'intégration pour systèmes à pas fixe ou variable, simulation interactive,…
Simulink est un environnement très adapté à la modélisation de comportements de phénomènes physiques. Enfin, un aspect très intéressant est la possibilité de le compléter par des bibliothèques de blocs spécialisés, les « Blocksets », qui viennent s'ajouter à la bibliothèque de base.

Les deux outils Statemate et Stateflow sont actuellement utilisés à PSA.

3.4.3. Conclusion

Pour conclure, les Statecharts représentent un moyen efficace de décrire des automates finis complexes, fondé sur une modélisation graphique modulaire et hiérarchique. Ce dernier aspect paraît très intéressant pour concevoir un système par raffinements successifs. L'approche hiérarchique permet également de disposer de plusieurs niveaux d'abstraction d'un système. Par exemple, les analyses de sûreté de fonctionnement pourraient se baser sur une description du système limitée aux différents modes de fonctionnement, sans descendre davantage dans les niveaux inférieurs plus détaillés, potentiellement superflus dans les études probabilistes. Ce concept constitue d'ailleurs un des principes de la méthode d'évaluation quantitative illustrée dans les prochains chapitres.

Des exemples de modélisation feront l'objet du chapitre 4. Nous avons utilisé Stateflow pour la description comportementale et fonctionnelle d'un système de contrôle-commande. Couplé à l'environnement Simulink, les composantes continue et discrète sont ainsi représentées.

3.5. AltaRica

Le langage AltaRica est issu d'une collaboration entre le LaBRI et de plusieurs industriels [GRIFF]. Il est né en 1996 de la volonté de chercheurs d'établir divers ponts entre : la sûreté de fonctionnement et les méthodes formelles, les analyses quantitatives des dysfonctionnements et les analyses qualitatives des comportements fonctionnels, afin de fournir aux ingénieurs concepteurs de systèmes complexes ou critiques, un atelier outillé. Il est ainsi possible à partir d'une description dans ce langage de simuler un modèle, de générer un arbre de défaillance ou de vérifier qu'il satisfait des propriétés logiques.
C'est un langage basé sur les automates à contraintes. Un des avantages est la possibilité de représenter un système de façon hiérarchique, jusqu'à un niveau atomique, les *composants*, décrits sous forme de *nœuds*. La description d'un composant comporte [POINT] :

- une liste de variables de *flux*. Elles modélisent les informations émises ou reçues par les composants (les entrées/sorties du composant, par exemple un courant dans un composant électrique).
- Une liste de variables *d'états*. Un état du composant est en réalité une affectation de ces variables. L'état d'un interrupteur peut par exemple être décrit par deux variables, l'une décrivant sa position (ouvert ou fermé) et l'autre son état de défaillance (marche ou panne).

- Une liste *d'événements*. Ceux-ci décrivent les phénomènes modifiant l'état interne du composant (c'est-à-dire les valeurs des variables d'état). Un événement est par exemple le changement de position de l'interrupteur ou l'occurrence d'une défaillance.
- Une liste *d'assertions*. L'assertion permet d'exprimer que les valeurs des variables de flux sont liées à celles des variables d'états. Par exemple, lorsque l'interrupteur est fermé, les entrées/sorties doivent avoir la même valeur.
- Une liste de *transitions*. Celles-ci décrivent les changements d'état sur occurrence d'un événement. Une transition est un triplet (c, e, a) où c est une condition, appelée garde, portant sur les variables du composant, e est un nom d'événement et a est une affectation des variables d'état. Une transition n'a lieu que lorsque l'événement se produit et que la condition c est vraie. Après la transition, les variables d'état reçoivent la valeur spécifiée par a.

Le petit exemple suivant modélise le fonctionnement d'une pompe. On définit deux variables d'états de type booléen : *ouvert* et *nominal*. Le premier renseigne sur la position de la pompe (ouvert ou fermé), et le deuxième sur son état de défaillance (nominal ou panne). On spécifie également 4 événements : *bascul*, *bloq*, *bloq_o* et *bloq_f*. L'événement *bascul* fait passer la pompe de l'état d'ouverture à l'état de fermeture et réciproquement. Les trois autres correspondent à des défaillances (respectivement blocage en position courante, en position ouverte et en position fermée). Enfin, on définit une variable de flux booléenne, *debit*, qui donne la valeur du débit fournit par la pompe en fonction de ses variables d'états.

```
node Pompe
        flow    debit : bool  ;
        event   bascul, bloq, bloq_o, bloq_f ;
        state   nominal, ouvert : bool ;

    trans
        nominal and ouvert |- bascul  -> ouvert := false ;
        nominal and not ouvert |- bascul  -> ouvert := true ;
        nominal |- bloq -> nominal := false ;
        nominal |- bloq_o -> nominal := false , ouvert := true ;
        nominal |- bloq_f -> nominal := false , ouvert := false ;

    assert
        debit = ouvert ;
edon
```

Figure 19 : Exemple de modélisation d'un composant

La description des composants est délimitée par les mots clés **node** et **edon**. Ils permettent de les définir en tant que *nœud*.
La hiérarchie permet de décrire l'encapsulation d'un ensemble de composants dans un autre composant (placé à un niveau hiérarchique supérieur) ;

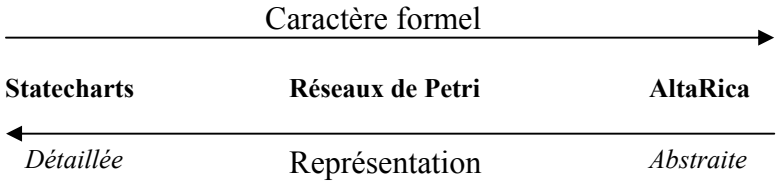
Pour conclure, le langage AltaRica permet de décrire les comportements fonctionnel/dysfonctionnel de composants dans un formalisme à états/transitions. Il est bien adapté à la description de modèles discrets. Bien qu'il n'intègre pas la notion de temps et la possibilité de décrire des comportements dynamiques de composantes continues, il offre néanmoins la possibilité de décrire de façon plus abstraite des comportements. Cette caractéristique présente des avantages certains car la structure des systèmes modélisés permet d'appliquer des techniques de vérification formelle comme le modèle checking, ou encore de

générer automatiquement un arbre de défaillance. Citons, pour finir, les outils de model-checking MEC et TOUPI utilisés conjointement avec des modélisation AltaRica pour vérifier formellement des propriétés [VIN03].

3.6. Conclusion

Nous avons brièvement relevé trois formalismes différents, basés sur une description à états/transitions, qui apparaissent comme étant des candidats potentiels à une modélisation des systèmes discrets de contrôle-commande dans le cadre d'une méthodologie de conception. Chacun possède ses propres caractéristiques et est supporté par des outils plus ou moins puissants. Plusieurs remarques peuvent être faites :

1. La description formelle ou l'application de méthodes de vérification de type model-checking sont directement liées à la capacité de représenter dans ces formalismes un comportement de façon abstraite. Ainsi, l'application de vérification formelle sur un modèle est d'autant plus facilitée que la représentation est abstraite. Cependant, le concepteur doit fournir un effort de modélisation plus important pour agréger ou synthétiser les nombreuses informations de son système. Nous pouvons ainsi classer les trois formalismes en fonction de ces critères :



2. Le caractère hybride des systèmes à modéliser nécessite de coupler ces langages avec une description continue, notamment pour représenter l'évolution dynamique du processus physique. Concernant les réseaux de Petri, de nombreuses approches différentes ont été définies (approches intégrée ou séparée) [AND96]. Nous pensons que les approches telles que les réseaux de Petri hybrides, couplant un réseau de Petri discret avec un réseau de Petri continu, pose quelques difficultés conceptuelles et reste difficilement exploitable au niveau industriel. La délimitation entre modèle continu et modèle discret est moins claire que lorsqu'on utilise deux formalismes ou deux modèles. La technique de modélisation séparée conserve un caractère plus général. L'utilisation de modèle propre à chaque aspect permet en particulier de conserver le potentiel de modélisation dans chacun des domaines. Ainsi, l'exploitation des réseaux de Petri en tant que formalisme de modélisation discrète, utilisé par exemple comme un moniteur d'équations, semble être une bonne réponse à des besoins de conception hybride. A l'heure actuelle, malheureusement peu d'outils de réseaux de Petri bénéficient du potentiel que possède par exemple l'environnement Simulink/Stateflow. Concernant AltaRica, des recherches sont en cours pour définir des extensions temporelles et hybrides [CAS02]. Ces voies sont donc prometteuses.

3. Enfin, la possibilité de décliner ces modèles comportementaux en des modèles de sûreté de fonctionnement ou, tout du moins, d'en faciliter la construction, constitue un aspect important dans une méthodologie. Comme nous l'avons vu, un modèle AltaRica est directement traduisible en un arbre de défaillance via une représentation en BDD (Diagramme de Décision Binaire) et les réseaux de Petri permettent d'élaborer facilement un graphe de Markov par la construction du graphe d'accessibilité. Nous estimons que l'approche markovienne constitue un moyen

efficace d'évaluation de la fiabilité des Systèmes Dynamiques Hybrides (SDH). Dans cet objectif, nous proposerons dans le chapitre 3 une méthode de construction d'un graphe de Markov qui s'affranchit en partie des hypothèses restrictives des réseaux de Petri markoviens et, plus généralement, du formalisme employé. De ce fait, un SDH modélisé par exemple à l'aide d'un outil comme Simulink/Stateflow, pourra être évalué par un graphe de Markov adapté.

Ainsi, face aux multiples difficultés de modélisation hybride, de vérification/validation et de sûreté de fonctionnement, ces formalismes répondent plus ou moins efficacement à nos besoins méthodologiques.
La nécessité d'intégration de toutes ces activités au sein de la conception a permis à des ateliers logiciels de voir le jour. Nous allons en examiner deux.

4. Quelques ateliers d'aide à la conception existants

4.1. Introduction

Les progrès de la micro-informatique, d'une part, et la complexité grandissante des systèmes à concevoir, d'autre part, ont contribué à la mise sur le marché de « progiciels » orientés sûreté de fonctionnement permettant d'aider l'ingénieur dans sa difficile tâche de modélisation et d'analyse (dys)fonctionnelle. Ces outils sont munis d'interfaces graphiques plus ou moins interactives, plus ou moins ergonomiques, plus ou moins « intelligentes » pour saisir, visualiser, animer, valider et calculer les modèles fonctionnels et/ou dysfonctionnels des systèmes étudiés [SIG96]. Ces ateliers permettent notamment de simuler le comportement des systèmes en présence de défaillances et de générer automatiquement des modèles de sûreté de fonctionnement. Ces générateurs automatiques facilitent grandement le travail de l'ingénieur, puisqu'ils le libèrent, au moins partiellement, de la construction d'un modèle d'un haut niveau d'abstraction, tâche beaucoup moins intuitive et naturelle qu'une description comportementale fonctionnelle de son système. De plus, des modifications fonctionnelles, même mineures, impliquent en général de recommencer intégralement la construction de ce modèle de défaillance.
Par ailleurs, ces ateliers logiciels deviennent de plus en plus performants ; ils ne se limitent plus à une simple simulation et/ou une génération automatique de modèles mais s'enrichissent au fur et à mesure de nouveaux modules dédiés à différents aspects de la sûreté de fonctionnement ainsi que de nombreuses passerelles vers d'autres outils du marché. Ils permettent notamment :

- de simuler graphiquement et interactivement la propagation des dysfonctionnements, injectés manuellement ou automatiquement,
- de vérifier la cohérence et la complétude des modèles,
- de générer automatiquement des modèles en fonction de l'étude à faire (AMDEC, arbre de défaillance, scénarios conduisant à une panne, réseau de Petri,…),
- de lancer des calculs sur les modèles obtenus,
- d'aider au diagnostic,
- d'appliquer des techniques de vérification formelle sur le modèle,
- …

Pour l'utilisateur, les aspects simulation sont concrétisés par un simulateur graphique qui, par le biais de couleurs qui changent, permet de visualiser facilement ce qui se passe. Il peut ainsi

vérifier que le comportement de son modèle est conforme à la réalité et le modifier le cas échéant.
Nous allons passer en revue deux ateliers existants : SIMFIA et OCAS.

4.2. SIMFIA

SIMFIA est un progiciel développé et distribué par la société Sofreten.
A partir de l'acquisition des connaissances issues de l'analyse fonctionnelle du système, il permet d'analyser et de simuler son comportement global et d'automatiser les études de FMDS (Fiabilité, Maintenabilité, Disponibilité, Sécurité).
Cet atelier est composé d'un ensemble structuré de modules qui s'organisent autour d'un noyau constituant le cœur, SOFIA. Il mémorise et gère les données (ou connaissances) saisies par l'utilisateur.

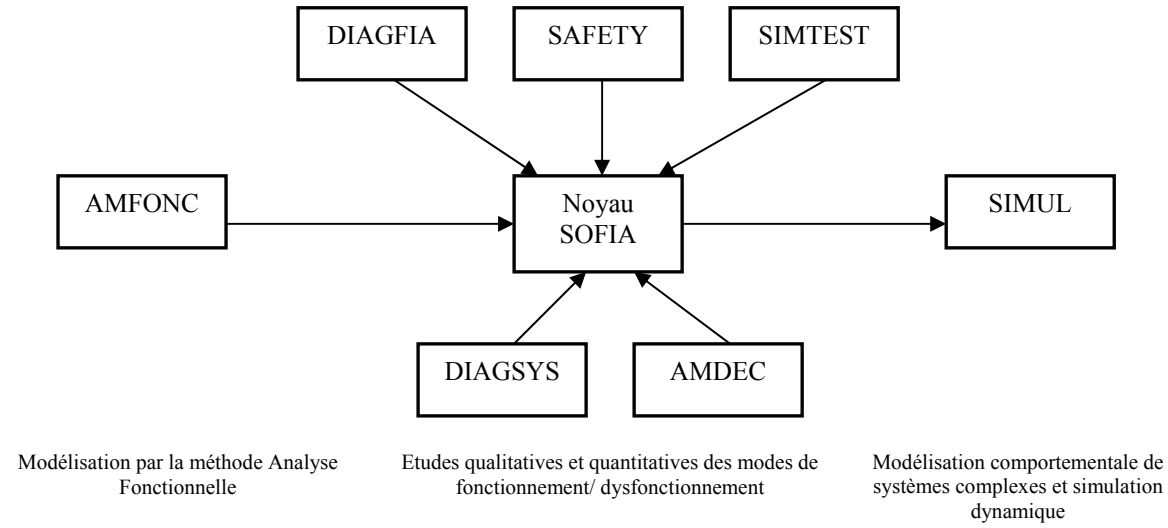


Figure 20 : Modules de SIMFIA [SIMFI]

Le principe de modélisation d'un modèle SOFIA consiste à représenter l'ensemble des éléments physiques ou fonctionnels d'un système sous la forme de *blocs fonctionnels*. Chaque bloc fonctionnel est caractérisé par son état qui peut prendre trois valeurs différentes : *nominal*, *absent* et *dégradé*. Les interactions entre les différents blocs fonctionnels sont ensuite décrites sous la forme de « polynômes logiques » qui permettent de définir des fonctions de sorties d'un bloc par rapport à ses fonctions d'entrée et à ses états internes.

Les principaux modules s'articulant autour du noyau SOFIA sont :

- AMFONC : celui-ci permet de créer un modèle SOFIA en utilisant un autre formalisme que le langage SOFIA du type « schéma de flux ».
- AMDEC : il permet de générer automatiquement les tableaux AMDEC à partir des données du noyau SOFIA sans que l'utilisateur n'ait à faire le travail d'analyse et de rédaction.
- SAFETY : il détermine et trace automatiquement l'arbre de défaillance (ou de « bon fonctionnement ») pour un événement spécifié (panne, marche, dégradé) et pour une mission donnée.

- DIAGFIA : il permet de construire les diagrammes de fiabilité depuis une modélisation élaborée avec le noyau SOFIA.
- DIAGSYS (SYStème d'aide au DIAGnostic) : il génère automatiquement un logiciel d'aide au diagnostic de pannes et de maintenance.
- SIMTEST : il permet d'évaluer la testabilité du système représenté dans le noyau SOFIA (aide à la mise en place de points de test, évaluation quantitative d'un dispositif de test, génération d'une stratégie de diagnostic).
- ALLOC : celui-ci permet à l'utilisateur de réaliser des allocations de type FMDS à différents niveaux d'un système.

L'atelier SIMFIA est actuellement utilisé à PSA pour construire des arbres de défaillance.

4.3. OCAS

L'outil OCAS (Outil de Conception et d'Analyse Systèmes) a été développé par Dassault Aviation et est distribué par GFI Consulting. Il permet de formaliser, structurer et de capitaliser les connaissances de sûreté de fonctionnement acquises sur des systèmes. Ce logiciel s'articule autour d'une modélisation dans le langage AltaRica permettant de décrire le comportement fonctionnel et dysfonctionnel de composants élémentaires. Ces composants, une fois modélisés, sont ré-exploitables grâce à l'utilisation de bibliothèques de composants.
Une fois ces composants (fonctionnels ou physiques) modélisés, ils sont assemblés graphiquement dans le logiciel afin de former l'architecture physique ou fonctionnelle d'un système sous forme hiérarchique. Celui-ci est alors simulable interactivement en injectant manuellement des événements définis lors de la formalisation des composants atomiques afin de valider et vérifier le comportement du système, notamment en présence de défaillance pour tester les mécanismes de tolérance aux fautes implémentés. Un vérificateur permet également de contrôler la cohérence de l'architecture par rapport aux entrées/sorties des composants, et de vérifier la complétude de ceux-ci.
Il est enfin possible de générer automatiquement des arbres de défaillance à partir d'événements redoutés identifiés au format Aralia Simtree et Cecilia Arbor.

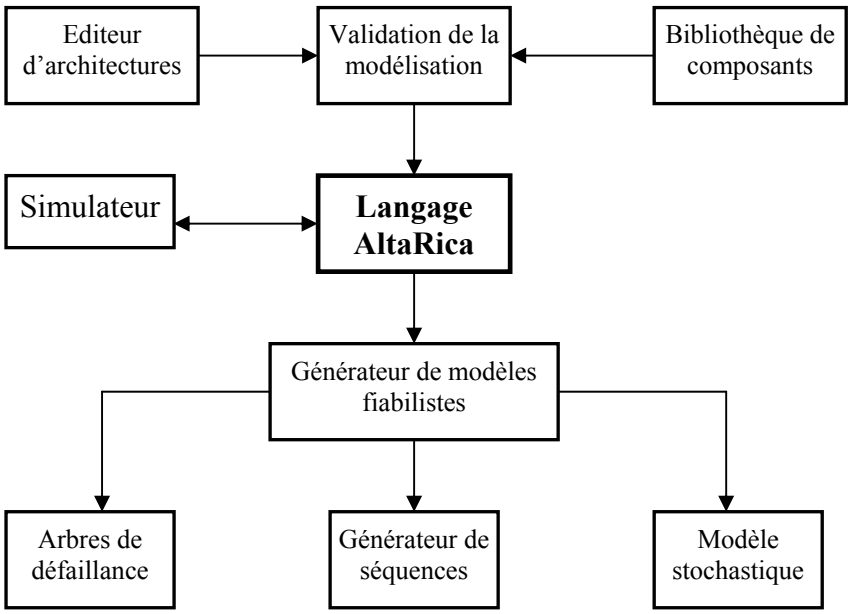


Figure 21 : Schéma directeur de OCAS

OCAS est actuellement utilisée par Dassault Aviation pour réaliser les études de sûreté de fonctionnement de ses différents modèles d'avion (génération automatique d'arbres de défaillance).

4.4. Conclusion

Ces ateliers logiciels ne cessent de s'améliorer, de par la multiplication de modules rattachés à un noyau principal. Ces modules permettent de traiter de plus en plus d'aspects différents dans la conception, et ne se limitent plus uniquement à des analyses dysfonctionnelles mais garantissent également la conformité d'un modèle par rapport à des exigences fonctionnelles (validation et vérification). De plus, les améliorations concernent également la puissance de modélisation dans le langage « noyau », qui à terme, ne se limitera plus à une vision « statique » et abstraite d'un système, mais inclura également les notions temporelles et hybrides. L'étape ultime sera atteinte lorsque les problématiques de fiabilité des SDH pourront être modélisées et traitées dans ce type de logiciel. Les voies sont prometteuses, car déjà certaines extensions permettent de générer automatiquement des modèles de type réseaux de Petri.
Notre travail méthodologique consiste, à l'instar de ces logiciels, à définir un environnement complet de conception, c'est-à-dire intégrant les éléments suivants :

- un formalisme permettant de représenter le comportement des SDH ; les langages basés sur les automates, couplés à un modèle continu, sont les plus pertinents pour concevoir des systèmes de contrôle-commande avec la partie opérative en interaction avec ceux-ci,
- un outil supportant ce formalisme, et permettant entre autres d'étudier le modèle par des simulations,
- des passerelles vers des outils d'analyse formelle tels que le model-cheking, en complément d'une représentation structurée ; ces outils pourraient également être intégrés dans l'environnement de conception, en encapsulant les principes théoriques (formulation des propriétés et intégration d'un moteur de preuve) afin que ceux-ci soient le plus transparent possible aux yeux du concepteur,
- faciliter les différentes démarches de sûreté de fonctionnement (modélisation et évaluation de la fiabilité des SDH).

Un aspect intéressant à développer consisterait à compléter les outils de modélisation existant (par exemple MOCA-RP pour les réseaux de Petri) avec les éléments manquants. Concernant l'analyse dysfonctionnelle, la génération automatique d'une AMDEC proposée dans [JAM01c] et la construction d'un graphe de Markov agrégé, présentée dans le chapitre 3, pourraient être automatisées et intégrées dans cet environnement.

Nous allons à présent détailler une technique de vérification formelle qui a été évoquée à plusieurs reprises dans ces premiers chapitres, à savoir le *model-checking*. Cette technique entre dans le cadre de l'analyse fonctionnelle des systèmes.

5. Analyse Fonctionnelle des systèmes de contrôle-commande

5.1. Vérification et validation formelles

5.1.1. Introduction

La vérification formelle et la validation des systèmes réactifs sont des éléments clé pour la sûreté et le fonctionnement. Ils doivent s'intégrer au plus tôt dans le cycle de vie.
Ils doivent permettre de garantir que le système répond bien aux exigences fonctionnelles et de sûreté mais également de s'affranchir des éventuelles fautes de conception.

On distingue classiquement trois types de propriétés pour ces systèmes :

- **les propriétés de sûreté** qui expriment que des configurations ne doivent jamais être réalisées (par exemple une commande peut-elle avoir des effets indésirés ?). Sur ce point précis, la vérification permet aussi d'apporter la preuve que des événements redoutés apparaîtront, dégageant ainsi la stratégie de mode dégradé.
- **des propriétés de temps de réponse** qui expriment des contraintes sur le temps minimum ou maximum séparant certains événements ou caractérisant certaines évolutions (par exemple la réponse à une commande peut-elle être retardée indéfiniment ?),
- **les propriétés de vivacité** qui expriment des évolutions qui doivent être possibles ou inévitables (par exemple pas de blocage du système, d'interblocage dû à des partages de ressources ou bien, par exemple, une commande peut-elle être sans effet ?…).

Notons que les erreurs de conception constituent l'une des sources de défaillance les plus probables (fautes humaines).

La vérification formelle est en général basée sur trois éléments :

- **un formalisme permettant la modélisation** (comportementale) du système, par exemple sous la forme d'un automate,
- **une spécification de la propriété désirée** ; on utilise pour cela un langage de spécification de propriété, par exemple une logique temporelle
- **un algorithme de vérification**

Les techniques classiquement utilisées pour mener à bien la vérification des propriétés sont la simulation (vérification dynamique) et les techniques de vérification formelle de type « model checking » (vérification statique).

5.1.2. La simulation

La simulation apparaît être la solution la plus simple et la plus utilisée pour mener à bien les études de vérification. L'avantage principal de cette approche est de pouvoir traiter facilement les systèmes hybrides (mélangeant les aspects continus et discrets). On peut représenter l'évolution des variables continues par exemple par des systèmes d'équations différentielles pilotés par des automates discrets (Statecharts, Grafcet, réseaux de Petri…). Il est également

possible de mesurer les performances temporelles des systèmes modélisés. La simulation s'applique à de nombreux modèles mais il est évident que cette approche expérimentale ne permet pas de considérer toutes les évolutions et ne garantit donc pas l'absence d'erreurs dans le modèle établi. C'est le problème **d'exhaustivité de la vérification**.
Un autre moyen permettant de révéler puis d'éliminer d'éventuelles erreurs de conception est de réaliser des tests. Le test exhaustif d'un système par rapport à toutes ses entrées possibles est pratiquement infaisable. Les méthodes de détermination des jeux de test peuvent être classées selon deux points de vues : sélection des entrées de test, génération automatique des entrées de tests.
Les problèmes de couverture de tests ou d'exhaustivité de la simulation amènent progressivement les concepteurs à développer des approches plus formelles telles que le model-checking.

5.1.3. Le Model-Checking

Cette technique permettant de vérifier si une propriété est vérifiée ou non est basée sur la description du système sous forme d'un automate. La vérification consiste à générer l'espace des états accessibles, d'examiner tous ces états et d'identifier ceux qui sont en contradiction avec la propriété [OFT97].
Le model-checking ne requiert aucune interaction avec l'utilisateur qui soumet à la fois son modèle du système et les propriétés à satisfaire à un « model-checker » et attend que celui-ci ait terminé l'examen de toutes les situations possibles.
La modélisation du système est basée généralement sur des formalisme à états/transitions (Statecharts, réseaux de Petri, AltaRica…) alors que les propriétés sont exprimées à l'aide de formules de logique temporelle (TCTL, CTL, CTL*).

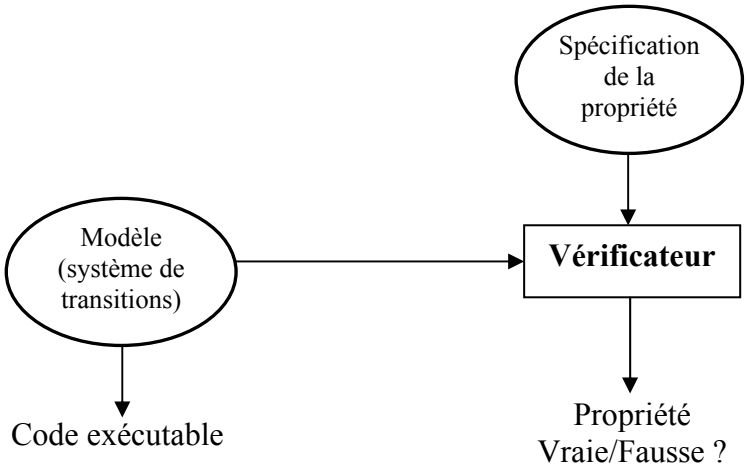


Figure 22 : Schéma de vérification

Un inconvénient de ce type d'approche est de ne pouvoir traiter que des modèles où le nombre d'états est fini. Il est en effet impossible de faire une énumération exhaustive d'un nombre d'états infini (problème d'explosion combinatoire générant des dépassements mémoires).

Ainsi, la prise en compte de l'environnement continu est très difficile, on utilise pour cela des *méthodes d'abstraction* ([SCH99], [VIN03]) pour simplifier le système.

L'avantage indéniable de ce type de méthode est l'obtention de l'exhaustivité des situations qui amènent le système à contredire la propriété. Si le résultat aboutit à la satisfaction de celle-ci, l'utilisateur sera assuré que la propriété est vérifiée (ce qui n'est pas le cas de la simulation). En général, les outils implémentant ce type de méthode donnent un contre-exemple (sous forme de scénarios, de valeurs des entrées/états…) si la propriété est fausse.

Pour résumer, le model-checking est une méthode exhaustive et, en grande partie, automatique. Le travail de l'ingénieur se limite à la construction d'un modèle formel du système et la spécification des propriétés.

5.2. Les propriétés à vérifier

Nous allons faire un inventaire, non exhaustif, des propriétés qui doivent être vérifiées. Ces propriétés sont relatives soit à une bonne conception, soit à une conception conforme aux spécifications.
Nous pouvons distinguer les propriétés « génériques » qui doivent être vérifiées pour tout système et les propriétés propres à chaque système.

5.2.1. Les propriétés de vivacité

1) Propriété d'atteignabilité :

Cette propriété doit répondre à la question : tous les états qui ont été définis sont-ils accessibles ? En effet si certains états ne peuvent être atteints, cela signifie soit qu'il y a une erreur de conception (existence de transitions infranchissables), soit que certains états définis sont inutiles (par exemple existence de modes dégradés qui n'ont pas lieu d'être) et qui ne feraient qu'alourdir le code exécutable.

2) Absence de blocage

Une propriété essentielle à vérifier est l'absence de blocage (ou d'état puits). Elle énonce que le système ne se trouve jamais dans une situation où il lui est impossible de progresser. C'est une propriété de correction pour des systèmes supposés ne jamais se terminer. La présence d'un blocage dans un système se manifeste par une « paralysie » de celui-ci qui ne peut plus évoluer. Le fonctionnement n'est donc plus assuré. D'un point de vue automate, l'existence d'un blocage est manifeste lorsque plus aucune transition n'est validée.
Le blocage peut être d'origine structurelle (aucune transition aval d'un état n'a été spécifiée, et donc le système ne peut plus en sortir), ou bien la condition rattachée à la transition n'est jamais vérifiée (par exemple *Vitesse_Véhicule>800 km/h* !!!).

3) Absence d'interblocage :

Le problème de l'interblocage est très classique pour les systèmes temps-réels. On distingue deux types : les interblocages de ressources (attente d'une ressource déjà attribuée à un autre processus) et les interblocages de communication (deux processus attendent des messages qui ne peuvent être émis car les processus sont en attente). Dans les deux cas, les interblocages correspondent donc à un état anormal dans l'exécution d'au moins un des deux processus. Cet état est caractérisé par le fait que chaque processus est bloqué et ne peut être débloqué que par l'autre processus.

4) La vivacité

Une propriété de vivacité énonce que, sous certaines conditions, quelque chose finira par avoir lieu. Les exemples sont nombreux : « toute requête finira par être satisfaite », « si on appelle l'ascenseur, la cabine arrivera un jour », « le feu passera au vert », …

5) Absence de conflit

Cette propriété permet notamment d'assurer le caractère déterministe du système. Un état initial est susceptible d'avoir plusieurs états finaux. Le franchissement d'une transition entre l'état initial et l'un des états en aval doit être unique car, dans le cas contraire, des comportements différents peuvent avoir lieu pour une même situation initiale. Cela se traduit par exemple par des conditions portées par plusieurs transitions en conflit (structurel) qui peuvent être à un instant donné toutes vraies. Ce type de situation doit être identifié. Les conditions portées par les transitions en conflit structurel doivent être complémentaires.

5.2.2. Les propriétés de sûreté

Une propriété de sûreté énonce, que sous certaines conditions, quelque chose ne se produit jamais. En général, il s'agit de vérifier que quelque chose d'indésirable ne se produira jamais (état redouté, débordement mémoire…). On peut également valider le fonctionnement du système en présence de défaillances ; par exemple, dans le cas de l'automobile où des modes dégradés sont implémentés, on peut vérifier :

6) qu'en cas de défaillance prévue, les mécanismes de tolérance aux fautes s'activent normalement et le système se place bien dans l'un des modes dégradés,

7) un scénario de défaillances non prévu peut-il amener le système vers un état redouté ? (perte de la fonction direction, consigne actionneur erronée et non détectée…). Un événement redouté évident est la perte de la fonction. La vérification de la propriété peut alors consister à caractériser dans un premier temps la perte de la fonction (certaines variables ne sont plus évaluées ? restent à 0 ?), puis à trouver l'origine de ce dysfonctionnement grâce au moteur de preuve qui fournira l'un des scénarios de défaillance recherché.

5.2.3. Les propriétés de performance fonctionnelle

Ces propriétés sont spécifiques à chaque système. Les systèmes de contrôle-commande sont en général soumis à des contraintes de temps réel fortes. On peut, par exemple, vérifier que les contraintes de temps critiques sont bien vérifiées (temps de réponse maximal, temps d'initialisation d'une fonction, …).
La prise en compte du temps dans les techniques de model-checking peut alors nécessiter de discrétiser la variable temporelle si le modèle est à temps continu afin de ramener celui-ci à une représentation du système sous la forme d'un automate à états finis. En effet, les systèmes temps réel sont non-dénombrables à cause de la densité du temps [OLI94].

Dans la majorité des cas, le modèle doit être *conditionné* pour pouvoir être soumis au moteur de preuve.

5.3. Mise en œuvre de la technique de model-checking sur un modèle

5.3.1. Abstraction d'un modèle

On désigne par « méthodes d'abstraction » une famille de techniques utilisées pour simplifier les automates. Le terme d'abstraction renvoie à la nature des simplifications mises en œuvre, qui généralement choisissent d'ignorer certains aspects de l'automate examiné.
Il y a globalement deux raisons pour lesquelles une simplification de l'automate est nécessaire :

- **Taille de l'automate** : le modèle est de taille trop grande pour être vérifié par l'outil dont on dispose. C'est un cas très fréquent puisque le model-checking est sujet à l'explosion combinatoire. Cette situation peut résulter de la présence de trop nombreuses variables, ou de trop nombreux automates mis en parallèle ou encore de nombreuses horloges dans les automates temporisés.

- **Type de l'automate** : le model-checker dont on dispose n'accepte pas les automates utilisant telle ou telle construction : variables entières, horloges, priorités…

Ainsi, l'établissement d'un modèle formel implique nécessairement d'appliquer un certain nombre de règles de construction afin que les démonstrations puissent être obtenues. Ces règles sont propres aux outils de model-checking. A partir d'un modèle comportemental issu des spécifications, des modifications sont susceptibles d'être apportées, on parle alors de « conditionnement du modèle ». Ces règles peuvent consister en l'élimination de variables continues ou non bornées, de compteurs, …

5.3.2. Démarche globale d'intégration du model-checking dans la conception

Tous les processus de développement classiques des systèmes, en particulier de contrôle-commande, doivent comprendre les phases de construction et de validation : définition des exigences, spécification, conception, programmation, tests. Pour les systèmes critiques, par exemple tels que le X-by-Wire, le processus de développement doit être plus riche puisque les

phases d'analyse de la sécurité doivent prendre une place importante. La prise en compte de ces analyses tout au long du processus de développement a un effet bénéfique sur la confiance que l'on pourra attendre de la sécurité du système puisque les erreurs de conception pourront être révélées le plus tôt possible.
L'utilisation des méthodes formelles à différents niveaux du cycle de développement permettra alors d'identifier les incohérences et/ou les incomplétudes du modèle quel que soit le degré de détail ou d'abstraction. En effet, un avantage de la technique de model-checking est de pouvoir procéder à des vérifications en oubliant des détails non significatifs, c'est-à-dire en travaillant à un grand degré de généralité, grâce notamment aux mécanismes d'abstraction. Au fur et à mesure que le modèle est raffiné, les propriétés pourront être formulées de façon plus précise.
Donc, à chaque niveau de raffinement, l'application du model-checking permettra d'éviter d'introduire des erreurs de conception (cf. figure 23), dans la mesure du possible :

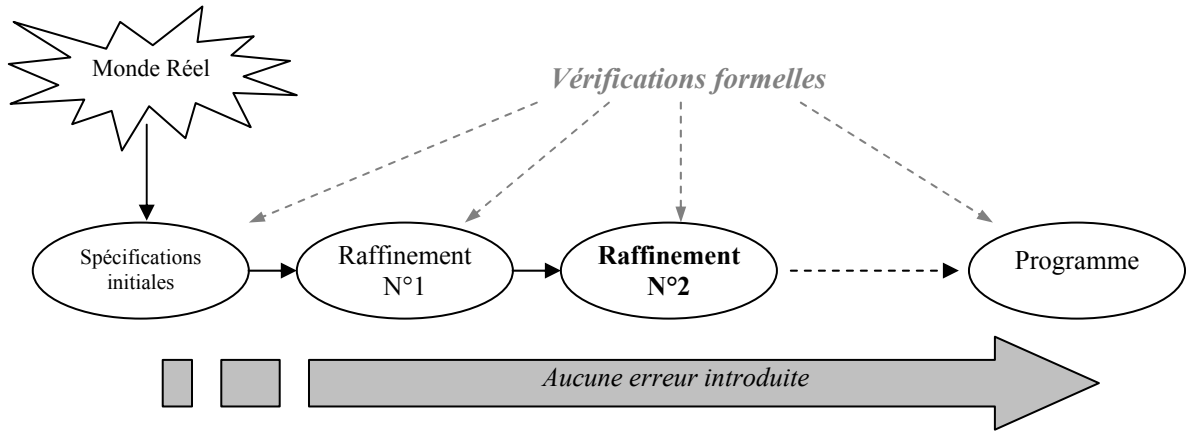


Figure 23 : Vérification du modèle à chaque niveau de raffinement [OFT97]

L'utilisation de méthodes formelles doit s'inscrire dans un cadre méthodologique bien stabilisé. Cependant, elles ne doivent pas fondamentalement bouleverser les méthodes de conception déjà bien ancrées (utilisation de langages de spécification de modèles et de propriétés spécifiques), car elles ne feraient qu'alourdir la tâche du concepteur/testeur et donc allonger considérablement les délais et engendrer des surcoûts. C'est dans cet objectif que de nouveaux outils supportant le model-checking font leur apparition sur le marché, destinés à un usage industriel. On peut citer par exemple l'outil SBC (Safety Checker Blockset) développé et distribué par la société TNI-Valiosys destiné à rendre les méthodes formelles de model-checking accessibles aux concepteurs modélisant sous Simulink/Stateflow. La formulation des propriétés, qui impliquait originellement le recours à des langages telles que la logique temporelle, pas toujours très accessibles au niveau industriel, devient totalement transparente grâce à l'utilisation d'opérateurs de preuve intégrés dans l'environnement Simulink/Stateflow.

Notons également que la vérification de propriétés par model-checking n'a pas la vocation à supplanter les techniques classiques (jeux de test, simulation), mais doit être considérée comme un complément. Au contraire, le model-checking peut même contribuer à orienter le concepteur vers des jeux de test spécifiques (par exemple pour mettre en défaut une propriété…).

Ainsi, la vérification de propriétés peut être effectuée parallèlement à la modélisation et être utilisée conjointement à la simulation comme décrit sur la figure 24 ; le résultat obtenu suite à une démonstration de propriété peut être :

- la propriété est prouvée VRAIE,
- la propriété est prouvée FAUSSE et un contre-exemple est trouvé.

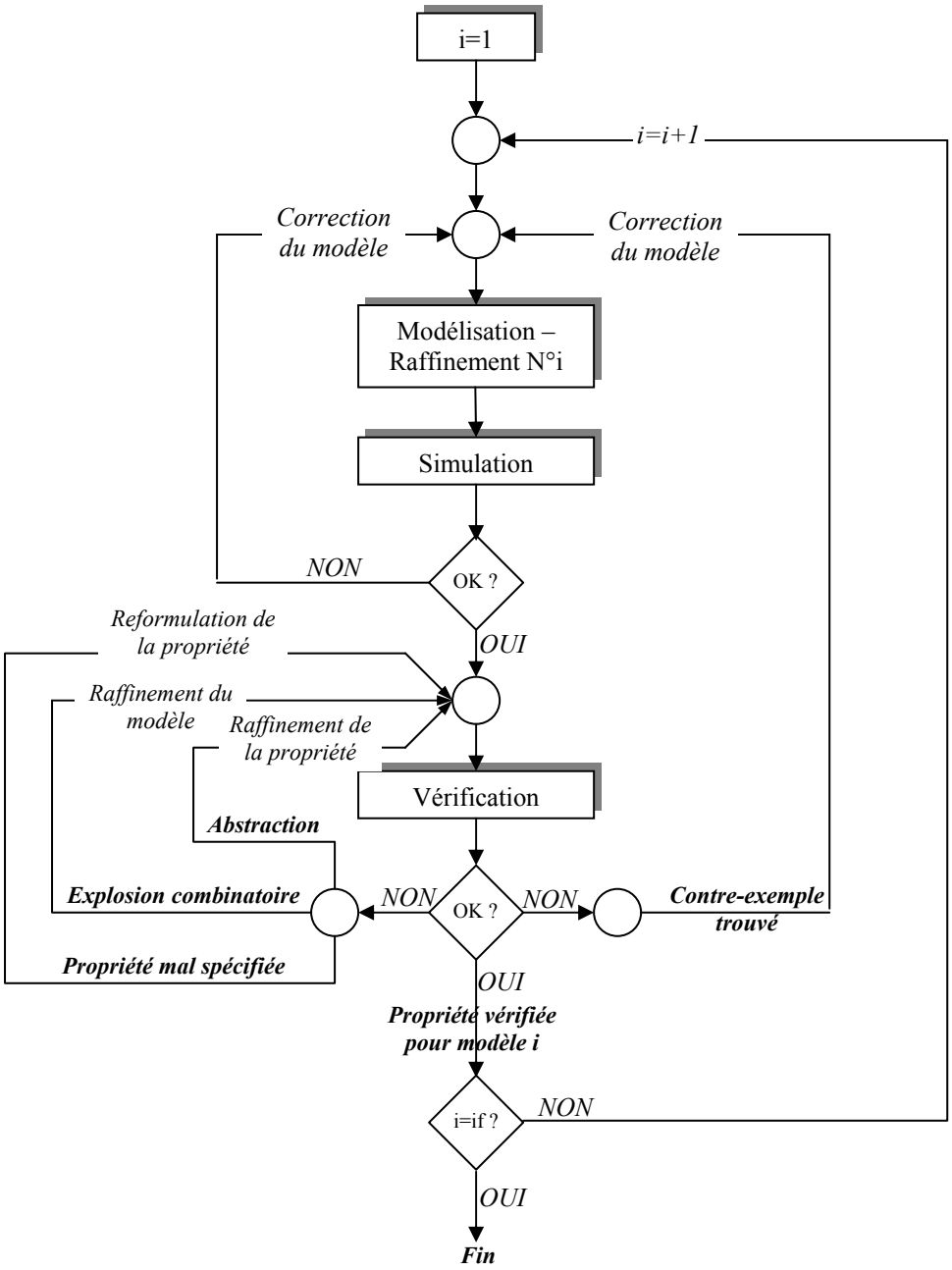


Figure 24 : Démarche rigoureuse de modélisation/vérification d'un système

Le contre-exemple correspond à une séquence d'entrées mettant en défaut la propriété. Cependant, remarquons que la découverte d'un contre-exemple peut ne pas toujours être attribuée à un bug dans les spécifications. Une propriété peut être violée dans les cas suivants :

- la propriété a été mal spécifiée et donc le résultat obtenu n'est pas pertinent,

- la séquence d'entrées ne correspond pas à une situation réelle (vitesse>800 km/h, exp(n)<0, …), en particulier lorsque des entrées sont soumises à des contraintes qui n'ont pas été formalisées (dépendance de deux entrées) ou aux abstractions faites par le moteur de preuve (incapacité à traiter rigoureusement les variables réelles sujettes à l'explosion combinatoire) ou par un conditionnement du modèle,
- le modèle contient véritablement un bug. Une correction doit s'en-suivre.

Il est donc indispensable de vérifier la conformité des résultats obtenus. Les différents cas sont représentés sur la figure 24. L'existence d'un contre-exemple doit demander un travail d'analyse de la part du concepteur pour en vérifier la pertinence. Soit celui-ci est évident, soit il peut par exemple être réinjecté sur le modèle initial sous la forme d'une simulation afin de contrôler le comportement effectif du système réel. Les contraintes supplémentaires, la prise en compte des variables réelles, des compteurs, du temps, … c'est-à-dire la considération d'un modèle dont le degré de raffinement est plus grand, fera que le scénario identifié n'amènera pas à contredire la propriété sur le modèle réel, et n'aura pas d'impact sur le fonctionnement ou la sécurité du système. Les résultats obtenus sont donc qualifiés de pessimistes (à l'instar des arbres de défaillance pour les analyses dysfonctionnelles quantitatives des SDH).
Par contre, si une propriété est démontrée comme étant vraie sur le modèle abstrait, alors a fortiori elle le sera également sur le modèle initial. En effet, le modèle formel possède des caractéristiques beaucoup plus générales et le niveau de détail est plus faible.

5.4. Conclusion

Jusqu'alors, la vérification par model-checking exigeait une certaine expertise de la part des utilisateurs. Loin d'être une approche « presse-bouton » quasi magique, elle était vite confrontée à des obstacles difficiles à surmonter. De plus, il était inévitable de devoir formaliser les propriétés dans des langages tels que la logique temporelle, loin d'être évidents à maîtriser.

Cependant, l'utilisation de cette technique commence à se répandre dans les milieux où la conception exige une attention particulière sur les problèmes de sûreté. De plus, son emploi devient relativement aisé grâce à la multiplication des outils dédiés et applicable sur des systèmes de taille conséquente de par l'amélioration de la puissance des machines. On arrive ainsi de plus en plus à définir une approche intégrée où la simulation, les tests et les méthodes formelles sont utilisés de façon complémentaire afin de garantir une bonne conception. Le nombre d'erreurs de conception et les comportements inattendus pourront être détectés au plus tôt dans le cycle de conception.

Néanmoins, cette technique nécessite de conditionner le modèle afin qu'il soit traitable par le moteur de preuve. Cela passe par des règles de modélisation, des recommandations ou des interdictions sur les principes de modélisation. D'autres recommandations permettent simplement de contourner les problèmes d'explosion combinatoire.

Enfin, concernant la méthodologie de conception définie dans [JAM01c], l'emploi de cette technique constitue un moyen efficace de vérifier qu'un ensemble de fonctions coopérant par l'intermédiaire d'une partie opérative (après l'ajout de l'interprétation sur les différents réseaux de Petri) possède les « bonnes » propriétés. Cette technique est donc complémentaire à l'utilisation de formalismes qualifiés de formels.

6. Analyse Dysfonctionnelle des systèmes de contrôle-commande

6.1. Introduction

Lors d'une étude prévisionnelle de sûreté de fonctionnement des systèmes en phase de conception, nous sommes amenés à étudier une ou plusieurs architectures fonctionnelles sur leur capacité à gérer et recouvrer un certain nombre de défaillances de composants. Une telle évaluation doit aider le concepteur à trouver un compromis optimal entre une solution fonctionnelle robuste (reconfigurations, ajout de modes dégradés) et le coût engendré par ces solutions.
Parmi les nombreuses techniques d'évaluation probabilistes existantes, deux catégories de méthodes sont à distinguer : les méthodes basées sur une approche analytique et les méthodes basées sur une approche expérimentale [LAP95]. Ces deux approches exploitent un modèle ayant un niveau d'abstraction du système plus ou moins grand ; ainsi les méthodes analytiques telles que les graphes de Markov, les arbres de défaillance reposent sur la construction d'un modèle axiomatique qui, après traitement, fournit l'ensemble des grandeurs recherchées, en général la disponibilité et la fiabilité. Les méthodes expérimentales s'appuient sur l'observation du comportement du système en cours de fonctionnement. Le modèle utilisé est soit empirique (un modèle de simulation), soit physique (étude d'un prototype ou d'un système réel). Le niveau de détail de ces modèles est donc plus grand et reflète plus fidèlement le fonctionnement réel du système réel. Les techniques d'injection de faute ou de simulation (par exemple Monte-Carlo) entrent dans cette catégorie. Les mesures de sûreté de fonctionnement correspondent à des estimateurs car elles sont issues d'un traitement statistique des résultats.

La simulation est la plus flexible des techniques utilisées car elle permet en principe d'étudier des modèles avec n'importe quel niveau de détail. Pour un non spécialiste, la simulation a plus de crédibilité que les modèles analytiques car elle est « plus proche » du système réel et nécessite moins de simplifications et quasiment pas d'hypothèse spécifique [INC01].
Cependant, la souplesse offerte par la simulation se paye cher : les simulations sont souvent gourmandes en ressources passives (par exemple la mémoire vive) et en temps de calcul lorsque les modèles utilisés sont très réalistes ou bien si l'on souhaite obtenir des résultats avec une précision élevée. De nombreuses techniques dites d'accélération de la simulation ont été développées afin de réduire ces temps de calcul, moyennant un certain conditionnement du modèle [GAR98].

Concernant les méthodes analytiques basées sur la construction d'un graphe de Markov, de nombreux travaux relatifs à leur application dans le domaine de la fiabilité ont été réalisés ces dernières années.
Les évaluations quantitatives menées dans le cadre de la sûreté de fonctionnement ont pour but de caractériser formellement la nature aléatoire des phénomènes engendrant les défaillances. Ainsi les changements d'états, les dates d'occurrence des défaillances ou d'autres événements indéterministes sont formalisés mathématiquement et étudiés à l'aide de processus stochastiques. Les processus de Markov constituent une classe importante, s'adaptant à de nombreux domaines de l'ingénieur (économie, météorologie, militaire, informatique, biologie…), et couramment utilisée pour l'étude des systèmes de production

(réseaux de file d'attente, politique de maintenance [AMO99]) et dans le domaine de la sûreté de fonctionnement (évaluation de la fiabilité, disponibilité de systèmes électroniques embarqués, production…). En effet, l'approche markovienne reste séduisante de par sa simplicité conceptuelle, la possibilité de représentation graphique simple sous la forme d'un graphe d'état et l'étude des propriétés géométriques et analytiques (communication des classes d'états, stationnarité, ergodicité [RAC97]). Dès lors que les hypothèses markoviennes sont vérifiées, la mise en équation du problème permet une résolution analytique exacte et rigoureuse par opposition à une évaluation statistique qui ne fournit qu'une estimation de la solution recherchée. De plus, les processus de Markov permettent de modéliser fidèlement un grand nombre de systèmes lorsque ces derniers possèdent un nombre d'états pas trop grand.
L'approche markovienne permet de pallier les insuffisances des méthodes dites statiques (arbres de défaillance, diagramme de fiabilité….) grâce à la possibilité d'intégrer des dépendances fonctionnelles entre plusieurs modules ou encore de modéliser des composants ayant des défaillances de mode commun grâce à l'ajout d'arcs et de transitions supplémentaires entre des états. Enfin, un avantage majeur réside dans la possibilité d'introduire implicitement la notion de temps ; celle-ci intervient à la fois dans l'analyse quantitative de la fiabilité, par exemple lorsque la durée de sollicitation d'un composant ou d'un équipement influe sur sa durée de vie ou sur son taux de défaillance mais aussi qualitativement lorsqu'un système de commande inclut un grand nombre de modes de reconfiguration assurant une continuité du service en cas de défaillances. Pour un ensemble donné de défaillances, une séquence particulière peut faire passer le système d'un état nominal à un état dégradé E1 mais toujours fonctionnel, alors que la même séquence ordonnée différemment a potentiellement des conséquences différentes : l'état d'arrivé est soit redouté ER, soit dégradé E2 mais différent de E1.

Néanmoins, on reproche habituellement à cette technique d'être difficilement applicable pour des systèmes complexes, en raison du nombre d'états générés (risques d'explosion combinatoire) et des hypothèses assez fortes (lois de transition exclusivement exponentielles, dans le cas d'une résolution analytique). Des études visant à diminuer le nombre de ces états par des méthodes de simplification ou d'agrégation ont été réalisées dans le domaine informatique ou des systèmes industriels ([BOB86], [BOU90], [AMO99], [BUC99]). L'inconvénient des techniques proposées est la nécessité de disposer dans un premier temps d'un graphe exhaustif initial puis d'appliquer les principes de simplification développés. Ces approches sont donc difficilement applicables au niveau industriel de par le flot d'équations nécessaire et des principes théoriques devant être mis en oeuvre.

Avant d'aborder le chapitre 3 consacré à la méthode de modélisation et d'évaluation dysfonctionnelle d'un système et basée sur une représentation markovienne d'un SDH, nous faisons un rappel sur la technique la plus courante d'évaluation de la fiabilité de ces systèmes, à savoir la simulation de Monte-Carlo ; puis nous abordons quelques concepts fondamentaux sur les processus stochastiques et markoviens, ainsi que sur la théorie des graphes. Ces rappels seront utiles pour introduire la méthode de construction d'un graphe de Markov agrégé.

Ces deux techniques d'évaluation ont été sélectionnées parmi les nombreuses méthodes existantes car elles constituent un début de solution au calcul de la fiabilité des SDH, contrairement aux méthodes statiques classiquement utilisées. Chacune a ses avantages et ses inconvénients.

Les travaux présentés dans ce mémoire exploitent donc largement l'approche analytique par les graphes de Markov. Le modèle construit pour l'étude des grandeurs de sûreté semble plus adapté à la définition d'une méthodologie de conception de systèmes embarqués. En effet, il fournit un support de représentation du système en présence de défaillances suffisamment « abstrait » pour garantir une bonne compréhension dysfonctionnelle globale du système. Le modèle markovien permet de recenser l'ensemble des modes de fonctionnement prévus (nominaux, dégradés) et les mécanismes de tolérance aux fautes (reconfigurations matérielles et logicielles) ; globalement un tel modèle fournit à l'analyste une bonne estimation qualitative de la robustesse du système vis-à-vis des défaillances.

6.2. La simulation de Monte-Carlo

Pour certains auteurs, les techniques de Monte-Carlo enveloppent toutes les méthodes qui utilisent des variables aléatoires ou pseudo-aléatoires pour modéliser les systèmes. Le système lui-même peut être déterministe ou stochastique [INC01].
C'est une méthode numérique basée sur le tirage de nombres aléatoires. La quantité que l'on désire estimer correspond à l'espérance mathématique d'une variable aléatoire. Le principe consiste à étudier l'évolution d'un système en simulant un modèle générique, représentant le comportement du système (la partie discrète relative à la commande et le processus physique continu), au cours de ce qu'on appelle un *scénario* ou *histoire*.
La quantification de la grandeur recherchée (par exemple la fiabilité, ou la probabilité d'apparition d'un événement redouté en ce qui nous concerne) est alors basée sur l'étude d'un certain nombre de scénarios différents, permettant d'en extraire des résultats statistiques (*estimateurs*).
Pour effectuer une estimation de la probabilité d'apparition d'un événement redouté, on peut associer un estimateur de type binaire à cette probabilité et incrémenter un compteur d'une unité pour chaque histoire dans laquelle l'événement redouté se produit. L'estimation de la probabilité est alors obtenue en faisant le rapport entre le nombre d'histoires ayant connu un événement redouté et le nombre total d'histoires [LAB02b].

L'avantage de ce type d'approche est la quasi-insensibilité par rapport à la complexité et à la taille des systèmes modélisés.
Cependant, dans le cadre de la sûreté de fonctionnement, le modèle simulé est régi par des événements très rares (les défaillances) et des événements très fréquents (événements internes de la partie contrôle-commande et du processus physique), et ce simultanément. La simulation est alors cadencée par de nombreuses occurrences d'événements fréquents qui ne reflètent pas le comportement du système en présence de défaillances. C'est le problème de *simulation des événements rares*. Un nombre important d'histoires est nécessaire pour voir apparaître un événement redouté, impliquant des temps de calcul importants. Ces temps deviennent faramineux si, en plus, l'on souhaite obtenir un intervalle de confiance acceptable.

De nombreuses techniques d'accélération de la simulation permettent de réduire ces temps. Elles sont basées soit sur une diminution de la complexité du modèle, soit sur la réduction du nombre de scénarios à simuler, par exemple en favorisant l'apparition des événements rares [GAR98]. Cependant, ces méthodes ne sont pas toujours faciles à mettre en œuvre, par exemple lorsqu'elles impliquent des hypothèses assez fortes, et/ou ne fournissent pas forcément des estimateurs de qualité.

Néanmoins, le principal reproche que l'on peut faire à cette technique d'évaluation de la sûreté de fonctionnement est l'absence d'un modèle représentatif du comportement dysfonctionnel du système, à l'instar de méthodes graphiques. Cet aspect prend une importance particulière dans le cadre d'une méthodologie de sûreté de fonctionnement pour diverses raisons : capitalisation de l'expérience, support de communication, etc…
Pour cette raison, nos travaux se sont orientés vers l'utilisation des graphes de Markov. Cette méthode a néanmoins été adaptée afin de contourner les limites intrinsèques à celle-ci : élargissement des hypothèses dans le cadre de l'évaluation de la fiabilité des SDH, évitement de l'explosion combinatoire, construction facilitée dans le domaine industriel. Avant d'aborder ce chapitre, nous rappelons quelques concepts et définitions relatifs aux méthodes analytiques.

6.3. Rappel de quelques concepts fondamentaux

6.3.1. Processus stochastique

On appelle processus stochastique un ensemble de variables aléatoires X(t), défini sur un espace de probabilité donné et indexées par un paramètre t appartenant à un ensemble T :

$$\{X(t), t \in T\}$$

En pratique T représente l'**espace du temps**. Il peut être discret ou continu.
Les variables X(t) prennent leurs valeurs dans un ensemble E formé par tous les états possibles du système. C'est l'**espace d'état** qui peut être discret ou continu indépendamment de T.

Un processus stochastique est parfaitement défini par les données suivantes :
- le domaine E des variables aléatoires,
- le domaine T du paramètre t,
- des relations statistiques entre les X(t) pour différentes valeurs de t définies par :

$$F_X(t) = Pr\ [\ X(t_1) \leq x_1\ ;\ \ldots\ ;\ X(t_n) \leq x_n\ ] \qquad \forall\ x=(x_1, \ldots, x_n),\ \forall\ t=(t_1, \ldots, t_n),\ \forall\ n$$

6.3.2. Processus de Markov

Plusieurs classes de processus de Markov doivent être distinguées selon la nature discrète ou continue de l'espace d'état E et du temps T. L'étude quantitative de la fiabilité des systèmes impose une modélisation avec un temps continu. Nous ne traiterons que le cas où l'espace E est discret. Nous emploierons indifféremment la terminologie chaîne ou processus de Markov pour indiquer une chaîne de Markov à temps continu.

	T discret	T continu
E discret	**Chaînes de Markov à temps discret**	**Chaîne de Markov à temps continu**
E continu	**Processus de Markov à temps discret**	**Processus de Markov à temps continu**

Figure 25 : Les différents processus de Markov

<u>Définition</u> : Processus markovien
Soit {X(t), t ∈ T} un processus stochastique à temps continu et à espace discret. {X(t), t ∈ T} est un processus markovien si :

$$\forall\ t=(t_1, \ldots, t_n, t_{n+1}) \text{ tels que } t_1 < t_2 < \ldots < t_n < t_{n+1}$$

alors

$$Pr\ [\ X(t_{n+1})=x_{n+1} / X(t_n)=x_n;\ X(t_{n-1})=x_{n-1}\ ;\ \ldots\ ;\ X(t_1)=x_1\ ] = Pr\ [\ X(t_{n+1})=x_{n+1} / X(t_n)=x_n\ ]$$

Ainsi, un processus markovien est un processus sans mémoire. La connaissance de l'état aux instants $t_1 < t_2 < \ldots < t_n < t_{n+1}$ est une information entièrement contenue dans la connaissance de l'état à l'instant t_{n+1}. Autrement dit, l'évolution future du processus ne dépend que de l'état à l'instant présent et non de l'évolution passée.

6.3.3. Vecteur de distribution des états

Soit $\Pr [X(t)=x_i]$ $t \in T$, $i \in [1,n]$ la probabilité de présence du système dans l'état x_i à l'instant t. Posons $\Pr [X(t)=x_i] = P_i(t)$.

On définit le vecteur de probabilité des états ou de distribution par :

$$\Pi(t) = [\ P_1(t)\ \ P_2(t)\ \ \ldots \ldots\ P_n(t)\ ]$$

Ce vecteur fournit la probabilité d'être à un instant t dans chacun des états. La connaissance de $\Pi(t)$ permet d'estimer la présence du système dans les différents états de marche ou de panne pour les études de sûreté de fonctionnement. Ce vecteur possède les propriétés suivantes :

$$0 \le P_i(t) \le 1 \qquad \forall\ t \in T \qquad \text{et} \qquad i \in [1,n]$$

et

$$\sum_{i=1}^{n} P_i(t) = 1 \qquad \forall\ t \in T$$

(système complet d'événements).

On définit également le vecteur de probabilité initial $\Pi(0)$ à l'instant t=0 :

$$\Pi^0 = \Pi(0) = [\ P_1(0)\ \ P_2(0)\ \ \ldots \ldots\ P_n(0)\ ]$$

et le vecteur de distribution limite:

$$\Pi^\infty = \lim_{t\to\infty} \Pi(t) = \Pi(\infty) = [\ P_1(\infty)\ \ P_2(\infty)\ \ \ldots \ldots\ P_n(\infty)\ ]$$

Ce dernier caractérise le comportement asymptotique du processus de Markov après franchissement d'un nombre infini de transitions.

6.3.4. Probabilité de transition et taux de transition

On définit le taux de transition (ou intensité de transition) $\lambda_{ij}(t)$ d'un état x_i à un état x_j pour un processus de Markov par la relation suivante :

$$\lambda_{ij}(t) = \lim_{dt\to 0} \Pr[X(t+dt) = x_j / X(t) = x_i]$$

Les taux de transition dépendent en général du temps. Lorsqu'ils sont indépendants du temps (taux constants), le processus de Markov est dit **homogène**.

Dans le cas contraire, le processus de Markov est **non homogène**. Lorsqu'il existe un ou plusieurs taux de transition dont la valeur dépend du temps écoulé dans l'état en amont, le processus est qualifié de **semi-markovien**.

A partir de la relation précédente, on peut en déduire la probabilité de transition $p_{ij}(dt)$ de l'état x_i à l'état x_j pendant dt :

$$p_{ij}(dt) = \int_0^{dt} \lambda_{ij}(u).du$$

Dans le cas où le processus de Markov est homogène, la probabilité de transition pendant dt devient :

$$p_{ij}(dt)=\lambda_{ij}.dt$$

6.3.5. Mise en équation

Considérons une chaîne de Markov finie (nombre d'états fini) {1, 2, 3, …, n}. Nous allons la représenter matriciellement en associant un système d'équations au processus. La résolution de ce dernier donnera le vecteur de distribution Π correspondant à l'ensemble des probabilités d'être dans chaque état x_i.

La probabilité $P_i(t+dt)$ que le système soit dans l'état x_i à l'instant t+dt est égale à :

$P_i(t+dt)$ = Pr[système est dans x_i à l'instant t et y reste pendant dt]
+ Pr[système est dans un état $x_j \neq x_i$ à l'instant t et dans l'état x_i à l'instant t+dt]

$$P_i(t+dt) = P_i(t).\left[1 - \sum_{j\neq i} p_{ij}(dt)\right] + \sum_{j\neq i} P_j(t).p_{ji}(dt)$$

$$= P_i(t).\left[1 - \sum_{j\neq i} \int_0^{dt} \lambda_{ij}(u).du\right] + \sum_{j\neq i} P_j(t).\int_0^{dt} \lambda_{ji}(u).du$$

Supposons que $p_{ij}(dt)$ est suffisamment de fois dérivables sur dt. En appliquant un développement de Taylor-Young de $p_{ji}(dt)$ au voisinage de 0, il vient :

$$p_{ij}(dt) = \int_0^{dt} \lambda_{ij}(u).du = p_{ij}(0) + \sum_{k=1}^{r} p_{ij}^{(k)}(dt)\frac{dt^k}{k!} + o\left(dt^k\right)$$

$$= \lambda_{ij}(dt).dt + o(dt)$$

Si on se limite au premier ordre k=1. On obtient alors la relation :

$$\frac{P_i(t+dt) - P_i(t)}{dt} = -P_i(t).\sum_{j\neq i} \lambda_{ij}(dt) + \sum_{j\neq i} P_j(t).\lambda_{ji}(dt) + \frac{o(dt)}{dt}$$

Dans le cas d'un processus markovien homogène, les taux de transitions sont constants et les $P_i(t)$ sont différentiables. En passant à la limite, l'équation devient :

$$\left[\dot{P}_1(t) \quad \dot{P}_2(t) \quad \ldots \quad \dot{P}_n(t)\right] = \left[P_1(t) \quad P_2(t) \quad \ldots \quad P_n(t)\right].A$$

Soit : $\boxed{\dot{\Pi}(t) = \Pi(t).A}$ [3]

Cette relation constitue l'équation d'état du processus ou équation de **Chapman-Kolmogorov**. Sa résolution donne l'ensemble des probabilités d'occupation de chaque état.

A est la **matrice de transition**, avec : $\begin{cases} a_{ij} = \lambda_{ij} & \text{pour } i \neq j \\ a_{ii} = -\sum_{j \neq i} a_{ij} & \text{sinon} \end{cases}$

Le système d'équations décrit les distributions des probabilités et sa résolution fournit notamment les régimes transitoire et permanent. Dans le cas d'un processus homogène, la solution s'exprime sous la forme suivante :

$$\Pi(t) = \Pi^0 . e^{At}$$

Ce système d'équations différentielles du premier ordre est ensuite résolu par les méthodes classiques : transformée de Laplace, discrétisation, calcul des valeurs propres de A, Runge-Kutta [PAG80].

6.3.6. Graphe associé

Un atout de l'utilisation des chaînes de Markov homogènes est la possibilité de les représenter graphiquement en associant au processus un graphe d'état. L'application de la théorie des graphes permet en particulier d'évaluer qualitativement le comportement du processus en identifiant les classes d'appartenance des états : classes transitoires et classes finales qui seront définies ultérieurement.

Un processus de Markov est représentable graphiquement par un modèle à états-transitions appelé **graphe de Markov**. C'est un graphe orienté G=(E,Λ) où E représente les sommets du graphe associés aux états du processus et Λ les arcs orientés reliant les sommets. Les arcs sont étiquetés par un taux de transition a_{ij} reliant le sommet x_i au sommet x_j. Généralement, les termes diagonaux a_{ii} ne sont pas représentés.

Exemple :

Soit la matrice de transition $A = \begin{pmatrix} -4 & 1 & 3 & 0 & 0 & 0 \\ 0 & -15 & 5 & 10 & 0 & 0 \\ 2 & 0 & -3 & 0 & 1 & 0 \\ 0 & 0 & 0 & -2 & 2 & 0 \\ 0 & 0 & 0 & 4 & -7 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

L'équivalence graphique est illustrée sur la figure suivante :

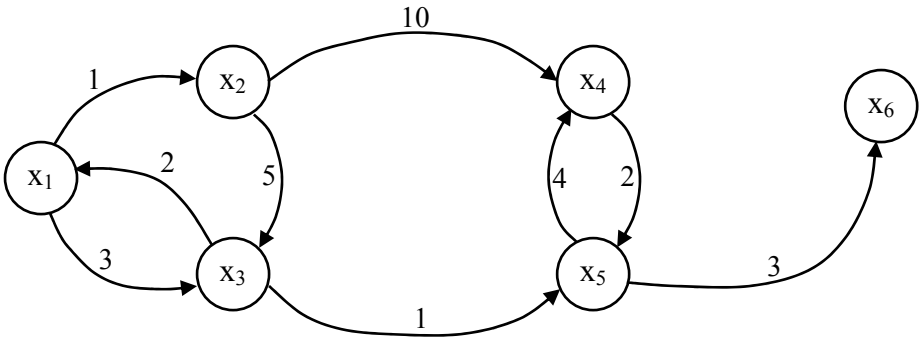


Figure 26 : Graphe de Markov équivalent à la matrice de transition A

La représentation d'une chaîne de Markov sous la forme d'un graphe s'avérera particulièrement intéressante pour les études de sûreté de fonctionnement ; le concepteur a la possibilité de visualiser l'ensemble des modes opératoires (nominaux, dégradés) et les états redoutés du système étudié, ainsi que l'ensemble des taux de défaillance (transitions) des composants, améliorant de ce fait la compréhension globale du comportement et l'évolution du système en présence de fautes.

6.3.7. Classes d'états

Quelques rappels sur la théorie des graphes

L'application de la théorie des graphes aux processus de Markov homogènes permettra de déduire un certain nombre de propriétés sur le comportement asymptotique du système. Nous commençons par rappeler quelques définitions relatives à la théorie des graphes.

Considérons un graphe de Markov G(E, Λ) sous-jacent à un processus de Markov homogène.

> *Définition : Classe d'états [BER98]*
> *Une classe d'états (ou « classe ») est un sous-ensemble non vide de l'ensemble des états E.*

> *Définition : Accessibilité d'un état*
> *On dit qu'un état j est accessible à partir d'un état i, s'il y a une probabilité strictement positive d'atteindre l'état j à partir de l'état i en un nombre fini de transitions.*

En terme de graphe, cette notion correspond à l'existence d'un chemin entre les sommets i et j.

> *Définition : Communication entre états*
> *Si l'état j est accessible à partir de l'état i et si, réciproquement, l'état i est accessible à partir de l'état j, on dit que ces deux états i et j communiquent.*

En termes de graphe, cette notion correspond à l'existence d'un circuit entre les sommets i et j.

Soulignons également que l'existence d'un circuit entre deux états est une relation d'équivalence notée $i\mathcal{R}j$. La figure suivante illustre le cas d'une classe d'états communicants :

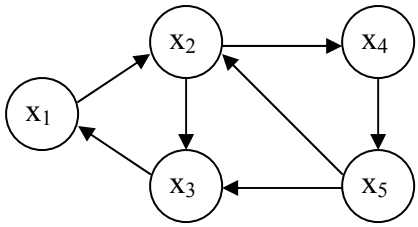


Figure 27 : Classe d'états communicants

> *Définition : Chaîne de Markov irréductible*
> *Soit E l'espace des états d'une chaîne de Markov finie. Elle est irréductible si un état j du système peut être atteint à partir de n'importe quel état i et ce, pour tout* $i,j \in E$.

Autrement dit, toute paire (i,j) d'états communiquent entre eux. D'un point de vue structurel, le graphe est fortement connexe.

> *Définition : Graphe fortement connexe*
> *Un graphe (orienté) est dit fortement connexe si pour tout couple de sommet (i,j), il existe une chaîne joignant i et j. On a alors* $\forall(i,j) \in E^2,\ i\mathcal{R}j$.

Le graphe de la figure 27 un exemple de graphe fortement connexe.
Cependant, la relation de forte connexité peut se limiter à une partie du graphe. On définit alors la notion de *composante fortement connexe* :

> *Définition : Composante fortement connexe*
> *C'est un ensemble de nœuds qui ont deux à deux la relation de forte connexité. Tout nœud en dehors de la composante n'a pas de relation de connexité avec aucun des éléments de la composante.*

Ainsi, un graphe est connexe s'il ne possède qu'une seule composante connexe. Les composantes fortement connexes constituent des **classes d'équivalence**. Leur identification sera très utile par la suite pour l'étude des différents modes de fonctionnement d'un système, connaissant son graphe de Markov sous-jacent et en vue d'une réduction de sa taille.

> *Définition : Classe transitoire d'une chaîne de Markov*
> *Une classe d'états (communicants) est dite transitoire si, après un nombre fini de transitions, le système sort de la classe et n'y retourne plus. Cette classe est composée uniquement d'états transitoires.*

> *Définition : Classe finale ou permanente*
> *Une classe d'états (communicants) est dite finale si, lorsque le système est dans un de ces états, il reste indéfiniment dans la classe.*

Les classes finales sont également qualifiées de récurrentes.
L'exemple suivant illustre le cas d'un graphe composé de deux classes d'équivalence, deux transitoires et une finale (composée d'un seul état) :

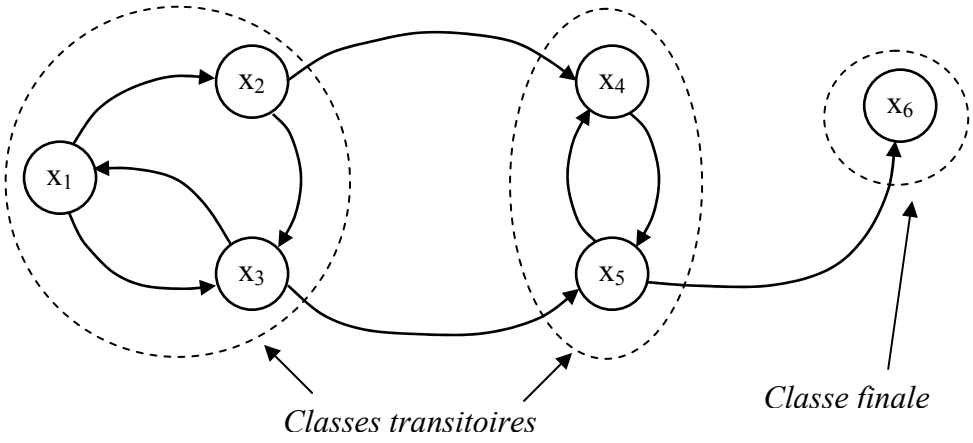


Figure 28 : Classes finale et transitoires

Une chaîne de Markov composée d'une seule classe finale est une chaîne de Markov irréductible.
Si une classe finale est composée d'un seul état, alors cet état est qualifié **d'absorbant**.
Ainsi, le graphe de Markov de la figure 28 contient deux classes transitoires $\{x_1, x_2, x_3\}$ et $\{x_4, x_5\}$ et un état absorbant $\{x_6\}$.

A partir de ces définitions, la notion de **graphe réduit** est introduite. Celle-ci est formée uniquement d'une partition des états en classes. La méthode de réduction qui sera décrite par la suite sera basée sur la recherche des classes d'équivalence et des transitions entre ces classes. Le concept de **Macro-Etat** désignera la réunion d'un ensemble d'états élémentaires communicants. D'un point de vue structurel, ces états forment une composante connexe ; d'un point de vue fonctionnel, cette connexité sera relative à l'existence de propriété de vivacité du système dans un mode de fonctionnement donné. Ainsi, la restriction du graphe aux seuls ensembles fortement connexes fournira une représentation du comportement du système très allégée d'un point de vue graphique. Cette vue réduite n'a bien sûr une véritable utilité qu'à la condition où un sens physique découle de cette représentation.

6.3.8. Graphes de Markov et distribution limite

> *Définition : Ergodicité*
> *Une chaîne Markov est ergodique, si dans son comportement asymptotique, les probabilités d'état de la chaîne tendent vers une distribution limite unique, indépendante des conditions initiales.*

Propriété : Loi stable unique [BER98]
Soit une chaîne de Markov à nombre fini d'états, formée d'une seule classe finale plus éventuellement d'une classe transitoire. Cette chaîne admet une distribution limite unique et indépendante des conditions initiales (chaîne de Markov ergodique).

Ainsi les propriétés structurelles d'une chaîne de Markov permettent d'évaluer très simplement les valeurs asymptotiques des probabilités d'états. Par exemple, dans le cas de la

chaîne de Markov de la figure 28, la distribution stationnaire est obtenue en résolvant le système d'équations linéaires du premier ordre suivant :

$$\lim_{t\to\infty} \dot{\Pi}(t) = 0 \;\Leftrightarrow\; \Pi^{\infty} \times A = 0$$

Avec :
$$\sum_{i=1}^{6} P_i(\infty) = 1$$

Nous supposons désormais que les chaînes de Markov traitées sont finies. Il existera donc toujours au moins une distribution stationnaire.

<u>Théorème</u> : Critère d'unicité
Une chaîne de Markov finie admet une distribution unique stationnaire si et seulement si elle comprend une seule classe récurrente.

Dans le cas où cette classe unique récurrente englobe l'ensemble des états de la chaîne de Markov ou encore si tous les états forment une composante fortement connexe, alors la chaîne de Markov est dite irréductible.

L'ensemble des notions qui ont été rappelées dans cette partie sera exploité ultérieurement dans un processus de simplification des graphes de Markov. Notamment, la recherche des classes d'états permettra d'identifier tous des états ayant des caractéristiques physiques communes : appartenance à un mode de fonctionnement nominal ou dégradé commun. Le principe général de la méthode consistant à raisonner non pas sur un graphe de Markov exhaustif très lourd visuellement (dans le meilleur cas) mais sur des ensembles d'états qui permettra, d'une part, d'alléger considérablement la taille du graphe (meilleure lisibilité, évitement de l'explosion combinatoire) et, d'autre part, de fournir au concepteur un modèle beaucoup plus compréhensible sur les modes de fonctionnement et le comportement du système en présence de défaillances. Ce modèle constitue un niveau d'abstraction différent du système, exploité à des fins d'analyse de sûreté de fonctionnement. Une fois établi, ce modèle fournit toutes les grandeurs caractéristiques recherchées de la sûreté de fonctionnement en transposant le graphe en un système d'équations, puis en le résolvant avec les méthodes d'analyse classiques.

Nous allons à présent rappeler quelques concepts mathématiques de la sûreté de fonctionnement.

6.4. Processus markoviens homogènes

6.4.1. Introduction

L'hypothèse markovienne, quand elle est vérifiée, est très séduisante de par sa simplicité conceptuelle et la facilité de résolution par des méthodes classiques. Lorsque les taux de défaillance sont constants (hypothèse d'homogénéité), le système d'équations de Chapman

Kolmogorov se ramène à un système linéaire du premier ordre. La résolution nécessite simplement de connaître la matrice de transition et les conditions initiales.

En supposant que le taux de défaillance λ d'une entité soit constant, la fiabilité devient d'après [2] :

$$R(t)=\exp(-\lambda t)$$

Ainsi les temps de séjour du système dans n'importe quel état suivent une distribution exponentielle négative, le paramètre de la densité de probabilité étant λ. Cette caractéristique mathématique traduit l'hypothèse markovienne selon laquelle l'évolution future de l'entité ou du système ne dépend que de l'état présent.

6.4.2. Domaine de validité des processus markoviens homogènes

Rappelons que l'utilisation de taux de défaillance constants doit être justifiée a priori. Des études statistiques ont montré que de nombreux composants ont un taux suivant une courbe « en baignoire » :

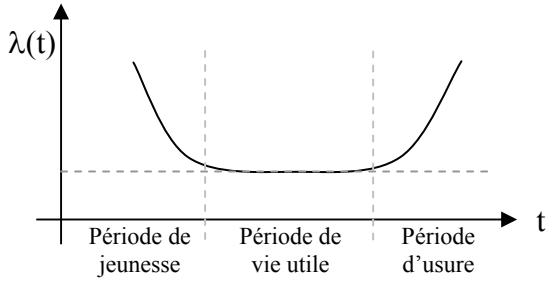


Figure 29 : Evolution du taux de défaillance de composants dans le temps

La courbe présente trois régions distinctes :

- la période de jeunesse caractérise le début de vie d'une entité où le taux de défaillance décroît rapidement jusqu'à atteindre un niveau constant.
- pendant la période de vie utile, le taux de défaillance reste constant ; les défaillances survenant pendant cette période sont dites accidentelles.
- pendant la période d'usure, le taux de défaillance croît avec le temps ; les défaillances sont dues à un processus de vieillissement : détérioration, corrosion…

Particulièrement, les composants électroniques ont une évolution de leur taux de défaillance obéissant à cette loi. Cependant, les études quantitatives sont effectuées généralement en supposant que les composants se trouvent dans leur période de vie utile, pour plusieurs raisons ; d'abord, par commodité car l'hypothèse des taux de défaillance constants permet d'exploiter les processus de Markov homogènes et donc de supposer que les durées de séjour suivent des distributions exponentielles. La mise en équation du système et sa résolution sont très simples. Ensuite pour des raisons pratiques : dans de nombreux domaines, les données récoltées sur les défaillances sont insuffisantes pour permettre la vérification de lois différentes. Les valeurs utilisées pour les études sont basées sur le calcul d'estimateur et de

l'intervalle de confiance. Ainsi, l'estimateur d'un taux de défaillance λ est calculé de la façon suivante, pour une population donnée de composants de même nature :

$$\lambda = \frac{N_f}{T_f} = \frac{\text{nombre de défaillances en fonctionnement}}{\text{durée cumulée de fonctionnement}}$$

L'hypothèse des λ constants permet d'avancer que le nombre de défaillances est distribué selon une loi de poisson. L'intervalle de confiance associé à l'estimateur peut alors être déduit des observations [VIL88].

Par contre, les taux de défaillance de composants mécaniques et électro-mécaniques sont rarement constants. La modélisation markovienne ne peut pas être appliquée ; l'existence d'une période de vie utile pour ce type de composant implique l'élimination de la période de jeunesse en vieillissant préalablement les éléments (par exemple dans le domaine spatial) et en assurant une maintenance préventive afin de remplacer les éléments d'usure avant défaillance. Néanmoins, la loi de Weibull est beaucoup plus réaliste dans ce cas de figure car un grand nombre de distributions expérimentales peuvent être représentées. La distribution de Weibull est une généralisation de la distribution exponentielle.
Le taux de défaillance $\lambda(t)$ dépend de trois paramètres :

$$\lambda(t) = \frac{\beta(t-\gamma)^{\beta-1}}{\sigma^{\beta}} \qquad \text{avec } \beta>0,\ \sigma>0,\ t>\gamma$$

γ est le paramètre de position, homogène à un temps. Il est en général nul. Le paramètre β est sans dimension et σ est homogène à un temps. L'intérêt de cette distribution est de pouvoir obtenir des formes de courbes de probabilité très différentes en fonction de la valeur du paramètre β.
Notamment :

- pour $\beta<1$, le taux de défaillance est décroissant dans le temps,
- pour $\beta>1$, le taux de défaillance est croissant dans le temps ; pour β voisin de 2, la densité de défaillance est proche d'une loi log-normale et pratiquement une gaussienne pour $\beta>3,4$,
- pour β voisin de 1, $\gamma=0$, la loi s'apparente à une exponentielle avec $\sigma=1/\lambda$.

Cependant, pour limiter la complexité des études quantitatives, en considérant que les entités défaillantes se trouvent dans la période de vie utile, l'hypothèse de distribution exponentielle sera utilisée pour modéliser les temps de séjour moyens dans les états. En effet les composants capteurs, calculateurs intégrés dans les architectures matérielles embarquées admettent des caractéristiques de durée de fonctionnement exponentielle. Il est néanmoins possible de modéliser des taux de défaillance croissants relatifs par exemple à un phénomène d'usure en utilisant exclusivement des distributions exponentielles.

6.4.3. Mise en équation d'un graphe de Markov

Détaillons sur un petit exemple le principe de construction et la structure de la matrice de transition associée à un graphe de Markov.

La mise en œuvre d'une méthode analytique pour quantifier des grandeurs comprend les phases suivantes :

- Le recensement de tous les états du système : ces états sont propres à des états de fonctionnement, des états de fonctionnement dégradés par exemple après une reconfiguration logicielle suite l'apparition et à la détection de défaillances ou soit encore à des états de dysfonctionnement dans lesquels le système n'est plus en mesure d'assurer le service attendu. Une attention particulière doit être portée sur l'exhaustivité du recensement de ces états, car une omission peut être préjudiciable quant à la qualité des estimateurs attendus. Un graphe de Markov peut être construit à partir d'une réflexion du fiabiliste en exploitant un modèle détaillé issu de spécifications fonctionnelles et/ou matérielles qui décrivent le comportement du système et les effets des défaillances sur celui-ci. Bien entendu, cette approche n'est envisageable que dans la mesure où le nombre d'états recensés reste modeste. La complexité et la taille des systèmes issus du monde industriel permettent difficilement d'exploiter les graphes de Markov à partir d'une construction « à la main ». Une construction automatique est également possible à partir d'une modélisation du système dans le formalisme des réseaux de Petri stochastiques [DAV89].

- Le recensement de toutes les transitions du graphe : notons qu'un graphe de Markov homogène ne comprend que des taux de transition constants relatifs à des distributions exponentielles. En général, pour un système non réparable, il s'agit de taux de défaillance faisant passer le système d'un état de fonctionnement à un état dégradé ou de panne. Cependant, dans un graphe d'état au sens large du terme, la plupart des transitions sont déterministes et correspondent à un fonctionnement normal du système. Par exemple, dans le cas où le système traité est une architecture fonctionnelle de contrôle-commande, les transitions peuvent être franchies sur occurrence d'événements internes propres au système (top d'horloge, franchissement de seuil de variables continues, échange de messages relatifs à des communications entre sous-fonctions…). Ces différents franchissements caractérisent la dynamique rapide du système par opposition à la dynamique des pannes. Nous verrons ultérieurement comment elle s'intègre dans le modèle markovien et dans quelle mesure est impactée la fiabilité. Bien entendu, un graphe de Markov n'est qu'une abstraction d'une modélisation détaillée dans laquelle n'apparaissent que des séquences de franchissements définies par des successions de défaillances. La construction d'un modèle dysfonctionnel résumant le comportement du système en présence de pannes constitue donc un travail d'analyse issue d'une bonne compréhension du système, tant au niveau fonctionnel que dysfonctionnel.

- L'obtention des grandeurs de sûreté de fonctionnement à partir du modèle ainsi construit. Une résolution d'un système d'équations du premier ordre sous-jacent au graphe de Markov construit est suffisante pour estimer l'évolution de la fiabilité, disponibilité et les grandeurs moyennes MTTF, MTBF, MTTR…

Illustrons à présent comment, à partir d'un graphe de Markov, ces grandeurs peuvent être calculées :

Considérons un système S composé de N états $\{x_1, x_2, \ldots, x_N\}$ appartenant à l'ensemble discret E. Nous allons supposer que ces états élémentaires correspondent à des modes de fonctionnement nominaux, dégradés ou de panne suivant le nombre de défaillances présentes au sein du système.
Soit n le nombre de modes de fonctionnement et N_i $i \in [1, n]$ le nombre d'états élémentaires caractérisant chaque mode de fonctionnement i.

On a donc : $N = \sum_{i=1}^{n} N_i$.

Soit M_i $i \in [1, n]$ les ensembles d'états relatifs à chaque mode de fonctionnement i, tels que :

$E = \bigcup_{i \in [1,n]} M_i$.

Réordonnons les états de la façon suivante :

- $M_1 \equiv \{x_1, x_2, ..., x_{N_1}\}$ les N_1 premiers états correspondent à un fonctionnement nominal du système, en l'absence de défaillances,

- $M_2 \equiv \{x_{N_1+1}, x_{N_1+2}, ..., x_{N_1+N_2}\}$ un mode de fonctionnement dégradé. Cet ensemble d'états forme un ensemble d'états transitoires,

- M_3 à M_{n-1} d'autres ensembles transitoires, relatifs également à des modes de fonctionnement dégradés ; par définition, le système arrive dans l'un de ces modes sur occurrence d'une ou de plusieurs défaillance(s) de composants. Nous les ordonnons de la façon suivante : en considérant deux modes M_i et M_j avec $i<j$, M_j sera supposé être « plus dégradé » que M_i. Ainsi le passage du système du mode M_i à M_j pourra se produire sur occurrence d'une ou de plusieurs défaillances. A l'inverse, le système ne pourra revenir à un mode de fonctionnement M_i à partir de M_j qu'à condition de réparer ou de remplacer une ou plusieurs entités défaillantes.

- $M_n \equiv \{x_{N_{n-1}+1}, x_{N_{n-1}+2}, ..., x_{N_{n-1}+N_n}\}$ le dernier ensemble d'états relatifs à des états de panne du système.

A partir de cette réorganisation, la matrice de transition associée au graphe de Markov prend une forme particulière décrite sur la figure 30 suivante :

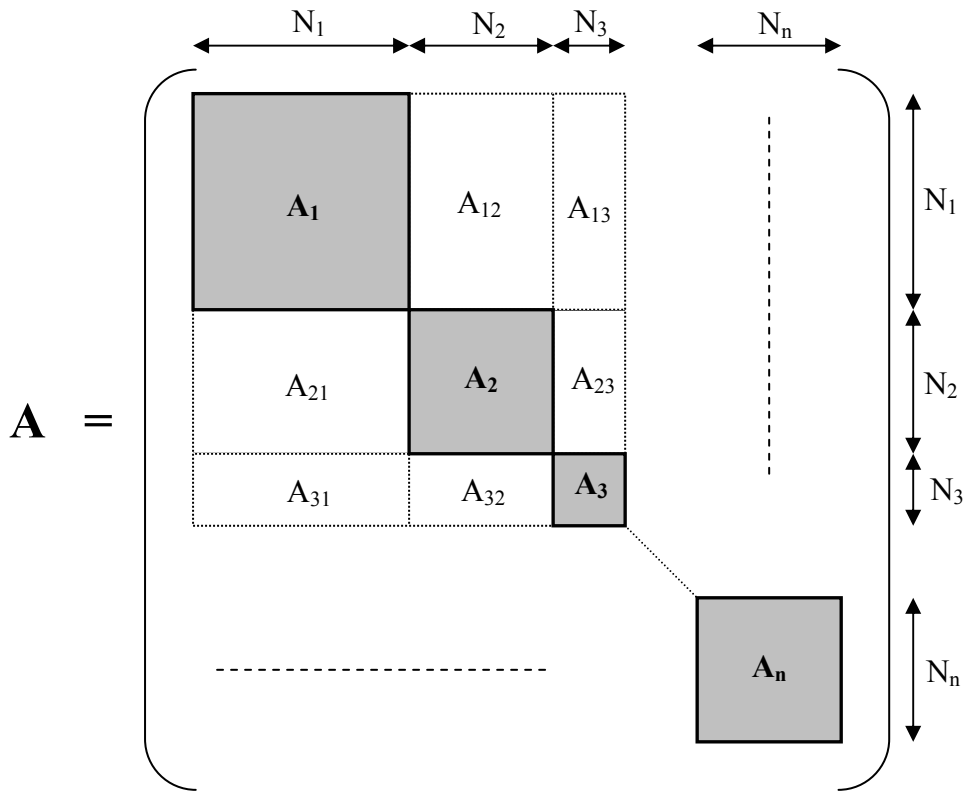


Figure 30 : Découpage de la matrice A et regroupement des états selon les modes de fonctionnement

Les blocs A_i, $i \in [1, n]$, sont des matrices carrées de taille N_i x N_i. Chacun des blocs A_i est composé des états du mode de fonctionnement M_i. Ce sont précisément ces blocs qui permettent d'évaluer la probabilité de présence du système dans chaque mode. En d'autres termes, ils serviront à calculer les grandeurs de sûreté recherchées. Pour l'instant, la structure interne de ces blocs n'est pas précisée, c'est-à-dire les transitions internes entre chaque état élémentaire.

Les blocs A_{ij} où $(i,j) \in [1, n]^2$, de taille N_i x N_j, représentent les transitions entre des états du bloc A_i et les états du bloc A_j. Physiquement, ces transitions traduisent le passage du système d'un mode de fonctionnement nominal ou dégradé à un mode de fonctionnement dégradé ou de panne. Les sous-matrices A_{ij} sont donc formées de taux de défaillance λ si $i<j$ ou de réparation μ si $i>j$ d'après le partitionnement défini précédemment.

En supposant maintenant que le système n'est pas réparable, la matrice A possède la structure suivante :

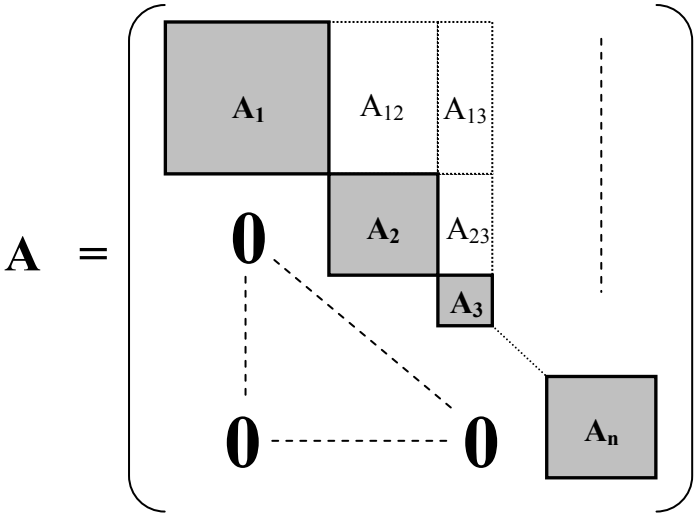


Figure 31 : Structure de la matrice de transition dans le cas d'un système non réparable

Définition : Matrice réductible [CAL97]
Une matrice $A=(a_{ij}) \in \mathbb{R}^{NxN}$ est réductible s'il existe une matrice de permutation $P \in \mathbb{R}^{NxN}$ telle que :

$$PAP^T = \begin{bmatrix} M & Q \\ 0 & T \end{bmatrix}$$

*où M et T sont deux sous-matrices carrées. S'il n'existe pas de matrice de permutation P satisfaisant l'équation précédente, alors A est **irréductible** ou **non décomposable**.*

Ainsi la matrice A définie en figure 31 est clairement réductible. Une interprétation de cette propriété consiste à dire que le graphe d'états se décompose en états transitoires et finaux. Ces états transitoires forment des ensembles identifiés comme étant des modes de fonctionnement M_i pour i différent de n. D'un point de vue comportemental, le système arrivera inévitablement au bout d'un certain temps dans le dernier ensemble d'états absorbant M_n.
En regroupant les états appartenant à un mode de fonctionnement M_i commun, la chaîne de Markov pourra par exemple prendre l'allure suivante :

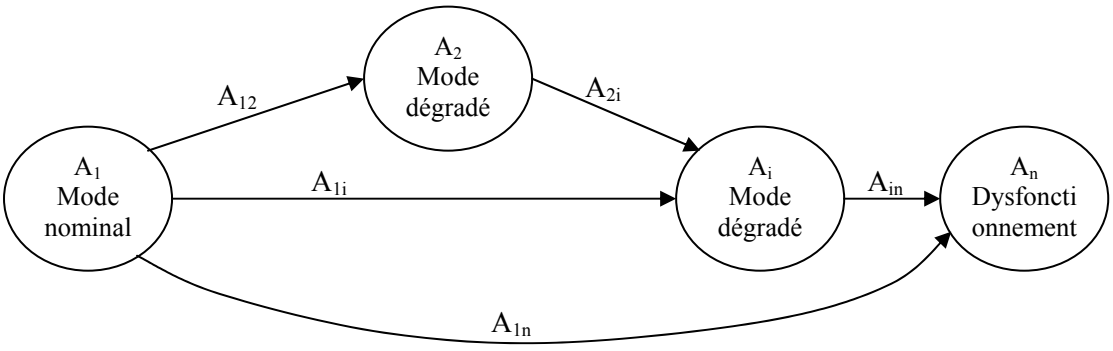


Figure 32 : Structure globale de la chaîne de Markov associée à la matrice A

Les arcs étiquetés par les matrices A_{ij} symbolisent l'ensemble des transitions entre les états de M_i vers les états de M_j, c'est-à-dire le flux total partant des états de M_i vers les états de M_j.

La propriété d'irréductibilité d'une chaîne de Markov finie homogène se traduit structurellement sur la matrice de transition qui est elle même irréductible [CAL97].

Le graphe de la figure 32 est une forme condensée du graphe d'états puisque seuls des regroupements d'états élémentaires apparaissent, tout en faisant abstraction des liens entre les états d'un même ensemble.
Cette technique, dite d'agrégation, a été abondamment traitée dans la littérature. L'objectif est principalement de ramener la résolution d'un problème complexe à un problème plus abordable. Elle vise notamment à éviter l'explosion combinatoire en réduisant le nombre de nœuds du graphe.
Bien que le graphe d'état agrégé fournisse une représentation allégée du graphe d'origine, des approximations sont réalisées sur les équations de Chapman-Kolmogorov sous-jacentes au graphe réduit et il est donc essentiel que les estimateurs calculés (en l'occurrence les grandeurs de sûreté de fonctionnement) restent proches de ceux qui seraient issus d'un traitement des équations initiales. Un grand nombre de techniques de réduction vise à faciliter la recherche d'une solution stationnaire ([CAL97]) de systèmes markoviens n'ayant pas d'états absorbants. L'analyse du régime permanent reste insuffisant puisque dans le contexte d'évaluation quantitative de systèmes non réparables et donc ayant des états absorbants, la disponibilité/fiabilité tendra obligatoirement vers zéro. Les valeurs asymptotiques ne permettent donc pas de comparer sur le plan de la sûreté la robustesse de plusieurs solutions d'architectures fonctionnelles/matérielles.
De plus, la connaissance des courbes de probabilités sur la période étudiée permet également d'estimer l'évolution d'un taux de défaillance global du système d'après la relation [1].

6.4.4. Calcul des grandeurs de sûreté

Soit un système S dont la matrice de transition est ordonnée comme celle donnée en figure 31. Dans le cas des systèmes non réparables, la disponibilité et la fiabilité sont égales. Pour cette raison le système sera supposé réparable afin de différencier les deux grandeurs. Les matrices A_{ij} $i>j$ sont donc non toutes nulles, en particulier A_{nj} car dans le cas contraire le dernier ensemble d'états M_n est absorbant (impossibilité de quitter ce mode).
Toutes les grandeurs sont estimées à partir de la connaissance des probabilités issues de la résolution des équations différentielles.

Disponibilité :

Nous avons vu que c'est la probabilité que le système S fonctionne à l'instant t. Lorsque la taille de l'espace d'état reste raisonnable, la disponibilité des systèmes markoviens découle directement de la résolution de la relation [3] et de la connaissance des états qui caractérisent un fonctionnement (dans le sens de la délivrance d'un service jugée acceptable). Cependant d'après la définition, la disponibilité est qualifiée comme étant une mesure binaire par rapport à un service délivré. La fonction est entièrement accomplie ou interrompue. Cependant, dans les systèmes mécatroniques considérés, la qualité du service peut se décliner en plusieurs modes d'accomplissement comprenant un certain nombre de modes dégradés. L'analyste doit alors porter un choix (subjectif) sur les états dégradés qui garantissent néanmoins une

continuité du service même en présence de défaillances (cas par exemple des reconfigurations matérielles…). D'autres états seront jugés trop dégradés pour être associés à la disponibilité. Ils devront être associés à des états de panne.

Sur l'ensemble des états E, on distingue alors ceux qui traduisent un dysfonctionnement de S. On suppose que M_n représente l'ensemble de ces états. Ceux appartenant à M_i avec $i \neq n$ correspondent soit à des modes nominaux, soit à des modes dégradés dans lesquels le système est supposé pouvoir assurer quand même un service minimal en présence de défaillance(s).
La disponibilité D(t) est alors égale à la somme des probabilités que le système se trouve dans l'un des modes M_i, $i<n$, soit,

$$D(t) = \Pr\left(\text{S fonctionne à l'instant t}\right) = \sum_{i,\, j<n} \Pr\left(X(t) = x_i,\, x_i \in M_j\right) = 1 - \sum_{i} \Pr\left(X(t) = x_i,\, x_i \in M_n\right)$$

$$= \sum_{i=1}^{N-N_n} P_i(t) = 1 - \sum_{i=N-N_n+1}^{N} P_i(t)$$

Ou en regroupant les probabilités $P_i(t)$ appartenant à un même mode M_j :

$$D(t) = \sum_{j=1}^{n-1} Q_j(t) = 1 - Q_n(t)$$

$$\text{Avec} \begin{cases} Q_1(t) = P_1(t) + P_2(t) + \ldots + P_{N_1}(t) \\ Q_2(t) = P_{N_1+1}(t) + P_{N_1+2}(t) + \ldots + P_{N_1+N_2}(t) \\ Q_3(t) = P_{N_1+N_2+1}(t) + P_{N_1+N_2+2}(t) + \ldots + P_{N_1+N_2+N_3}(t) \\ \ldots \\ Q_n(t) = P_{N_1+N_2+\ldots+N_{n-1}+1}(t) + P_{N_1+N_2+\ldots+N_{n-1}+2}(t) + \ldots + P_{N_1+N_2+\ldots+N_n}(t) \end{cases}$$

Cette forme d'écriture est particulièrement intéressante, puisqu'elle condense les équations différentielles à résoudre en faisant abstraction des probabilités élémentaires $P_i(t)$ potentiellement trop nombreuses et inutiles dans une étude macroscopique d'un système de contrôle-commande. Ainsi, le calcul de la disponibilité se ramène à une recherche de macro-variables à partir d'une résolution d'un système d'équations de taille modérée.

Fiabilité :

Nous avons vu que par définition la fiabilité est la probabilité pour que le système S soit non défaillant sur l'intervalle de temps [0, t]. L'évaluation de la fiabilité est analogue à celle de la disponibilité puisque c'est la probabilité que le système soit en état de marche à un instant t, mais sans être passé par un état de panne entre 0 et t. Il suffit de modifier la matrice de transition en supprimant tous les taux de transitions permettant au système de passer d'un état de panne à un état de marche (taux de réparation). La structure de la matrice est celle de la figure 31.

Maintenabilité :

Une étude de maintenabilité se traduit au niveau de la politique de maintenance (niveaux d'intervention, procédures de tests, de maintenance…). Les systèmes dont la maintenabilité

est étudiée étant sujets à des réparations (correctives ou préventives), et compte-tenu des hypothèses posées, nous n'entrerons pas davantage dans le détail du calcul de cette grandeur.

Evènements redoutés :

A partir de la connaissance des événements redoutés susceptibles d'apparaître au cours de la vie du système, une évaluation quantitative permettra d'estimer la fréquence d'apparition en termes de probabilité (ou dans l'unité ppm).
La nature et le nombre de ces événements redoutés sont issus d'une étude qualitative (analyse préliminaire des risques) et sont ensuite classés selon leur niveau de gravité.
Chaque événement redouté est ensuite formalisé en état ou classe d'états sur un modèle markovien. La mise en équation du graphe de Markov et sa résolution fournit alors les évolutions des probabilités d'être dans chacune des classes d'états redoutés.
La matrice de transition associée prend la forme suivante :

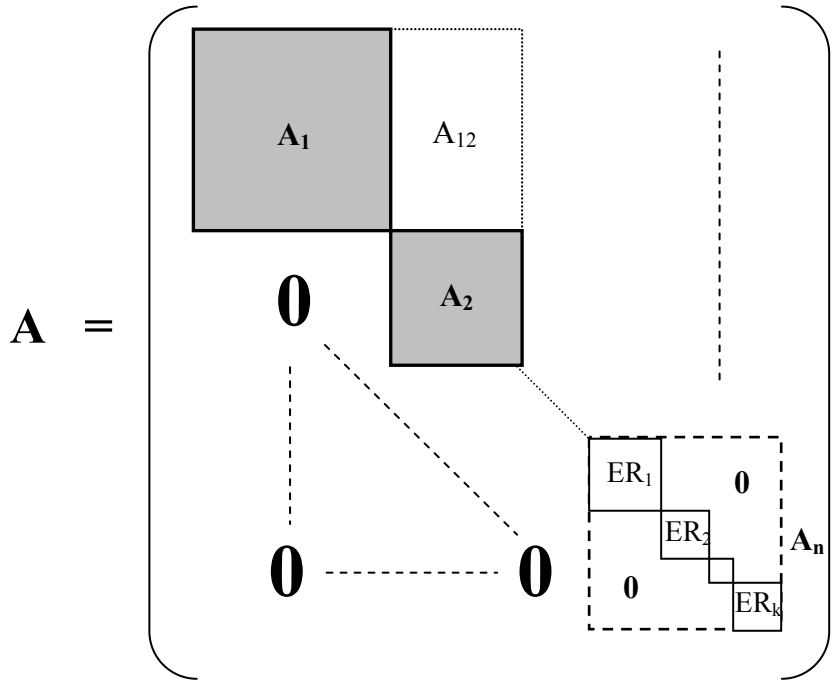


Figure 33 : Découpage de la matrice de transition en isolant les états redoutés

La dernière sous-matrice A_n s'écrit sous une forme bloc diagonale. Elle est découpée en autant de sous-blocs ER_i qu'il y de classes d'états redoutés, ici k.

Bien entendu, la distinction de chaque état redouté n'a d'intérêt dans une étude quantitative, que si l'on cherche à évaluer séparément les probabilités d'occurrence de chaque événement redouté, par exemple pour comparer les probabilités que le système arrive dans les états redoutés de disponibilité ou de sécurité. En effet, tous les états redoutés n'ont pas la même importance d'un point de vue de la gravité et la fréquence d'apparition de certains d'entre eux peut orienter le concepteur à mettre en place des barrières de sécurité spécifiquement pour diminuer criticité de ces défaillances potentiellement catastrophiques.
Par contre, un calcul global de la fiabilité n'impose pas un tel découpage.

6.4.5. Application à un exemple simple

L'exemple théorique suivant illustre le cas d'une chaîne de Markov d'un système possédant un mode de fonctionnement nominal MN, deux modes dégradés MD1 et MD2 et deux ensembles d'états redoutés ER1 et ER2.

Les états élémentaires sont regroupés selon leur appartenance à un même mode de fonctionnement :

- e_1 et e_2 appartiennent à Mode Nominal MN,
- e_3, e_4 et e_5 se rapportent au Mode Dégradé 1 MD1,
- e_6 est l'unique état du Mode Dégradé 2 MD2,
- e_7 et e_8 sont relatifs à l'Evènement Redouté 1 ER1,
- e_9 correspond à l'Evènement Redouté 2 ER2.

Les taux de transition entre états élémentaires d'un même mode de fonctionnement sont notés α_i. Ce ne sont pas des taux de défaillance mais ils sont associés au comportement du système en l'absence de défaillance dans chacun des modes. Ils reflètent donc la dynamique propre du système. Pour l'instant, nous ne détaillons pas davantage ces paramètres.

Le passage d'un mode de fonctionnement à un autre se fait sur occurrence d'une ou de plusieurs défaillance(s).

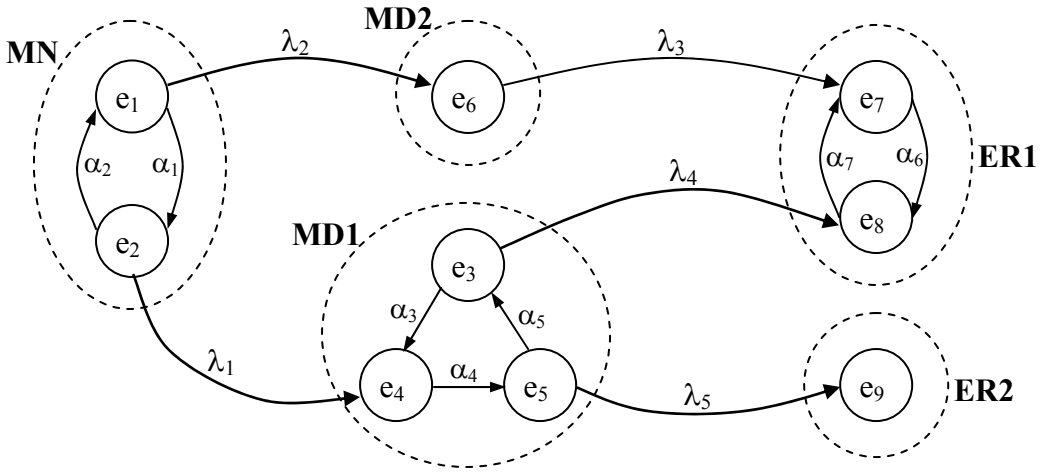


Figure 34 : Exemple de graphe de Markov d'un système à plusieurs modes de fonctionnement

La matrice de transition A sous-jacente à ce graphe possède la structure en sous-blocs définie plus haut, c'est-à-dire :

$$A = \begin{pmatrix} -(\alpha_1+\lambda_2) & \alpha_1 & 0 & 0 & 0 & \lambda_2 & 0 & 0 & 0 \\ \alpha_2 & -(\alpha_2+\lambda_1) & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -(\alpha_3+\lambda_4) & \alpha_3 & 0 & 0 & 0 & \lambda_4 & 0 \\ 0 & 0 & 0 & -\alpha_4 & \alpha_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_5 & 0 & -(\alpha_5+\lambda_5) & 0 & 0 & 0 & \lambda_5 \\ 0 & 0 & 0 & 0 & 0 & -\lambda_3 & \lambda_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\alpha_6 & \alpha_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha_7 & -\alpha_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 35 : Matrice de transition associée

Choisissons des valeurs arbitraires pour les α_i et les λ_i, avec néanmoins les conditions suivantes : $\alpha_i >> \lambda_j \quad \forall(i,j)$ pour refléter la différence de dynamique entre le système et l'occurrence des défaillances.
En supposant que le système se trouve en mode nominal (état e_1) à l'instant initial, la résolution de l'équation différentielle de Chapman-Kolmogorov fournit les probabilités d'avoir les événements redoutés ER1 et ER2, ainsi que la fiabilité ou disponibilité :

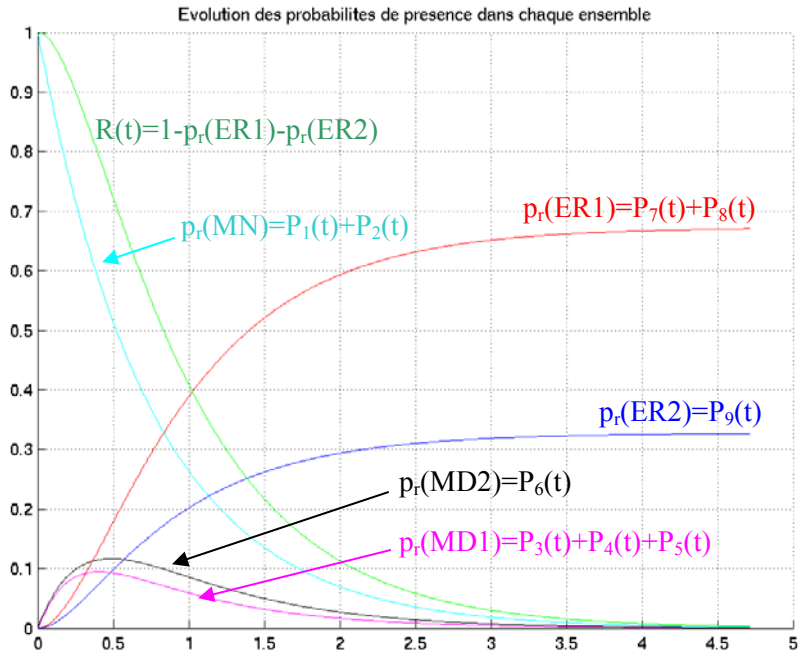


Figure 36 : Représentation de l'évolution des probabilités

L'analyste n'a d'intérêt dans cette démarche que la connaissance des probabilités de présence dans les modes de fonctionnement et de panne et non les probabilités élémentaires issues de la résolution de l'équation différentielle initiale. Cependant, dans le cas présent, les probabilités

globales relatives aux événements redoutés ou, plus généralement, à la présence dans les macro-états ont été obtenues en sommant des probabilités élémentaires.

La méthode d'agrégation qui sera proposée consistera précisément à calculer directement les « macro-probabilités » en résolvant un système d'équations différentielles de taille réduite, égale au nombre d'inconnues utiles au calcul des grandeurs de sûreté de fonctionnement, c'est-à-dire le nombre de modes de fonctionnement et d'événements redoutés recensés.
L'objectif ultime étant de pouvoir générer directement un graphe de Markov réduit, en évitant la phase de regroupement des états élémentaires en macro-états qui implique le dénombrement exhaustif de l'ensemble des états élémentaires et de toutes les transitions. Cette tâche est bien entendue illusoire dans le cas de systèmes complexes de taille industrielle.

Précisons que les taux de transitions α_i figurant sur l'exemple sont arbitrairement de nature stochastique. Cependant, en fonctionnement nominal ou dégradé, le comportement d'un système embarqué est **déterministe** et celui-ci n'évolue qu'en fonction d'informations provenant de l'extérieur (stimuli, échanges de messages inter-fonctions, top d'horloge…). Par exemple, l'alternance entre des états d'ouverture/fermeture d'un actionneur est commandée par un système de contrôle-commande qui envoie une consigne adéquate en fonction de mesures capteurs…. La transition d'un état à un autre n'a rien de stochastique. Ces taux de transitions ont simplement servi à illustrer la présence de deux dynamiques distinctes sur le processus de Markov, propriété qui sera à la base du processus d'agrégation.

7. Conclusion

Ce chapitre a été consacré à la description des systèmes de contrôle-commande. Face à la spécificité de ces systèmes, différents éléments méthodologiques permettant d'améliorer les différents aspects de la conception ont été relevés :

- choix du formalisme,
- méthodes de vérification et validation des modèles dès les premières phases de la spécification,
- choix d'une méthode de représentation et d'analyse de la sûreté de fonctionnement.

Il apparaît que les formalismes à états transitions conviennent le mieux pour traduire la nature discrète de ces systèmes. Couplés à un modèle continu, le comportement complexe de l'ensemble contrôle-commande+processus physique peut être analysé et simulé. Ce modèle hybride, caractérisé par un niveau de détail très fin, constitue la représentation du système de plus haut niveau dans la méthodologie proposée.

L'utilisation de formalisme basé sur les automates permet ensuite d'appliquer des méthodes de vérification/validation formelle comme le model-checking, afin de garantir la conformité de son fonctionnement par rapport à des exigences. La mise en œuvre de ces méthodes peut néanmoins impliquer de conditionner le modèle raffiné afin d'en déduire un modèle plus abstrait.

Enfin, l'utilisation de la méthode markovienne apparaît comme étant une solution efficace pour disposer d'un modèle dysfonctionnel d'un SDH pour évaluer les grandeurs de sûreté de fonctionnement. Cependant l'approche classique de cette méthode souffre de ces limites et reste difficilement applicable en tant que tel.

La méthode que nous proposons dans le chapitre 3 permettra :

- d'intégrer dans un modèle markovien un certain nombre de paramètres traduisant notamment l'interaction entre les aspects fonctionnels et les défaillances,
- de construire un graphe simplifié, sans générer un graphe de Markov exhaustif soumis à l'explosion combinatoire. Les états qui figureront sur le graphe seront facilement interprétables physiquement, puisqu'ils pourront correspondre, par exemple, à l'ensemble des modes opératoires du système.

Un aspect intéressant à développer ultérieurement, serait d'inclure tous ces éléments méthodologiques dans un outil déjà existant, supportant l'un des formalismes décrits, afin de faciliter et d'automatiser chaque étape de la conception (analyse fonctionnelle et dysfonctionnelle).

Nous allons à présent aborder la partie relative à la méthode d'agrégation des graphes de Markov ; les principaux concepts et outils mathématiques utilisés seront présentés. Nous détaillerons également des méthodes permettant d'évaluer les performances de l'agrégation en termes de temps de calcul et la pertinence des résultats.

CHAPITRE 3

UNE METHODE D'AGREGATION DES GRAPHES DE MARKOV

Chapitre 3

Une méthode d'agrégation des graphes de Markov

1. Introduction

1.1. Motivations

Dans de nombreuses situations, le maître d'œuvre chargé de l'étude de sûreté de fonctionnement doit construire un modèle représentatif du comportement du système en présence de défaillances ; ce modèle doit être suffisamment abstrait pour ne contenir que des éléments pertinents et essentiels à l'évaluation des performances recherchées. Ce modèle doit également être concis afin d'être facilement interprétable par d'autres acteurs participant à la conception et à la construction de la sécurité. Ainsi, les méthodes graphiques telles que les arbres de défaillance ont eu et ont toujours un succès dans le milieu industriel car la construction d'un modèle dysfonctionnel du système est relativement aisée et permet de visualiser instantanément les coupes minimales de défaillances conduisant à l'occurrence d'événements redoutés. Comme il a été remarqué précédemment, ce type de méthode n'offre qu'une vision statique des combinaisons de défaillances et, de ce fait, n'est pas adapté à des systèmes dits « événementiels » qui sont très orientés temps réel.
Les méthodes qualifiées de dynamiques viennent combler ce manque à condition d'offrir, à l'instar des arbres de défaillance, un support visuel donnant le comportement du système en présence de défaillances. Les graphes de Markov sont pourvus de ces avantages. La composante temporelle est implicitement présente dans la modélisation et les mécanismes de reconfiguration peuvent être naturellement intégrés dans le modèle. Les séquences ordonnées de pannes conduisant le système (non réparable) d'un état nominal à un état redouté sont également visualisables en extrayant les chemins entre les deux sommets considérés.

Le calcul de la « fiabilité dynamique » nécessite l'intégration dans le modèle des interactions entre les phénomènes de défaillance et le processus physique. L'ensemble des données décrivant ce dernier est potentiellement très grand (variables discrètes et continues). Une formulation mathématique rigoureuse de la fiabilité dynamique impliquerait de connaître

l'expression analytique des variables évoluant dans le temps, puis d'exprimer les grandeurs de sûreté de fonctionnement recherchées en fonction de toutes ces variables physiques. En d'autres termes, le taux de défaillance global (relatif à la fiabilité) d'un système varie constamment en fonction de l'état interne du système et de sa dynamique. Dans l'hypothèse d'une modélisation markovienne à taux constants, l'évolution de la valeur d'un taux de défaillance doit nécessairement se traduire par des changements d'état. Le nombre de transitions est potentiellement infini si l'on souhaite substituer un taux de défaillance évoluant continûment dans le temps par une succession discrète de changements d'états. La transcription mathématique du processus stochastique en un système d'équations différentielles devient impossible car sa dimension est infinie.

La construction d'un modèle markovien réduit fournissant des informations de type qualitatif (modes de fonctionnement, séquences de défaillance) est le premier objectif de la méthodologie proposée.

Dans le cadre d'une évaluation quantitative de sûreté de fonctionnement, nous souhaitons exploiter ce modèle afin de quantifier des mesures. Elles dépendent bien entendu des probabilités d'occurrence des pannes de composants élémentaires mais également de l'état du système à l'instant d'apparition des pannes.
Intuitivement, le temps de séjour dans un état ou un mode de fonctionnement du système a une influence sur les conséquences possibles d'une séquence de défaillances. A titre d'exemple, la défaillance d'un actionneur ou d'un calculateur embarqué a des répercussions différentes selon que le véhicule est à l'arrêt ou en roulage sur autoroute. Ces temps de séjour sont fonctions de la dynamique du système, de l'état de l'environnement et des informations échangées entre le système et l'environnement. Cette constatation n'est rien d'autre qu'un problème de représentation et de résolution de la « fiabilité dynamique ».
La difficulté réside en grande partie dans la recherche des variables influençant la dynamique des défaillances et de les intégrer dans le modèle « markovien » réduit.

Ainsi la modélisation et la résolution d'un problème de fiabilité dynamique est le deuxième objectif de la méthode proposée.

1.2. Principe de la méthode de simplification

Lorsque les systèmes de contrôle-commande à concevoir ont un nombre important de modes opératoires permettant de répondre au mieux à des exigences de fiabilité et de sécurité, le modèle markovien sous-jacent est de taille importante et devient rapidement difficile à exploiter et à analyser. De nombreuses méthodes de simplification ont été développées afin de diminuer la complexité ou la taille de ces modèles. Parmi ces méthodes, deux grandes classes peuvent être distinguées :

- les simplifications « graphiques »
- les simplifications « analytiques »

Les simplifications graphiques reposent sur une modification de la structure du modèle. Elles sont souvent appliquées à des formalismes à état-transition comme les réseaux de Petri ou les graphes de Markov. L'idée est de réduire le nombre de nœuds (places, transitions ou états) afin de diminuer la taille du modèle tout en conservant certaines propriétés entre le modèle initial et le modèle simplifié. Dans le cas des graphes de Markov, la simplification graphique consiste à éliminer ou agréger plusieurs états élémentaires. Cette réduction est réalisée soit

pour éviter les problèmes d'explosion combinatoire, soit pour éliminer des états élémentaires lorsque leur connaissance n'est pas utile aux évaluations quantitatives. Pour les systèmes dont le niveau de détail est assez fin ou pour des besoins de modélisation, le nombre d'états peut augmenter très rapidement (par exemple, l'ajout d'états fictifs pour rendre un système markovien). De plus, dans de nombreuses situations, l'utilisateur n'a besoin d'informations que sur des agrégats ou classes d'états. Cela peut se produire en particulier lorsque les paramètres de sûreté à calculer sont directement associés à des ensembles. Ainsi, une situation d'indisponibilité ou de mise en défaut de la sécurité sera évaluée par la probabilité que le système se trouve dans un ensemble d'états relatifs à des événements redoutés.

Nous allons mettre en œuvre la simplification graphique des graphes de Markov en agrégeant les états selon leur appartenance à des classes, tout en nous attachant à conserver une interprétation physique des classes mises en évidence.

Le deuxième type de simplification est analytique. En examinant le processus d'évolution des probabilités dans les différents états du graphe de Markov, on s'aperçoit que l'équation de Chapman Kolmogorov s'apparente à une équation d'état d'un système linéaire continu. La reformulation de l'équation [3] sous forme transposée donne :

$$\dot{\Pi}^T(t) = A.\Pi^T(t) \qquad \text{avec } \Pi^T(t) = \begin{bmatrix} P_1(t) \\ P_2(t) \\ \cdots \\ P_n(t) \end{bmatrix} \in [0,1]^n \quad \text{vecteur de probabilités}$$

A partir de cette forme, les simplifications d'équations d'état issues du domaine de l'automatique peuvent être appliquées. Les systèmes que l'on considère possèdent la propriété de double échelle de temps : une échelle de temps rapide relative à la dynamique interne au système dans ses modes opératoires et une échelle de temps plus lente associée à l'occurrence des défaillances des composants. La méthode des perturbations singulières sera exploitée au regard de cette remarque. La technique consiste à découpler le système suivant ses dynamiques puis à réduire la dimension de l'équation d'état en ne conservant que la dynamique dominante.

La partie suivante sera consacrée aux principaux concepts de la théorie des perturbations singulières de systèmes possédant deux échelles de temps. Celle-ci sera ensuite appliquée à des processus de Markov à deux dynamiques. Nous verrons comment une simplification analytique et graphique permettent de diminuer l'ordre du système et de faciliter le calcul des grandeurs de sûreté de fonctionnement.
La méthode sera ensuite formalisée dans le cas général en recherchant l'équation de Chapman-Kolmogorov du système réduit. La théorie des perturbations singulières ne pouvant être appliquée que sous certaines conditions, des critères d'applicabilité analytique et graphique seront exposés. Un exemple théorique servira de support pour illustrer ces notions et une analyse de performance sera menée.
Nous ferons ensuite le lien avec des formalismes de type état-transition tels que les réseaux de Petri.

2. Méthode des perturbations singulières

Le principe de la méthode des perturbations singulières s'applique à des systèmes possédant un petit paramètre dans leur modèle pouvant être négligé et qui, par annulation, introduit une singularité correspondant à une diminution de l'ordre du modèle.
En effet, l'existence de petits paramètres dans le modèle accroît l'ordre et introduit des variables et des transitoires rapides en regard de la vitesse d'évolution des grandeurs recherchées.

Par la suite, nous allons présenter la méthode des perturbations singulières dans le cas d'un système linéaire continu à double échelle de temps. Nous allons voir comment cette technique peut s'appliquer dans le cadre des processus de Markov de systèmes possédant la propriété de double échelle de temps. Des approches graphique et analytique seront ensuite décrites afin de mesurer le découplage des dynamiques et d'estimer la pertinence de la simplification sur un exemple de processus markovien.

2.1. Mise sous forme singulièrement perturbée

On appelle modèle singulièrement perturbé, un modèle de la forme :

$$\begin{bmatrix} \dot{x} \\ \varepsilon \dot{z} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} . \begin{bmatrix} x \\ z \end{bmatrix} \text{ avec } \begin{cases} x \in R^n \\ z \in R^m \end{cases} \text{ et les conditions initiales } \begin{bmatrix} x(0) \\ z(0) \end{bmatrix}$$

x est le vecteur formé des n variables lentes et z est le vecteur formé des m variables rapides.

$\varepsilon \in]0,1]$ est un petit paramètre correspondant donc au rapport des échelles de temps lente et rapide. Plus ce paramètre est petit et tend vers 0, plus les ordres de grandeur entre les deux échelles de temps sont différents.

2.2. Application de la technique des perturbations singulières : réduction de l'ordre du système

A partir de la connaissance du système d'équations différentielles mis sous la forme d'un système singulièrement perturbé, nous appliquons la méthode des perturbations singulières afin de diminuer sa dimension. Celui-ci s'écrit :

$$\begin{cases} \dot{x} = A_{11}x + A_{12}z \\ \varepsilon \dot{z} = A_{21}x + A_{22}z \end{cases}$$

La première équation, d'ordre n, décrit l'évolution de la partie lente, la deuxième, d'ordre m, décrit l'évolution de la partie rapide. Nous supposons que le paramètre ε est suffisamment petit pour que la différence des échelles de temps soit bien marquée. Les deux systèmes sont étudiés séparément.

2.2.1. Etude du modèle lent

Le modèle réduit lent est obtenu en supposant que toutes les variables rapides ont atteint leur régime permanent. Les variables x et z sont alors approchées par leurs composantes lentes x_L et z_L . Le système approché s'écrit alors :

$$\begin{cases} \dot{x}_L = A_{11}x_L + A_{12}z_L \\ 0 = A_{21}x_L + A_{22}z_L \end{cases}$$

En supposant que la matrice A_{22} est inversible :

$$\begin{cases} \dot{x}_L = A_L x_L \\ z_L = -A_{22}^{-1}A_{21}x_L \end{cases} \qquad \text{avec } A_L = A_{11} - A_{12}A_{22}^{-1}A_{21}$$

La deuxième équation en z_L correspond à la composante rapide entraînée par les variables lentes x_L. A_L est la matrice d'évolution de la partie lente. Les conditions initiales s'expriment en fonction de x(0) et z(0) :

$$\begin{cases} x_L(0) = x(0) \\ z_L(0) = -A_{22}^{-1}A_{21}x(0) \neq z(0) \end{cases}$$

Ainsi les variables lentes x du système initial sont approximées par les variables lentes approchées x_L sur toute la période d'évolution $t \geq 0$. Par contre, les conditions initiales des variables rapides z_L ne sont pas égales à z(0) car l'approximation du modèle lent suppose que les variables z ont atteint leur régime permanent.

2.2.2. Etude du modèle rapide

L'approximation de z par z_L n'est pas valable sur tout l'intervalle de temps. On introduit alors la variable z_R, composante rapide qui n'est valable que sur un court intervalle de temps appelé domaine de couche limite, pendant laquelle les variables lentes sont supposées immobiles. Cette composante est égale à la différence entre le système réel et le système lent approché, soit :

$$z_R = z - z_L$$

L'équation décrivant l'évolution du sous-système rapide est appelée **équation de couche limite**. Elle s'écrit en temps dilaté $\sigma = \frac{t}{\varepsilon}$ (échelle de temps « rapide »). Sur l'intervalle de temps du domaine de couche limite, les variables lentes sont immobiles, soit : $\frac{dx_L}{d\sigma} = 0$. On montre que l'équation de couche limite s'écrit :

$$\frac{dz_R}{d\sigma}(\sigma) = A_{22}z_R(\sigma) \qquad \text{avec } z_R(0) = z(0) + A_{22}^{-1}A_{21}x(0)$$

Ainsi, le système continu découplé peut s'écrire sous la forme :

$$\begin{pmatrix} \dot{x}_L \\ \dfrac{dz_R}{d\sigma} \end{pmatrix} = \begin{bmatrix} A_L & 0 \\ 0 & A_R \end{bmatrix} \begin{pmatrix} x_L \\ z_R \end{pmatrix}$$

Cependant, l'application de la théorie des perturbations singulières dans le cadre d'études de sûreté de fonctionnement ne sera intéressante que pour approcher les probabilités lentes x par leurs composantes lentes réduites x_L. En effet, les grandeurs de sûreté de fonctionnement recherchées sont régies par la dynamique lente, qui est fonction de l'ensemble des défaillances de composants. Quant à la dynamique rapide, dépendante du comportement du système dans ses modes opérationnels, elle ne sera utile que dans son comportement moyen ou asymptotique. Le régime transitoire décrit par l'équation de couche limite, ne sera donc pas exploité.

2.3. Passage d'une forme non-standard à une forme standard

Dans le cas général, le système d'équations différentielles n'est pas mis sous la forme standard, dans laquelle le vecteur des probabilités X est découpé suivant les composantes lentes et rapides. Il est alors nécessaire de trouver une matrice de passage permettant de passer à la forme standard et de pouvoir identifier les variables lentes et rapides et donc mettre en évidence la présence d'une double échelle de temps.
Ce paragraphe contribue également à montrer comment les différentes composantes peuvent être interprétées physiquement à partir d'une modélisation markovienne. Le système d'équations s'apparente alors un système d'équations de Chapman-Kolmogorov.

Soit un système continu linéaire formalisé, par commodité, dans les deux échelles de temps rapide et lente :

$$\varepsilon \dot{X} = A(\varepsilon).X, \qquad \frac{dX}{d\sigma} = F(\varepsilon)X, \qquad X \in R^{n+m} \qquad [4]$$

avec comme condition initiale : $X(t=0)=X_0$.

Nous supposons que la matrice A est composée de deux types de termes d'ordres de grandeur différents. D'où la dépendance de A en ε, ce dernier étant le terme multiplicatif entre les deux échelles.

Si det $A(0)\neq 0$, alors toutes les composantes de X seront rapides. Ainsi, si le système possède deux échelles de temps, on aura nécessairement det $A(0)=0$.
Posons $A(\varepsilon)=A_0+\varepsilon A_1(\varepsilon)$. Supposons que la dimension du noyau $Ker(A_0)$ est n et que l'espace image $Im(A_0)$ est de dimension m. Ces deux sous-espaces étant complémentaires, ils engendrent R^{n+m}, c'est-à-dire :

$$Ker(A_0) \oplus Im(A_0) = R^{n+m} \qquad [5]$$

Cela signifie que A_0 a n valeurs propres nulles dans l'espace propre $Ker(A_0)$ et m valeurs propres non nulles dans $Im(A_0)$. On montre alors [KOK99] que pour obtenir les variables

rapides, il suffit de choisir m vecteurs linéairement indépendants dans R^{n+m} orthogonaux à $Ker(A_0)$ et formant les lignes d'une matrice Q de dimension m*(m+n), telle que :

$$QX = 0 \Leftrightarrow A_0X = 0 \Leftrightarrow X \in Ker(A_0)$$

En particulier, Q peut être composée de m lignes indépendantes de A_0. On pourra alors prendre QX comme candidat aux variables rapides z.
Pour obtenir les variables lentes x, on exploite la propriété d'invariance des composantes lentes sur le domaine de couche limite. La dérivée de x par rapport à σ est donc nulle et x est constante pour ε=0. On cherche une matrice P de dimension n*(n+m) dont les lignes forment une base du noyau de A_0, c'est-à-dire vérifiant PA_0=0. Ainsi, pour ε=0, il vient :

$$P\frac{dX}{dt} = PA_0X = 0, \qquad \forall X \in R^{n+m}$$

De ce fait, les n vecteurs PX, constants sur le domaine de couche limite σ≥0, constituent un ensemble de n variables lentes x.

Ainsi, si A_0 satisfait [5], alors le changement de coordonnées :

$$\begin{cases} x = PX \\ z = QX \end{cases}$$

qui transforme le système [4] en :

$$\begin{cases} \dot{x} = A_{11}(\varepsilon)x + A_{12}(\varepsilon)z \\ \varepsilon\dot{z} = \varepsilon A_{21}(\varepsilon)x + A_{22}(\varepsilon)z \end{cases}$$

Ce système représente la forme standard recherchée, à condition que $A_{22}(0)$ soit non singulière.

Démonstration :

Soit $T^{-1} = \begin{bmatrix} V & W \end{bmatrix}$ la matrice inverse de $T=\begin{bmatrix} P \\ Q \end{bmatrix}$. Les colonnes de V et W forment respectivement des bases de $Ker(A_0)$ et $Im(A_0)$. Ainsi, PA_0 et A_0V sont nuls. D'où :

$$T\left(\frac{1}{\varepsilon}A_0 + A_1(\varepsilon)\right)T^{-1} = \begin{bmatrix} PA_1(\varepsilon)V & PA_1(\varepsilon)W \\ QA_1(\varepsilon)V & \frac{1}{\varepsilon}QA_0W + QA_1(\varepsilon)W \end{bmatrix}$$

Par identification, les termes matriciels recherchés sont :

$$\begin{cases} A_{11}(\varepsilon) = PA_1(\varepsilon)V \\ A_{12}(\varepsilon) = PA_1(\varepsilon)W \\ A_{21}(\varepsilon) = QA_1(\varepsilon)V \\ A_{22}(\varepsilon) = QA_0W + \varepsilon QA_1(\varepsilon)W \end{cases}$$

2.4. Exemple d'application à un processus markovien élémentaire

Soit le graphe de Markov représenté par la figure suivante :

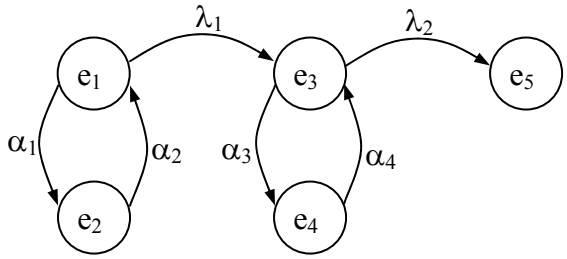


Figure 37 : Exemple applicatif

Nous allons faire l'hypothèse que les taux de transitions λ_1 et λ_2 sont du même ordre de grandeur. Les taux α_i sont supposés être beaucoup plus grands que les λ_j, c'est-à-dire $\lambda_j << \alpha_i$ $\forall$ (i,j).

Ecrivons l'équation de Chapman-Kolmogorov associée sous forme transposée :

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \end{pmatrix} = \begin{bmatrix} -(\alpha_1+\lambda_1) & \alpha_2 & 0 & 0 & 0 \\ \alpha_1 & -\alpha_2 & 0 & 0 & 0 \\ \lambda_1 & 0 & -(\alpha_3+\lambda_2) & \alpha_4 & 0 \\ 0 & 0 & \alpha_3 & -\alpha_4 & 0 \\ 0 & 0 & \lambda_2 & 0 & 0 \end{bmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix}$$

où $q_i(t)$ représente la probabilité d'être dans l'état e_i à l'instant t. Nous allons exploiter le fait que la matrice de transition est composée de termes d'ordres de grandeur différents. On introduit un petit paramètre $\lambda=\max(\lambda_1, \lambda_2)$. On suppose par exemple que $\lambda=\lambda_1$ et on pose $\lambda_2=\delta\lambda$. La matrice $A(\lambda)$ peut alors se décomposer sous la forme suivante $A(\lambda)=A_0+\lambda.A_1$:

$$A(\lambda) = \begin{bmatrix} -\alpha_1 & \alpha_2 & 0 & 0 & 0 \\ \alpha_1 & -\alpha_2 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_3 & \alpha_4 & 0 \\ 0 & 0 & \alpha_3 & -\alpha_4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \lambda. \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -\delta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta & 0 & 0 \end{bmatrix}$$

D'après les résultats énoncés précédemment, la dimension du noyau donne le nombre de variables lentes ; dans le cas présent, il est clair que dim (Ker A_0)=3 et dim(A_0)=2. Le système possède donc bien des composantes lentes et des composantes rapides.

Les matrices de passage P, Q, V et W sont recherchées afin de mettre le système sous la forme standard :

- P est de dimension 2x3 et vérifie l'égalité $PA_0=0$. Les lignes de A_0 forment donc une base du noyau de A_0^T, on obtient ainsi la matrice :

$$P = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Pour construire la matrice Q, on choisit deux lignes de A_0 formant deux vecteurs indépendants. D'où :

$$Q = \begin{bmatrix} -\alpha_1 & \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_3 & \alpha_4 & 0 \end{bmatrix}$$

La matrice de changement de cordonnées T est alors connue :

$$T = \begin{bmatrix} P \\ Q \end{bmatrix} = \left[\begin{array}{ccccc} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \hline -\alpha_1 & \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_3 & \alpha_4 & 0 \end{array}\right]$$

On en déduit la matrice inverse :

$$T^{-1} = \begin{bmatrix} V & W \end{bmatrix} = \left[\begin{array}{ccc:cc} \frac{\alpha_2}{\alpha_1+\alpha_2} & 0 & 0 & -\frac{1}{\alpha_1+\alpha_2} & 0 \\ \frac{\alpha_1}{\alpha_1+\alpha_2} & 0 & 0 & \frac{1}{\alpha_1+\alpha_2} & 0 \\ 0 & \frac{\alpha_4}{\alpha_3+\alpha_4} & 0 & 0 & -\frac{1}{\alpha_3+\alpha_4} \\ 0 & \frac{\alpha_3}{\alpha_3+\alpha_4} & 0 & 0 & \frac{1}{\alpha_3+\alpha_4} \\ 0 & 0 & 1 & 0 & 0 \end{array}\right]$$

Il est ensuite très facile de trouver la forme singulièrement perturbée de l'équation d'origine. Connaissant les matrices de passage P et Q, nous en déduisons les variables lentes et rapides par un changement de base.

Les 3 variables lentes sont obtenues par le changement de variable x=Pq :

$$\begin{cases} x_1 = q_1 + q_2 \\ x_2 = q_3 + q_4 \\ x_3 = q_5 \end{cases}$$

Les 2 variables rapides sont obtenues par le changement de variable z=Qq :

$$\begin{cases} z_1 = -\alpha_1 q_1 + \alpha_2 q_2 \\ z_2 = -\alpha_3 q_3 + \alpha_4 q_4 \end{cases} \qquad [6]$$

Soulignons que ces changements de base ne sont pas uniques. Cependant, un choix pertinent des matrices de passage P et Q met en évidence des regroupements judicieux d'états élémentaires. Ainsi, les variables lentes correspondent à des sommes de probabilités d'états élémentaires. Remarquons que si les arcs du graphe de Markov de la figure 37 pondérés par les taux « lents » λ_i sont supprimés, il reste trois ensembles d'états formant des classes d'états.

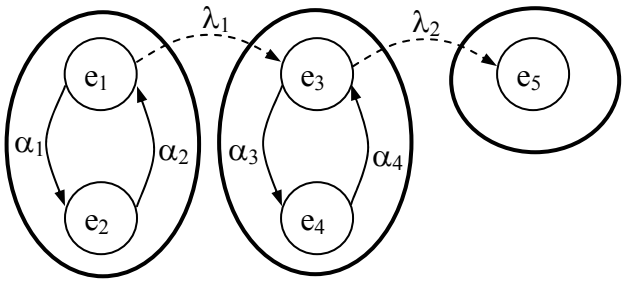


Figure 38 : Identification des variables lentes et des ensembles d'états

Les deux premières classes sont transitoires, et la troisième est finale. Les variables lentes correspondent aux probabilités d'être dans chacune de ces classes. L'équation réduite donnant l'évolution de ces probabilités, en supposant que le régime permanent des variables rapides est atteint, est donnée par :

$$\dot{x} = [A_{11}]x$$

soit, dans l'échelle de temps rapide :

$$\begin{cases} \dot{x}_1 = -\dfrac{\alpha_2}{\alpha_1 + \alpha_2} x_1 \\ \dot{x}_2 = \dfrac{\alpha_2}{\alpha_1 + \alpha_2} x_1 - \dfrac{\delta\alpha_4}{\alpha_3 + \alpha_4} x_2 \\ \dot{x}_3 = \dfrac{\delta\alpha_4}{\alpha_3 + \alpha_4} x_2 \end{cases}$$

puis en repassant dans l'échelle de temps lente :

$$\begin{cases} \dot{x}_1 = -\lambda_1 \dfrac{\alpha_2}{\alpha_1 + \alpha_2} x_1 \\ \dot{x}_2 = \lambda_1 \dfrac{\alpha_2}{\alpha_1 + \alpha_2} x_1 - \lambda_2 \dfrac{\alpha_4}{\alpha_3 + \alpha_4} x_2 \\ \dot{x}_3 = \lambda_2 \dfrac{\alpha_4}{\alpha_3 + \alpha_4} x_2 \end{cases}$$

Cette forme constitue une forme réduite du système d'équations d'origine. On reconnaît aisément une équation de Chapman-Kolmogorov associée à un graphe de Markov à trois états. Graphiquement, le processus de Markov sous-jacent est :

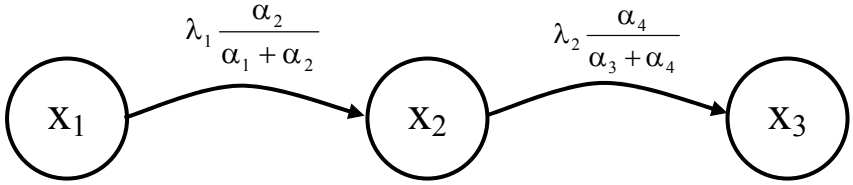


Figure 39 : Graphe de Markov associée à l'équation réduite

Chaque macro-état représenté correspond à la probabilité d'être dans l'une des classes d'états identifiées précédemment. Remarquons que les taux de transitions « lents » λ_i se retrouvent sur le graphe réduit mais sont affectés de « coefficients de pondération » dépendants des taux de transitions « rapides » α_i. Cependant, tous les arcs affectés par ces taux ont été éliminés.

Dans une étude quantitative des paramètres de sûreté de fonctionnement, la construction d'un graphe de Markov simplifié est très intéressante, si ce dernier fournit des résultats proches des résultats du graphe initial qui ne dépendent pas des états élémentaires mais uniquement des regroupements mis en évidence. Dans l'exemple présent, la dynamique rapide portée par les taux de transition rapides α_i a été simplifiée. Les regroupements d'états selon leur appartenance à des classes sont pertinents, en particulier si un sens physique se dégage de ces ensembles.

Nécessairement, les transitions entre états appartenant à un mode commun, ont une « vitesse » de franchissement beaucoup plus rapide que celles relatives aux taux de défaillance, puisqu'elles sont associées à une dynamique propre au système. Nous retrouvons alors la propriété de double échelle de temps.

L'objectif de la technique d'agrégation est d'aboutir à un graphe de Markov simplifié. Néanmoins, sa construction est fastidieuse si elle contraint de procéder à des regroupements, puisque l'établissement du graphe de Markov initial (avant regroupements) est sujet à l'explosion combinatoire du nombre d'états.

Afin d'interpréter le sens physique des paramètres de pondération, nous allons chercher la forme standard de l'équation de Chapman-Kolmogorov, connaissant les variables lentes.

Recherche de la forme standard

Etablissons le système d'équations différentielles donnant l'évolution des variables lentes ; par définition :

$$\begin{cases} \dot{x}_1 = \dot{q}_1 + \dot{q}_2 \\ \dot{x}_2 = \dot{q}_3 + \dot{q}_4 \\ \dot{x}_3 = \dot{q}_5 \end{cases}$$

Connaissant la matrice de transition A du graphe de Markov initial :

$$\begin{cases} \dot{x}_1 = -\lambda_1 q_1 \\ \dot{x}_2 = \lambda_1 q_1 - \lambda_2 q_3 \\ \dot{x}_3 = \lambda_2 q_3 \end{cases}$$

Soit de façon équivalente :

$$\begin{cases} \dot{x}_1 = -\lambda_1 \dfrac{q_1}{q_1 + q_2}(q_1 + q_2) \\ \dot{x}_2 = \lambda_1 \dfrac{q_1}{q_1 + q_2}(q_1 + q_2) - \lambda_2 \dfrac{q_3}{q_3 + q_4}(q_3 + q_4) \\ \dot{x}_3 = \lambda_2 \dfrac{q_3}{q_3 + q_4}(q_3 + q_4) \end{cases}$$

$$\Rightarrow \begin{cases} \dot{x}_1 = -\lambda_1 \dfrac{q_1}{q_1 + q_2} x_1 \\ \dot{x}_2 = \lambda_1 \dfrac{q_1}{q_1 + q_2} x_1 - \lambda_2 \dfrac{q_3}{q_3 + q_4} x_2 \\ \dot{x}_3 = \lambda_2 \dfrac{q_3}{q_3 + q_4} x_2 \end{cases} \qquad [7]$$

$$\text{Posons} \quad \begin{cases} p_1 = \dfrac{q_1}{q_1 + q_2} \\ p_2 = \dfrac{q_2}{q_1 + q_2} \\ p_1 + p_2 = 1 \end{cases} \quad \text{et} \quad \begin{cases} p_3 = \dfrac{q_3}{q_3 + q_4} \\ p_4 = \dfrac{q_4}{q_3 + q_4} \\ p_3 + p_4 = 1 \end{cases} \qquad [8]$$

L'équation [7] devient :

$$\begin{cases} \dot{x}_1 = -\lambda_1 p_1 x_1 \\ \dot{x}_2 = \lambda_1 p_1 x_1 - \lambda_2 p_3 x_2 \\ \dot{x}_3 = \lambda_2 p_3 x_2 \end{cases} \qquad [9]$$

Les variables p_i sont des fonctions du temps.

Remarque :

Les fonctions p_1 et p_3 se retrouvent dans les équations différentielles des variables lentes. Il est aisé se constater que ces fonctions correspondent à des probabilités conditionnelles. p_1 est la probabilité que le système se trouve à l'instant t dans l'état élémentaire e_1 sachant qu'il appartient au « Macro-Etat » x_1 (composé des états (e_1, e_2)). La fonction p_3 est la probabilité conditionnelle que le système se trouve à l'instant t dans l'état e_3 sachant qu'il appartient au Macro-Etat x_2, etc… En effet, par définition :

$$p_1 = \Pr\left[\text{Système} \in e_1 \middle| \text{Système} \in \{e_1 \cup e_2\}\right] = \frac{\Pr\left[S \in e_1 \text{ et } S \in \{e_1 \cup e_2\}\right]}{\Pr\left[S \in \{e_1 \cup e_2\}\right]} = \frac{\Pr\left[S \in e_1\right]}{\Pr\left[S \in \{e_1 \cup e_2\}\right]} = \frac{q_1}{q_1 + q_2}$$

Remarquons également que la suppression des arcs portant λ_1 et λ_2 donne deux composantes fortement connexes, dont les équations de Chapman Kolmogorov associées sont :

$$\begin{cases} \dot{r}_1 = -\alpha_1 r_1 + \alpha_2 r_2 \\ \dot{r}_2 = \alpha_1 p_1 - \alpha_2 r_2 \\ \dot{r}_3 = -\alpha_3 r_3 + \alpha_4 r_4 \\ \dot{r}_4 = \alpha_3 r_3 - \alpha_4 r_4 \end{cases} \qquad [10]$$

Avec les contraintes suivantes (probabilités totales) : $\begin{cases} r_1 + r_2 = 1 \\ r_3 + r_4 = 1 \end{cases}$

En dérivant les expressions [8] par rapport au temps :

$$\begin{cases} \dot{p}_1 = -\lambda p_1 p_2 - \alpha_1 p_1 + \alpha_2 p_2 \\ \dot{p}_2 = \lambda p_1 p_2 + \alpha_1 p_1 - \alpha_2 p_2 \\ \dot{p}_3 = \lambda\left[- p_3 p_1 \frac{x_1}{x_2} + \delta p_3^2 + p_1 \frac{x_1}{x_2} - \delta p_3 \right] - \alpha_3 p_3 + \alpha_4 p_4 \\ \dot{p}_4 = \lambda\left[- p_4 p_1 \frac{x_1}{x_2} + \delta p_3 p_4 \right] + \alpha_3 p_3 - \alpha_4 p_4 \end{cases} \qquad [11]$$

Notons au passage que l'annulation de λ transforme [11] en [10].

En changeant d'échelle de temps dans [9] et [11] : $\tau = \lambda t$

$$\begin{cases} \frac{dx_1}{d\tau} = -p_1 x_1 \\ \frac{dx_2}{d\tau} = p_1 x_1 - \delta p_3 x_2 \\ \frac{dx_3}{d\tau} = \delta p_3 x_2 \end{cases}$$

$$\begin{cases} \lambda \dfrac{dp_1}{d\tau} = -\lambda p_1 p_2 - \alpha_1 p_1 + \alpha_2 p_2 \\ \lambda \dfrac{dp_2}{d\tau} = \lambda p_1 p_2 + \alpha_1 p_1 - \alpha_2 p_2 \\ \lambda \dfrac{dp_3}{d\tau} = \lambda \left[- p_3 p_1 \dfrac{x_1}{x_2} + \delta p_3^2 + p_1 \dfrac{x_1}{x_2} - \delta p_3 \right] - \alpha_3 p_3 + \alpha_4 p_4 \\ \lambda \dfrac{dp_4}{d\tau} = \lambda \left[- p_4 p_1 \dfrac{x_1}{x_2} + \delta p_3 p_4 \right] + \alpha_3 p_3 - \alpha_4 p_4 \end{cases}$$

On reconnaît ici la forme standard recherchée : $\begin{cases} \dot{x} = f(x, p, \lambda, \tau) \\ \lambda \dot{p} = g(x, p, \lambda, \tau) \end{cases}$.

Rappelons que le nombre de variables rapides est de 2 ($\dim(A_0)=2$). Or le système d'équations rapides montre 4 équations. Cependant, celles-ci sont liées par les contraintes [8]. Il ne reste alors que deux degrés de liberté.

Il faut à présent montrer que ces probabilités conditionnelles peuvent être assimilées à des variables rapides ou, en d'autres termes, que ces fonctions peuvent être approximées par leur valeur asymptotique dans les équations des variables lentes. En effet, les variables rapides z mises en évidence par le changement de variable z=Qq n'aboutissent pas directement aux probabilités conditionnelles.

Il faut donc s'assurer que l'approximation des perturbations singulières faite sur z revient à négliger le transitoire des fonctions probabilités conditionnelles, c'est-à-dire qu'elles sont régies par une dynamique rapide.

En dérivant les expressions [6] et compte-tenu de [8] et [9] :

$$\begin{cases} \dot{z}_1 = -\lambda_1 p_1 x_1 (\alpha_2 p_2 - \alpha_1 p_1) - \dot{p}_1 x_1 (\alpha_1 + \alpha_2) \\ \dot{z}_2 = (\lambda_1 p_1 x_1 - \lambda_2 p_3 x_2)(\alpha_4 p_4 - \alpha_3 p_3) - \dot{p}_3 x_2 (\alpha_3 + \alpha_4) \end{cases}$$

Ainsi, à partir de cette forme, on applique l'approximation des d'ordre 0 en négligeant les paramètres λ_1 et λ_2. On constate alors que l'annulation de $\dot{z}$ revient à annuler $\dot{p}$. De ce fait, dans l'hypothèse où le paramètre λ est suffisamment petit pour justifier la réduction de l'ordre du système, les probabilités conditionnelles s'assimilent à des variables rapides. Leur valeur asymptotique se calcule facilement :

$$\begin{cases} 0 = -\alpha_1 \overline{p}_1 + \alpha_2 \overline{p}_2 \\ 0 = \alpha_1 \overline{p}_1 - \alpha_2 \overline{p}_2 \\ 0 = -\alpha_3 \overline{p}_3 + \alpha_4 \overline{p}_4 \\ 0 = \alpha_3 \overline{p}_3 - \alpha_4 \overline{p}_4 \end{cases} \quad \text{avec les contraintes} \quad \begin{cases} \overline{p}_1 + \overline{p}_2 = 1 \\ \overline{p}_3 + \overline{p}_4 = 1 \end{cases}$$

La résolution est triviale et donne :

$$\begin{cases} \overline{p}_1 = \dfrac{\alpha_2}{\alpha_1 + \alpha_2} \\ \overline{p}_2 = \dfrac{\alpha_1}{\alpha_1 + \alpha_2} \end{cases} \qquad \begin{cases} \overline{p}_3 = \dfrac{\alpha_4}{\alpha_3 + \alpha_4} \\ \overline{p}_4 = \dfrac{\alpha_3}{\alpha_3 + \alpha_4} \end{cases}$$

Ces résultats s'identifient bien aux coefficients de pondération trouvés par la méthode directe. Les probabilités conditionnelles asymptotiques sont déduites des différents graphes de Markov issus du graphe d'origine en supprimant les taux λ_1 et λ_2. Dans l'exemple présenté, ces graphes sont ergodiques et constituent des classes d'états. Les matrices de transition associées sont irréductibles, assurant l'unicité des probabilités p_i et l'indépendance des conditions initiales d'après le critère d'unicité évoqué dans la partie 6.3.8.

Nous avons vu sur un exemple simple comment un graphe de Markov possédant deux types de transition d'ordre de grandeur différent, se simplifie structurellement par l'agrégation d'états élémentaires en Macro-Etats ; l'évaluation des probabilités d'être dans les Macro-états est également simplifiée, puisque la résolution d'une équation différentielle de dimension n+m se ramène à un système d'équations différentielles de dimension n. Bien entendu le processus d'agrégation n'a d'intérêt que si les paramètres (de sûreté de fonctionnement) attendus sont issus du calcul des probabilités $Q_i(t)$.

Nous allons à présent illustrer la démarche de simplification dans le cas général d'un processus markovien de taille quelconque et possédant deux échelles de temps.

3. Application de la technique des perturbations singulières dans le cas général

3.1. Hypothèses et notations

Dans le cas général, des hypothèses structurelles sont effectuées sur les graphes de Markov traités:

- les graphes sont composés de deux types de transitions : des transitions « rapides » notées α et des transitions lentes notées λ. Ainsi $\alpha >> \lambda$.
- les états élémentaires liés entre eux par des taux de franchissement rapides α seront supposés former des classes d'équivalence (transitoires ou finales). Les graphes issus de ces classes forment des composantes fortement connexes. La matrice de transition associée à chaque composante est donc irréductible. Nous allons appeler ces composantes *Macro-Etats*. Les taux de franchissement lents λ permettent de passer d'un macro-état à un autre. Il n'y a pas de taux λ à l'intérieur des macro-états.

Nous supposons de ce fait que le graphe de Markov est « bien structuré » et que les regroupements d'états sont connus a priori comme l'illustre la figure 40 :

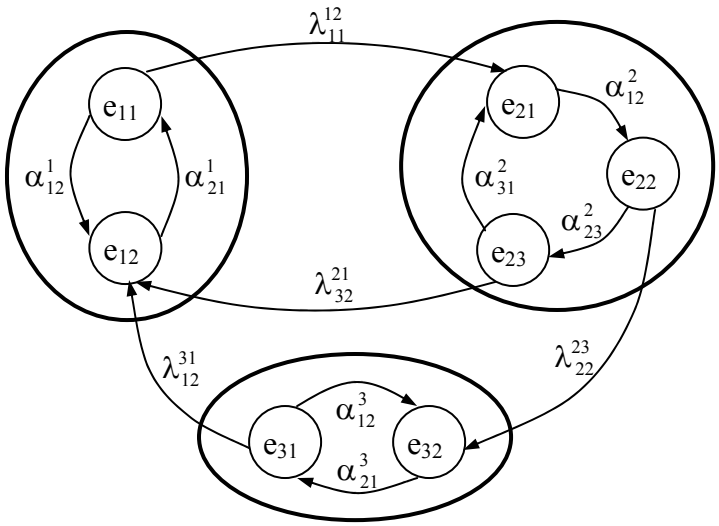


Figure 40 : Notations utilisées

Nous allons utiliser des notations qui faciliteront l'identification des états et leur appartenance à un macro-état grâce à des variables indicées. Il en sera de même pour les taux de transition pour lesquelles les indices indiqueront les états amont et aval et les macro-états amont et aval.

E : espace des états e_i. Card(E)=N

N : nombre total d'états de E (dans l'exemple de la figure 40, N=7)

n : nombre total de macro-états (n=3)

J_i : macro-état n°i avec $i \in \{1..n\}$. Ainsi, $E = \bigcup_{i\in[1..n]} J_i$

N_i : nombre d'états dans chaque macro-état, $i \in \{1..n\}$ (ici $N_1=2$, $N_2=3$, $N_3=2$)

On a donc : $N = \sum_{k=1}^{n} N_k$

e_{ji} : état i dans le macro-état j avec $i \in \{1..N_j\}$, $j \in \{1..n\}$

α_{ij}^{k} : taux de transition de l'état i vers l'état j dans le macro état k

λ_{ij}^{kl} : taux de transition de l'état i du macro-état k vers l'état j du macro-état l

$q_{ji}(t)$: probabilité de se retrouver à l'instant t dans l'état i du macro-état j avec $i \in \{1..N_j\}$, $j \in \{1..n\}$

$p_{ji}(t)$: probabilité conditionnelle de se retrouver à l'instant t dans l'état i du macro-état j sachant qu'on est dans le groupe J_j avec $i \in \{1..N_j\}$, $j \in \{1..n\}$

$$\text{Ainsi : } p_{ji}(t) = \mathbb{P}_t[x_{ji}/J_j] = \mathbb{P}_t[x_{ji} \cap J_j\,]/\,\mathbb{P}_t[J_j] = \frac{q_{ji}(t)}{q_{j1}(t)+q_{j2}(t)+\ldots+q_{jN_j}(t)} = \frac{q_{ji}(t)}{\sum_{k=1}^{N_j} q_{jk}(t)}$$

$Q_j(t)$: probabilité de se retrouver dans le macro-état J_j à l'instant t

Avec les notations utilisées : $Q_j(t) = \sum_{k=1}^{N_j} q_{jk}(t)$

3.2. Mise en équation

La première étape consiste à établir une équation générale donnant les probabilités d'être dans les macro-états à un instant t en faisant apparaître les probabilités conditionnelles.

Par définition :

$q_{ji}(t+dt)$ = Pr [système dans un état $\neq e_{ji}$ à t et dans l'état e_{ji} à t+dt] [12]
+ Pr [système dans l'état e_{ji} à t et y reste] [13]

avec

[12]= Pr [système dans un état $\neq e_{ji}$ à t dans un autre macro-état $(\neq J_j)$] [14]
+ Pr [système dans un état $\neq e_{ji}$ à t dans le même macro-état $(=J_j)$] [15]

On développe alors chacun des termes :

$$[15] = \sum_{\substack{k=1 \\ k\neq i}}^{N_j} q_{jk}(t).\alpha_{ki}^{j}.dt$$

$$[14] = \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} q_{lk}(t).\lambda_{ki}^{lj}.dt$$

$$[13] = q_{ji}(t)\left[1 - \left(\sum_{\substack{k=1 \\ k\neq i}}^{N_j} \alpha_{ik}^{j} + \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} \lambda_{ik}^{jl}\right)dt\right]$$

On en déduit :

$$\dot{q}_{ji}(t) = \frac{q_{ji}(t+dt) - q_{ji}(t)}{dt} = \sum_{\substack{k=1 \\ k\neq i}}^{N_j} q_{jk}(t).\alpha_{ki}^{j} + \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} q_{lk}(t).\lambda_{ki}^{lj} - q_{ji}(t).\left(\sum_{\substack{k=1 \\ k\neq i}}^{N_j} \alpha_{ik}^{j} + \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} \lambda_{ik}^{jl}\right) \quad [16]$$

Connaissant les dérivées des probabilités élémentaires, on obtient facilement les dérivées des probabilités d'être dans les macro-états :

$$\dot{Q}_j(t) = \sum_{p=1}^{N_j} \dot{q}_{jp}(t) = \sum_{p=1}^{N_j} \sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jk}(t).\alpha_{kp}^{j} + \sum_{p=1}^{N_j} \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} q_{lk}(t).\lambda_{kp}^{lj} - \sum_{p=1}^{N_j} q_{jp}(t).\left(\sum_{\substack{k=1 \\ k\neq p}}^{N_j} \alpha_{pk}^{j} + \sum_{\substack{l=1 \\ l\neq j}}^{n} \sum_{k=1}^{N_l} \lambda_{pk}^{jl}\right) \quad [17]$$

Or, des membres de l'équation s'annulent, en effectuant un changement de variables :

$$\sum_{p=1}^{N_j} \sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jk}(t).\alpha_{kp}^{j} - \sum_{p=1}^{N_j} \sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jp}(t).\alpha_{pk}^{j} = \sum_{p=1}^{N_j} \sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jk}(t).\alpha_{kp}^{j} - \sum_{k=1}^{N_j} \sum_{\substack{p=1 \\ p\neq k}}^{N_j} q_{jp}(t).\alpha_{pk}^{j}$$

$$= \sum_{p=1}^{N_j} \left(\sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jk}(t).\alpha_{kp}^{j} - \sum_{\substack{k=1 \\ k\neq p}}^{N_j} q_{jk}(t).\alpha_{kp}^{j} \right) = 0$$

L'équation [17] devient alors :

$$\dot{Q}_j(t) = \sum_{p=1}^{N_j} \sum_{\substack{l=1 \\ l \neq j}}^{n} \sum_{k=1}^{N_l} q_{lk}(t).\lambda_{kp}^{lj} - \sum_{p=1}^{N_j} \sum_{\substack{l=1 \\ l \neq j}}^{n} \sum_{k=1}^{N_l} q_{jp}(t).\lambda_{pk}^{jl}$$

D'après la définition des probabilités conditionnelles, les probabilités élémentaires q sont substituées par :

$$\begin{cases} q_{lk}(t) = p_{lk}(t).Q_l(t) \\ q_{jp}(t) = p_{jp}(t).Q_j(t) \end{cases}$$

On obtient finalement le système d'équations différentielles recherché :

$$\dot{Q}_j(t) = \sum_{\substack{l=1 \\ l \neq j}}^{n} Q_l(t) \left[\sum_{p=1}^{N_j} \sum_{k=1}^{N_l} \lambda_{kp}^{li}.p_{lk}(t) \right] - Q_j(t) \left[\sum_{p=1}^{N_j} \sum_{\substack{l=1 \\ l \neq j}}^{n} \sum_{k=1}^{N_l} \lambda_{pk}^{jl}.p_{jp}(t) \right] \qquad \forall j \in [1..n] \qquad [18]$$

La résolution de ce système donne les probabilités d'être dans chacun des macro-états J_j pour tout $j \in [1, n]$.

En posant : $\begin{cases} \beta_l^j = \sum_{p=1}^{N_j} \sum_{k=1}^{N_l} \lambda_{kp}^{lj}.p_{lk}(t) \\ \beta_j^j = -\sum_{p=1}^{N_j} \sum_{\substack{l=1 \\ l \neq j}}^{n} \sum_{k=1}^{N_l} \lambda_{pk}^{jl}.p_{jp}(t) \end{cases}$ le système précédent se réduit simplement à :

$$\boxed{\dot{Q}_j(t) = \sum_{l=1}^{n} \beta_l^j.Q_l(t) \qquad \forall j \in [1..n]} \qquad [19]$$

En ajoutant les conditions initiales : $[Q_1(0)\ Q_2(0)\ \ldots\ Q_n(0)]^T$.

L'équation de Chapman Kolmogorov mise sous la forme [19] est particulièrement intéressante ; elle met en évidence les probabilités de présence dans les macro-états et fait apparaître des **taux de transition « équivalents » β** entre ces macro-états. Ces taux de transition sont des combinaisons linéaires des taux de transition lents λ pondérés par les probabilités conditionnelles p(t). La dynamique rapide apparaît donc dans les taux de transition β par l'intermédiaire de ces p(t). Un taux de transition équivalent entre deux macro-états est égal à la somme de tous les taux de transitions λ issus du macro-état amont vers le macro-état aval de la transition, pondérés par les probabilités conditionnelles d'être dans les états élémentaires à l'origine de ces transitions lentes. L'exemple suivant illustre ces taux équivalents :

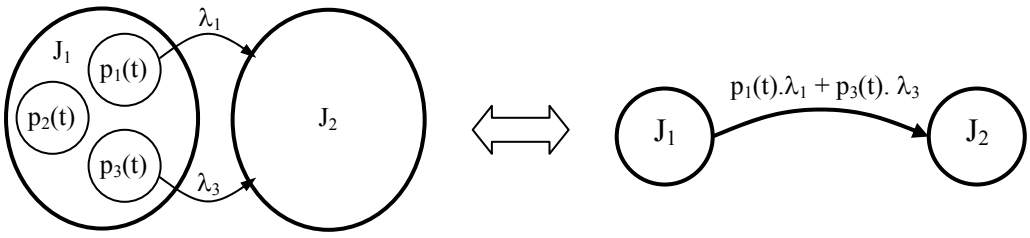


Figure 41 : Exemple de taux équivalent

Précisons que la résolution du système [18] donne une solution exacte car aucune simplification analytique n'a été réalisée sur les équations d'origine. Seuls des changements de variables mettant en évidence l'existence des macro-états ont été effectués. Le nombre d'inconnus et la dimension du système d'équations différentielles restent donc identiques au système d'équation de Chapman-Kolmogorov initial (c'est-à-dire N).

La réduction de la dimension du système consistera à diminuer le nombre de fonctions dépendantes du temps en les substituant par des constantes comme sur l'exemple présenté. La méthode des perturbations singulières sera appliquée sur le système après avoir séparé les dynamiques. La dimension sera ramenée à une valeur égale au nombre de variables lentes.

3.3. Recherche d'une forme standard

Par définition : $q_{ji}(t)=p_{ji}(t).Q_j(t) \qquad \forall\, j\in[1, n]$ et $\forall\, i\in[1, N_j]$

En dérivant cette équation par rapport à t : $\dot{q}_{ji} = \dot{p}_{ji}Q_j + p_{ji}\dot{Q}_j$

En remplaçant les termes $\dot{q}_{ji}$ et $\dot{Q}_j$ par les expressions [16] et [18] :

$$\begin{aligned}\dot{p}_{ji} = &-p_{ji}\sum_{p=1}^{N_j}\sum_{\substack{l=1\\ l\neq j}}^{n}\frac{Q_l}{Q_j}\sum_{k=1}^{N_l}p_{lk}.\lambda_{kp}^{lj} + p_{ji}\sum_{p=1}^{N_j}\sum_{\substack{l=1\\ l\neq j}}^{n}\sum_{k=1}^{N_l}p_{jp}.\lambda_{pk}^{jl} \;+\; \sum_{\substack{k=1\\ k\neq i}}^{N_j}p_{jk}.\alpha_{ki}^{j} + \sum_{\substack{l=1\\ l\neq j}}^{n}\frac{Q_l}{Q_j}\sum_{k=1}^{N_l}p_{lk}.\lambda_{ki}^{jl}\\ &- p_{ji}\left(\sum_{\substack{k=1\\ k\neq i}}^{N_j}\alpha_{ik}^{j} + \sum_{\substack{l=1\\ l\neq j}}^{n}\sum_{k=1}^{N_l}\lambda_{ik}^{jl}\right)\end{aligned} \qquad [20]$$

Posons $\lambda = \max_{j,i,k,l}\left\{\lambda_{ji}^{kl}\right\}$ et $\lambda_{ji}^{kl} = \lambda\delta_{ji}^{kl} \;\; \forall\{j,i,k,l\}$. On effectue alors le changement de variable $\tau=\lambda t$ dans les équations [19] et [20] :

$$\begin{cases}\dfrac{dQ_j}{d\tau} = \displaystyle\sum_{\substack{l=1\\ l\neq j}}^{n}Q_l\left[\sum_{p=1}^{N_j}\sum_{k=1}^{N_l}\delta_{kp}^{li}.p_{lk}\right] - Q_j\left[\sum_{p=1}^{N_j}\sum_{\substack{l=1\\ l\neq j}}^{n}\sum_{k=1}^{N_l}\delta_{pk}^{jl}.p_{jp}\right]\\ \lambda\dfrac{dp_{ji}}{d\tau} = \lambda\left[-p_{ji}\displaystyle\sum_{p=1}^{N_j}\sum_{\substack{l=1\\ l\neq j}}^{n}\frac{Q_l}{Q_j}\sum_{k=1}^{N_l}p_{lk}.\delta_{kp}^{lj} + p_{ji}\sum_{p=1}^{N_j}\sum_{\substack{l=1\\ l\neq j}}^{n}\sum_{k=1}^{N_l}p_{jp}.\delta_{pk}^{jl} + \sum_{\substack{l=1\\ l\neq j}}^{n}\frac{Q_l}{Q_j}\sum_{k=1}^{N_l}p_{lk}.\delta_{ki}^{jl} - p_{ji}\sum_{\substack{l=1\\ l\neq j}}^{n}\sum_{k=1}^{N_l}\delta_{ik}^{jl}\right]\\ \qquad + \displaystyle\sum_{\substack{k=1\\ k\neq i}}^{N_j}p_{jk}.\alpha_{ki}^{j} \;-\; p_{ji}\sum_{\substack{k=1\\ k\neq i}}^{N_j}\alpha_{ik}^{j}\end{cases} \qquad [21]$$

pour tout $j\in[1, n]$ et $i \in[1, N_j]$.

Le système d'équations [21] est la forme standard recherchée $\begin{cases} \dot{Q} = f(Q,p,\lambda,t) \\ \lambda\dot{p} = g(Q,p,\lambda,t) \end{cases}$

La réduction de la dimension du système peut être maintenant opérée sur cette forme.

3.4. Application des perturbations singulières : approximation d'ordre 0

Dans les études de sûreté de fonctionnement, une analyse quantitative consistera à évaluer la probabilité d'être dans les différents modes de fonctionnement. La connaissance de l'ensemble de ces probabilités sera suffisante pour estimer l'évolution de la fiabilité ou de la disponibilité. En effet, un dysfonctionnement total ou partiel du système étudié se traduit structurellement par la présence dans un macro-état identifié comme étant l'ensemble des états de défaillances. Bien entendu, l'évolution des probabilités de présence dans ces modes suit une dynamique lente puisque les changements ont lieu sur occurrence d'événements (stochastiques) relatifs à une (ou plusieurs) défaillance(s). La dynamique rapide liée à l'évolution du système dans chaque mode de fonctionnement intervient implicitement sur la dynamique lente en ajustant les taux de défaillance. Les termes de pondération initialement dépendants du temps, atteignent quasiment instantanément leur régime permanent au regard de la dynamique lente.
Comme nous l'avons vu dans le chapitre introductif, le comportement dynamique d'un système peut être vu comme une succession de sections d'évolutions déterministes connectées entre elles par des événements aléatoires. Dans notre modèle markovien, chaque section est approchée par un comportement « moyen » ; celui-ci se traduit mathématiquement par une substitution de fonctions dépendantes du temps p(t) par leur valeur asymptotique.

Remarque : La recherche de limites asymptotiques de fonctions déterministes liées à un fonctionnement nominal peut sembler surprenante (par exemple le cycle périodique d'ouverture/fermeture d'un actionneur). En réalité, les variables rapides apparaissant dans la forme standard représentent des probabilités de présence dans des états élémentaires. Ainsi, toutes les variables ou événements traduisant le fonctionnement du système dans chaque mode de fonctionnement sont convertis sur un modèle analytique markovien par un ensemble d'indicateurs probabilistes qui synthétise « statistiquement » le comportement du système. Par exemple, les cycles d'ouverture/fermeture périodique seront modélisés par des changements d'états. La dynamique rapide se traduira par la probabilité que l'actionneur soit en état d'ouverture et de fermeture.

Mathématiquement, ce comportement « moyen » est formulé par l'approximation d'ordre 0 dans l'équation [21].

On aboutit à un ensemble d'équations algébriques à résoudre :

$$0 = \sum_{\substack{k=1 \\ k\neq i}}^{N_j} \overline{p}_{jk}.\alpha_{ki}^{j} \; - \; \overline{p}_{ji} \sum_{\substack{k=1 \\ k\neq i}}^{N_j} \alpha_{ik}^{j} \qquad [22]$$

(Les variables barrées représentent la solution de l'approximation).

Or, les équations de Chapman-Kolmogorov donnant les probabilités conditionnelles s'écrivent :

$$\dot{p}_{ji}(t) = \frac{p_{ji}(t+dt) - p_{ji}(t)}{dt} = \sum_{\substack{k=1 \\ k\neq i}}^{N_j} p_{jk}(t).\alpha_{ki}^{j} - p_{ji}(t)\sum_{\substack{k=1 \\ k\neq i}}^{N_j} \alpha_{ik}^{j} \qquad i\in[1, N_j]$$

Les distributions asymptotiques pour chaque macro-état J_j sont obtenues en annulant $\dot{p}_{ji}$.
On ajoute les contraintes suivantes (système complet d'événements) :

$$\sum_{i=1}^{N_j} \overline{p}_{ji} = 1 \quad \text{pour tout macro-état } J_j,\ j\in[1, n]$$

Connaissant toutes les probabilités p_{ji} asymptotiques, l'équation de Chapman-Kolmogorov sous-jacente au graphe agrégé n'est plus que de dimension n.

3.5. Calcul des probabilités conditionnelles p_{ji}

Supposons que tous les états élémentaires sont identifiés selon leur appartenance à une composante fortement-connexe. La structure de la matrice de transition associée est alors analogue à celle de la figure 33 (dans le cas des systèmes non réparables).
L'approximation d'ordre 0 donne une matrice bloc-diagonale après suppression de l'ensemble des taux λ.

Le calcul des probabilités conditionnelles est ensuite très simple :

- chaque bloc de la matrice représente la matrice de transition du graphe de Markov associée à un mode de fonctionnement.
- dans l'hypothèse d'irréductibilité de ces blocs matriciels, les distributions asymptotiques ne dépendent pas des conditions initiales.

Reprenons l'exemple de la figure 34 après suppression des taux lents :

$$A = \begin{pmatrix}
-\alpha_1 & \alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\alpha_2 & -\alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\alpha_3 & \alpha_3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\alpha_4 & \alpha_4 & 0 & 0 & 0 & 0 \\
0 & 0 & \alpha_5 & 0 & -\alpha_5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\alpha_6 & \alpha_6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha_7 & -\alpha_7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure 42 : Blocs matriciels obtenus après suppression des taux lents

Dans le cas présent, le graphe de Markov est composé de trois composantes fortement connexes non triviales, il y a donc trois distributions limites à calculer :

$$\begin{cases} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{bmatrix} -\alpha_1 & \alpha_1 \\ \alpha_2 & -\alpha_2 \end{bmatrix} \begin{pmatrix} \overline{p}_1 \\ \overline{p}_2 \end{pmatrix} \\ \overline{p}_1 + \overline{p}_2 = 1 \end{cases} \quad \begin{cases} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \begin{bmatrix} -\alpha_3 & \alpha_3 & 0 \\ 0 & -\alpha_4 & \alpha_4 \\ \alpha_5 & 0 & -\alpha_5 \end{bmatrix} \begin{pmatrix} \overline{p}_3 \\ \overline{p}_4 \\ \overline{p}_5 \end{pmatrix} \\ \overline{p}_3 + \overline{p}_4 + \overline{p}_5 = 1 \end{cases} \quad \begin{cases} \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{bmatrix} -\alpha_6 & \alpha_6 \\ \alpha_7 & -\alpha_7 \end{bmatrix} \begin{pmatrix} \overline{p}_7 \\ \overline{p}_8 \end{pmatrix} \\ \overline{p}_7 + \overline{p}_8 = 1 \end{cases}$$

Le graphe agrégé est alors très simple :

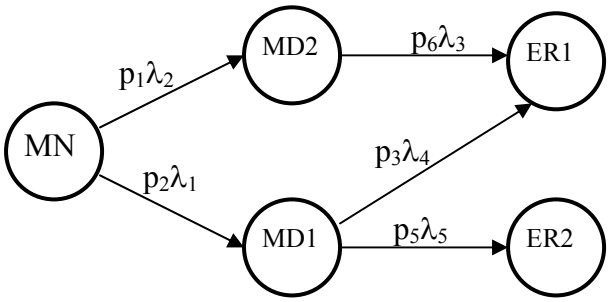


Figure 43 : Graphe de Markov agrégé

Remarque :
Les probabilités conditionnelles asymptotiques pondèrent des taux de transition d'un macro-état amont vers un macro-état aval. Le calcul des distributions limites est réalisé à partir de la structure de la composante fortement connexe relative au **macro-état amont**. Les coefficients de pondération ne dépendent pas de la structure avale. De ce fait, les propriétés de connexité et d'ergodicité ne sont nécessaires que pour les composantes transitoires (sur la figure précédente, MN, MD1 et MD2). La propriété de connexité des macro-états absorbants (ER1 et ER2) n'a pas besoin d'être vérifiée. Cette remarque n'est valable que pour **les systèmes non réparables**.

4. Mise en évidence des dynamiques

4.1. Propriété de double échelle de temps

Définition : Système à double échelle de temps [BEN82]
Soit un système linéaire :

$$\dot{X} = AX \qquad \text{avec } X \in R^{n+m}$$

Ce système a la propriété de double échelle de temps, s'il peut être décomposé en deux sous-systèmes :

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A_l & 0 \\ 0 & A_r \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} \qquad \text{avec } \begin{cases} x \in R^n \\ z \in R^m \end{cases}$$

tels que :

$$|\lambda_{max}(A_l)| << |\lambda_{min}(A_r)| \qquad [23]$$

où λ_i sont les valeurs propres de la matrice A.

A_l est la sous-matrice regroupant les modes lents et A_r est la sous-matrice regroupant les modes rapides.

La relation [23] s'exprime également sous la forme suivante [BOU90] :

$$\left\|A_r^{-1}\right\| << \left\|A_l\right\|^{-1} \qquad [24]$$

où $\|A\|$ est la norme de A.

Un critère d'applicabilité de la méthode des perturbations singulières est la présence de deux variétés de variables suffisamment séparées. Dans le cas général, la matrice de transition d'un processus markovien possédant les deux échelles de dynamique n'est pas sous forme bloc-diagonale et le critère énoncé n'est pas vérifiable directement. La mise en évidence des dynamiques implique de trouver la transformation de l'équation d'état permettant de comparer les normes de deux sous-matrices. Une autre possibilité consiste à visualiser graphiquement les dynamiques. Nous allons passer en revue la méthode analytique et la méthode géométrique de mise en évidence des dynamiques.

4.2. Mise en évidence analytique des dynamiques

Soit l'équation d'état décomposée sous la forme suivante :

$$\begin{bmatrix}\dot{x}\\ \dot{z}\end{bmatrix} = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}.\begin{bmatrix}x\\ z\end{bmatrix} \quad \text{avec} \begin{cases} x \in R^n \\ z \in R^m \end{cases} \qquad [25]$$

x représentant l'ensemble des variables lentes et z l'ensemble des variables rapides.
La première étape consiste à bloc-diagonaliser le système [25].

Dans un premier temps, ce système est bloc-triangularisé par une transformation :

$$\begin{bmatrix}x\\ \tilde{z}\end{bmatrix} = \begin{bmatrix}I & 0\\ L & I\end{bmatrix}.\begin{bmatrix}x\\ z\end{bmatrix} \qquad [26]$$

I est la matrice identité et L une matrice à identifier, de dimensions adéquates.

Le modèle obtenu après la transformation est :

$$\begin{bmatrix}\dot{x}\\ \dot{\tilde{z}}\end{bmatrix} = \begin{bmatrix}\tilde{A}_{11} & A_{12}\\ \Re(L) & \tilde{A}_{22}\end{bmatrix}.\begin{bmatrix}x\\ \tilde{z}\end{bmatrix} \qquad [27]$$

avec
$$\begin{cases} \tilde{A}_{11} = A_{11} - A_{12}L \\ \tilde{A}_{22} = A_{22} + LA_{12} \\ \Re(L) = A_{21} + LA_{11} - LA_{12}L - A_{22}L \end{cases}$$

Le système est bloc-triangularisé si $\Re(L)=0$, c'est-à-dire si L est solution de l'équation de Riccati suivante :

$$A_{21} + LA_{11} - LA_{12}L - A_{22}L = 0 \qquad [28]$$

On effectue alors un nouveau changement de variables sur x :

$$\begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix} = \begin{bmatrix} I & -M \\ 0 & I \end{bmatrix} . \begin{bmatrix} x \\ \tilde{z} \end{bmatrix}$$

Par cette transformation, le système devient :

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{z}} \end{bmatrix} = \begin{bmatrix} \tilde{A}_{11} & \Re(M) \\ 0 & \tilde{A}_{22} \end{bmatrix} . \begin{bmatrix} \tilde{x} \\ \tilde{z} \end{bmatrix}$$

avec $$\Re(M) = A_{12} + \tilde{A}_{11}M - M\tilde{A}_{22}$$

Ainsi, l'existence d'une solution de l'équation de Lyapunov :

$$\Re(M) = A_{12} + \tilde{A}_{11}M - M\tilde{A}_{22} = 0 \qquad [29]$$

transforme le système initial [25] en un système bloc-diagonal.

L'existence et la solution aux équations [28] et [29] ont été étudiées par de nombreux auteurs [KOK99].

Détermination de la matrice L :

Plusieurs approches existent. Celle que nous exposons consiste à définir une suite :

$$\begin{cases} L_{k+1} = A_{22}^{-1}\left(A_{21} + L_k A_{11} - L_k A_{12} L_k\right) \\ L_0 = A_{22}^{-1} A_{21} \end{cases}$$

Cette suite converge vers une racine bornée, solution de [28], si les conditions suivantes sont remplies [KOK99] :

$$\begin{cases} A_{22} \text{ est inversible,} \\ \left\| A_{22}^{-1} \right\| << \left(\|A_0\| + \|A_{12}\| . \|L_0\| \right)^{-1} \end{cases} \quad \text{avec } A_0 = A_{11} - A_{12}L_0$$

La matrice L_0 est alors utilisée comme une approximation de L à l'ordre $o(\varepsilon)$, ce qui conduit au système suivant :

$$\begin{cases} \tilde{A}_{11} = A_{11} - A_{12}A_{22}^{-1}A_{21} + o(\varepsilon) \\ \tilde{A}_{22} = A_{22} + o(\varepsilon) \end{cases}$$

Détermination de la matrice M :

On définit cette fois-ci la suite :

$$\begin{cases} M_{k+1} = \left[(A_{11} - A_{12}L)M_k - M_k L A_{12}\right]A_{22}^{-1} + A_{12}A_{22}^{-1} \\ M_0 = A_{12}A_{22}^{-1} \\ L = L_0 + o(\varepsilon) \end{cases}$$

La solution à l'équation [29] est obtenue en cherchant le point de convergence de cette suite. Les conditions suffisantes d'existence d'une solution bornée sont les mêmes que précédemment. On a alors :

$$M = A_{12}A_{22}^{-1} + o(\varepsilon)$$

Suite à la transformation de la matrice A sous une forme bloc-diagonale, les normes des sous-matrices lente et rapide sont comparées via la relation [24] justifiant ou non la présence de deux échelles de temps.
Le critère d'existence de deux échelles de temps peut aussi être estimé en calculant le rapport :

$$\varepsilon = \frac{\left|\lambda_{max}(A_r)\right|}{\left|\lambda_{min}(A_1)\right|} \qquad \varepsilon \in]0, 1]$$

Nous verrons dans les paragraphes suivants comment appliquer l'algorithme de transformation d'une matrice A en une forme bloc-diagonale sur un processus de Markov à n macro-états et dont la différence des dynamiques est de l'ordre de r.

Nous allons à présent détailler la deuxième méthode, graphique, permettant de visualiser le découplage des dynamiques.

4.3. Mise en évidence graphique des dynamiques

La méthode proposée permet de détecter graphiquement l'existence de variables lentes et rapides par le tracé des **cercles de Gershgorin**. D'autres méthodes graphiques existent, dont les régions de Gudkov et les ovales de Cassini [BOU90] mais sont plus complexes et difficiles à appliquer. La méthode des cercles de Gershgorin est la plus intéressante, en raison de sa simplicité. Elle consiste à visualiser les valeurs propres de la matrice de transition dans le plan complexe. Les différentes dynamiques du système sont représentées par la présence de plusieurs domaines disjoints dans ce plan complexe.

Théorème de Gershgorin : [DAU85]
Notons $\{a_{ij}$; $i,j \in [1, N]\}$ les éléments d'une matrice A. Les valeurs propres de A appartiennent au domaine formé par l'union des cercles de rayon R_i et de centre g_{ii} (cercles de Gershgorin) définis par :

$$\sum_{\substack{j=1 \\ j \neq i}}^{N} \left|a_{ij}\right| = R_i \qquad \textbf{définition en ligne}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{N} |a_{ij}| = R_i' \qquad \textbf{définition en colonne}$$

$$\mathbf{g_{ii} = a_{ii}}$$

Si s cercles $C(g_{ii}, R_i)$ forment un domaine séparé des autres cercles, alors il y a exactement s valeurs propres à l'intérieur de ce domaine.

Soit D_1 le domaine du plan complexe formé par la réunion des régions intérieures aux N circonférences d'équations :

$$|z - a_{ii}| = R_i \qquad i \in [1, N]$$

et D_2 le domaine du plan complexe formé par la réunion des régions intérieures aux N circonférences d'équations :

$$|z - a_{ii}| = R_i' \qquad i \in [1, N]$$

Alors, l'ensemble des valeurs propres de A se situe dans l'intersection de ces deux domaines $D=D_1 \cap D_2$.

<u>Théorème</u> : Si l'on peut définir deux ensembles d'indice L et K avec $L \cap K = \varnothing$ et $L \cup K = \{1, 2, \ldots, N\}$ tel que $\forall (l,k) \in LxK$, les cercles $C_l(g_{ll}, R_l)$ et $C_k(g_{kk}, R_k)$ vérifient :

$$|a_{kk} - a_{ll}| >> (R_k + R_l) \qquad \forall\, l \in L, \qquad \forall\, k \in K$$

alors la matrice A possède deux ensembles de valeurs propres séparées. Le taux de séparation peut être estimé par le rapport :

$$\tau = \max_{\substack{k\in K \\ l\in L}} \left(\frac{R_l + R_k}{|a_{ll} - a_{kk}|} \right)$$

Ce taux doit être inférieur à 1.

La figure suivante illustre le cas où les cercles associés à une matrice A forment deux domaines séparés dans le plan complexe :

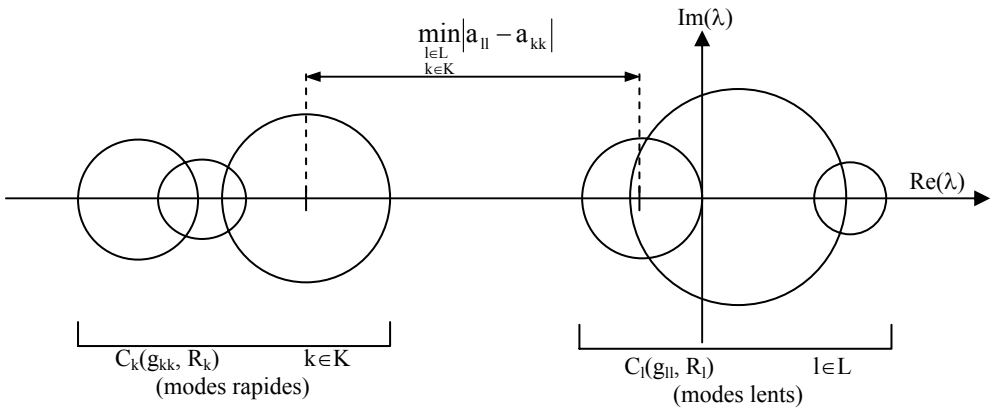


Figure 44 : Cercles de Gershgorin

Cependant cette méthode de détection des modes lents et rapides ne permet de conclure que si la matrice A est à diagonale dominante, ce qui est rarement le cas. Dans le cas des processus markoviens, les éléments de la diagonale de la matrice de transition A sont les compléments à

zéro des lignes correspondantes ; tous les cercles de Gershgorin sont donc tangents à l'origine. Ils ne font pas apparaître deux domaines bien distincts dans le plan complexe. Il est nécessaire alors de **conditionner** la matrice A.
Des **algorithmes de calibrage** [WIL85], [BOU90] consistent à appliquer un changement de base diagonal. Il permet de calibrer des termes hors diagonaux afin de minimiser les rayons des cercles.

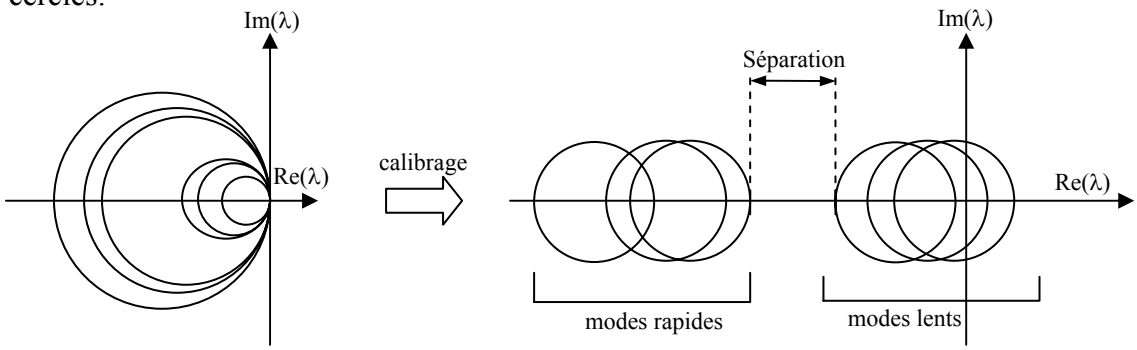


Figure 45 : Calibrage des cercles

Calibrage de la matrice A :

La méthode de calibrage proposée par [WIL85] minimise la norme infinie d'une matrice carrée d'ordre N formée par les éléments hors diagonaux de la matrice. Elle utilise un algorithme d'homogénéisation des cercles de Gershgorin et effectue un changement de base diagonal D.

$$D = \prod_{j=0} D_j$$

Avec D_0 la matrice identité et $D_j = \begin{bmatrix} d_1^j & 0 & \cdots & 0 \\ 0 & d_2^j & \cdots & 0 \\ \cdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_N^j \end{bmatrix}$ pour j=1, 2, …

Chaque matrice D_j se calcule par itération. Pour une itération j donnée, l'élément d_i^j se calcule de la façon suivante :

- Si la somme des modules des éléments de la ligne i est maximale, aucune opération n'est effectuée.
- Sinon, on cherche la matrice $D = \begin{bmatrix} 1 & & & \\ & 1 & 0 & \\ & & d_i^j & \\ & 0 & & 1 \\ & & & & 1 \end{bmatrix}$ avec $0 < d_i^j < 1$. Cette transformation permet d'augmenter la somme des éléments de la ligne i et de réduire les sommes des modules maximales de façon à avoir l'égalité. Soir m l'indice de la ligne dont la somme des modules des éléments est maximale, d_i^j est solution de l'équation :

$$\left|a_{mi}^{j-1}\right|\left(d_i^j\right)^2+\left(\sum_{\substack{k=1\\k\neq i}}^{N}\left|a_{mk}^{j-1}\right|-\left|a_{mm}\right|\right)d_i^j-\sum_{\substack{k=1\\k\neq i}}^{N}a_{ik}^{j-1}=0 \qquad [30]$$

où a_{ik}^{j-1} représentent les éléments de la matrice $D^{-1}AD$. L'itération s'arrête lorsque les rayons des cercles de Gershgorin de la matrice $D^{-1}AD$ sont tous égaux avec une précision suffisante.

<u>Lemme</u> [BOU90] : l'équation [30] a une solution d_i^j unique telle que $0<d_i^j<1$. En plus, $d_i^j=1$ si et seulement si :

$$\sum_{k=1}^{N}\left|a_{ik}^{j-1}\right|=\sum_{k=1}^{N}\left|a_{mk}^{j-1}\right| \qquad [31]$$

L'application de cet algorithme sur une matrice de transition A non conditionnée permet de confirmer ou non l'existence de deux domaines séparés représentant les modes lents et rapides. Il est certain que plus les domaines sont nettement séparés, plus l'approximation des perturbations singulières sera correcte.

L'algorithme suivant décrit les différentes étapes de calcul de la matrice de passage diagonale D permettant la calibration des rayons des cercles d'une matrice de transition quelconque A :

```
A0 = A ;
D = In ;

Repeter

      Pour i=1 à n,

            m=indice de la ligne telle que la somme des   modules
               hors diagonaux est maximale
            Si [31] est vérifiée, alors di=1
                                  Sinon résoudre [30] pour 0<di<1

            Pour j=1 à n

                  Si j≠i alors aij=aij/di
                                  aji=aji*di
                  FinSi

            FinPour

            D=D*diag(di)

      FinPour

      Calculer Rayon_min(A) et Rayon_max(A)

Jusqu'à (Rayon_min/Rayon_max)>(1-precision)
```

4.4. Exemple illustratif

Considérons le processus de Markov suivant formé de n macro-états identiques, sauf éventuellement le dernier. Chaque macro-état M_i est composé de trois états élémentaires reliés entre eux par des transitions stochastiques α_i. Un taux de franchissement λ relie chaque macro-état M_{i-1} au macro-état M_i à partir des mêmes états élémentaires. Le graphe est donc composé de n-1 macro-états transitoires et un macro-état final.
La figure suivante illustre une partie du graphe :

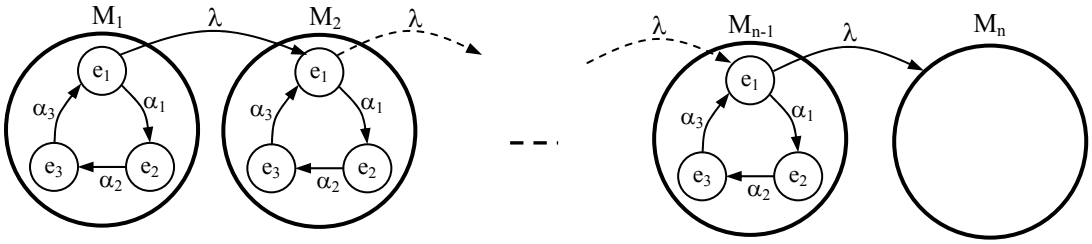


Figure 46 : Chaîne de Markov à n macro-états

La matrice de transition associée A est :

$$A = \begin{bmatrix} -\alpha_1 & \alpha_1 & 0 & \lambda & 0 & 0 & & & \\ 0 & -\alpha_2 & \alpha_2 & 0 & 0 & 0 & & & \\ \alpha_3 & 0 & -\alpha_3 & 0 & 0 & 0 & & & \\ 0 & 0 & 0 & -\alpha_1 & \alpha_1 & 0 & \lambda & & \\ 0 & 0 & 0 & 0 & -\alpha_2 & \alpha_2 & 0 & & \\ 0 & 0 & 0 & \alpha_3 & 0 & -\alpha_3 & 0 & & \\ & & & & & & -\alpha_1 & \alpha_1 & 0 & \lambda \\ & & 0 & & & & 0 & -\alpha_2 & \alpha_2 & 0 \\ & & & & & & \alpha_3 & 0 & -\alpha_3 & 0 \\ & & & & & & & & & 0 \end{bmatrix}$$

La taille de la matrice est 3(n-1)+1=3n-2 correspondant au nombre d'états élémentaires de la chaîne.

Dans l'hypothèse où $\lambda << \alpha_i$, $i=\{1,2,3\}$, la technique d'agrégation ramène la dimension de la matrice à n. En effet, la chaîne agrégée est de la forme :

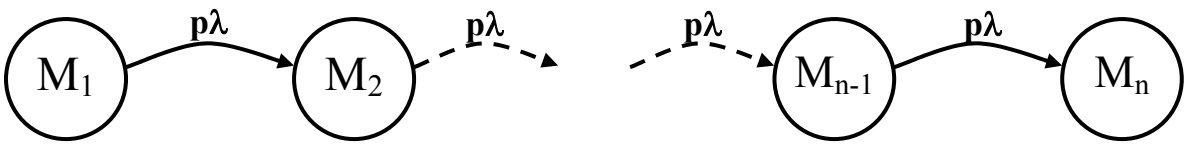


Figure 47 : Chaîne de Markov agrégée

Avec p la probabilité conditionnelle d'être dans l'état e_1 sachant qu'on est dans le macro-état M_i (toutes les probabilités conditionnelles sont identiques) :

$$p = \frac{\alpha_2\alpha_3}{\alpha_1\alpha_2 + \alpha_1\alpha_3 + \alpha_2\alpha_3}$$

La matrice de transition réduite associée est très simple :

$$A_r = \begin{bmatrix} -p\lambda & p\lambda & 0 & & \\ 0 & -p\lambda & p\lambda & & \\ & & \ddots & \ddots & \\ & 0 & & -p\lambda & p\lambda \\ & & & 0 & 0 \end{bmatrix}$$

(n lignes, n colonnes)

Des familles de processus vont être analysées. Ces processus sont définis par leur matrice de transition A(n,r) de taille 3n+1 et de paramètre r. Ce paramètre traduit le rapport entre les ordres de grandeur des échelles de temps.
A chaque matrice A(n,r) est associée une matrice réduite $A_r(n)$ de taille n+1 obtenue par découplage des dynamiques et application des perturbations singulières. La dynamique rapide ayant été simplifiée, la matrice réduite ne dépend plus du paramètre r (ou de $1/\varepsilon$ avec ε le petit paramètre à négliger).

Nous allons effectuer des comparaisons entre les familles de processus A(n,r) et les familles réduites associées $A_r(n)$:

- sur les temps nécessaires au calcul des probabilités de présence dans les macro-états,
- sur l'erreur commise entre les probabilités de présence dans le dernier macro-état (final)

Ces comparaisons sont réalisées en fonction de n et de r.

Comparaison des temps de calcul :

Les valeurs numériques des taux de transition sont choisies arbitrairement :

- pour les taux « rapides » : $\alpha_1=1$, $\alpha_2=10$, $\alpha_3=1$ (valeurs normalisées)
- pour le taux lent : $\lambda=1$

Les taux rapides réels de la chaîne de Markov sont multipliés par le facteur r.
Les temps de calcul sont évalués pour le processus initial et le processus réduit avec n variant de 10 à 60 et r variant de 1 à 10.
Pour chaque couple (n,r), le rapport entre les temps de calcul du processus initial et du processus réduit est évalué. Les résultats sont représentés graphiquement sur la figure 48.

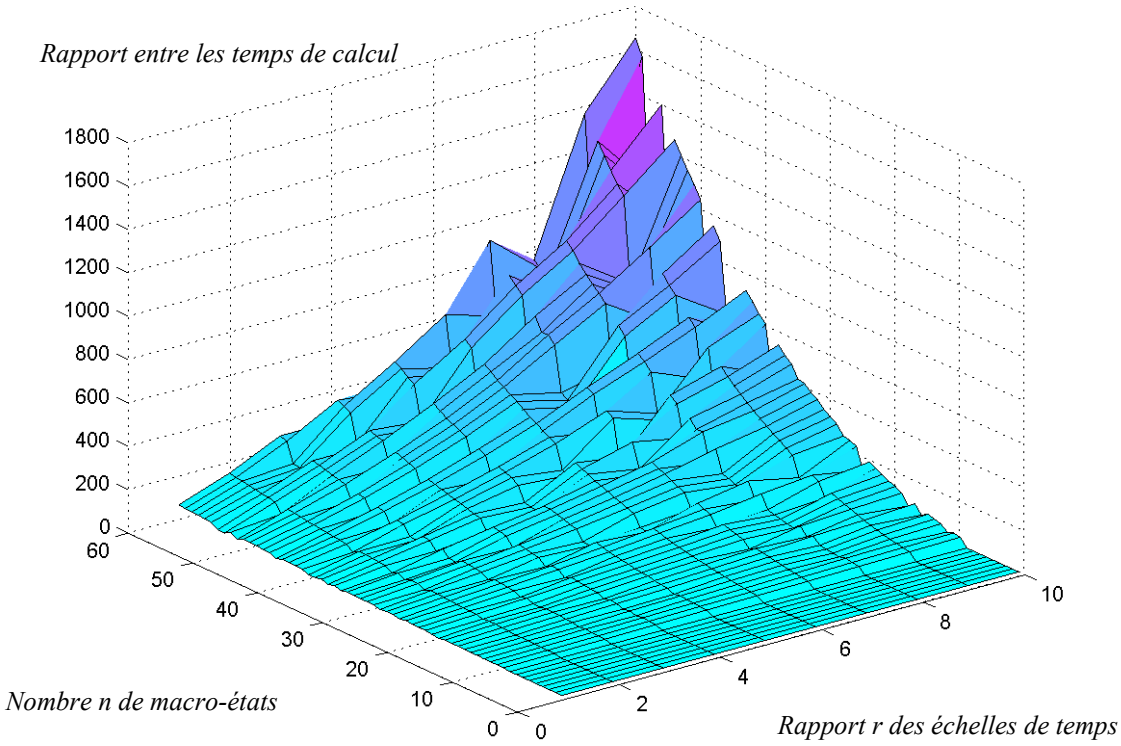


Figure 48 : Rapport des temps de calcul entre les deux processus en fonction de (n,r)

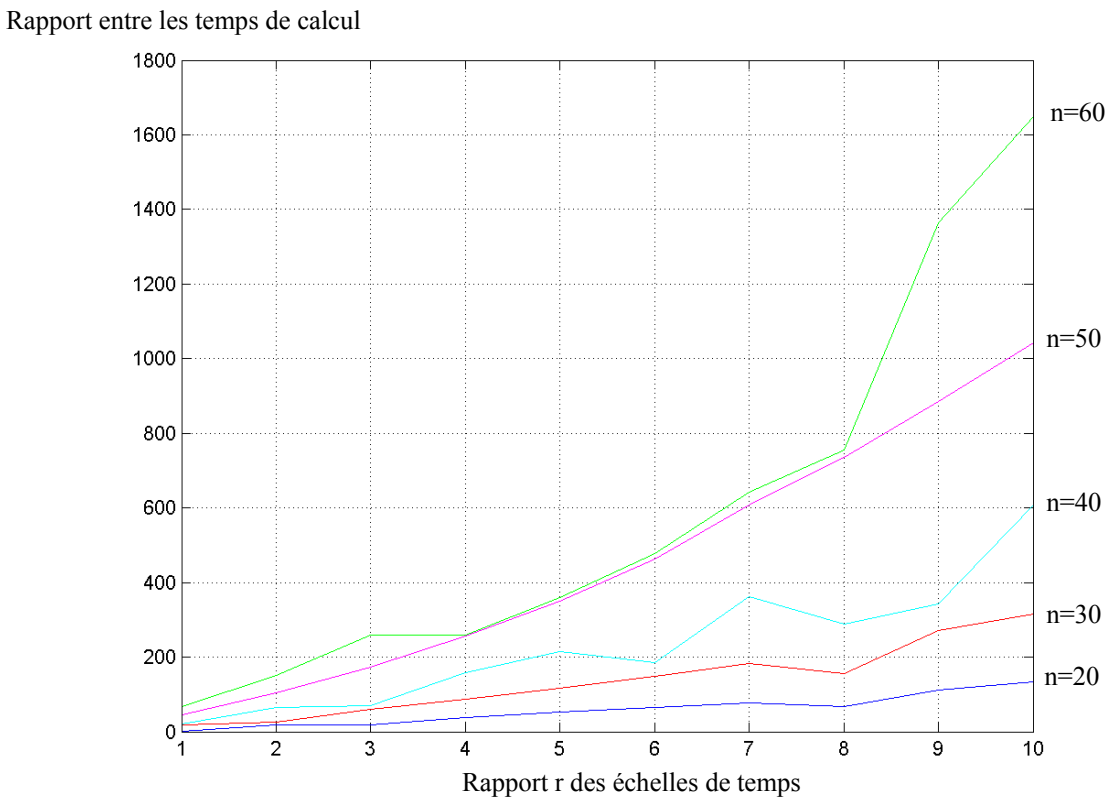


Figure 49 : Evolution du gain en temps de calcul en fonction de r pour n=20, 30, 40, 50, 60.

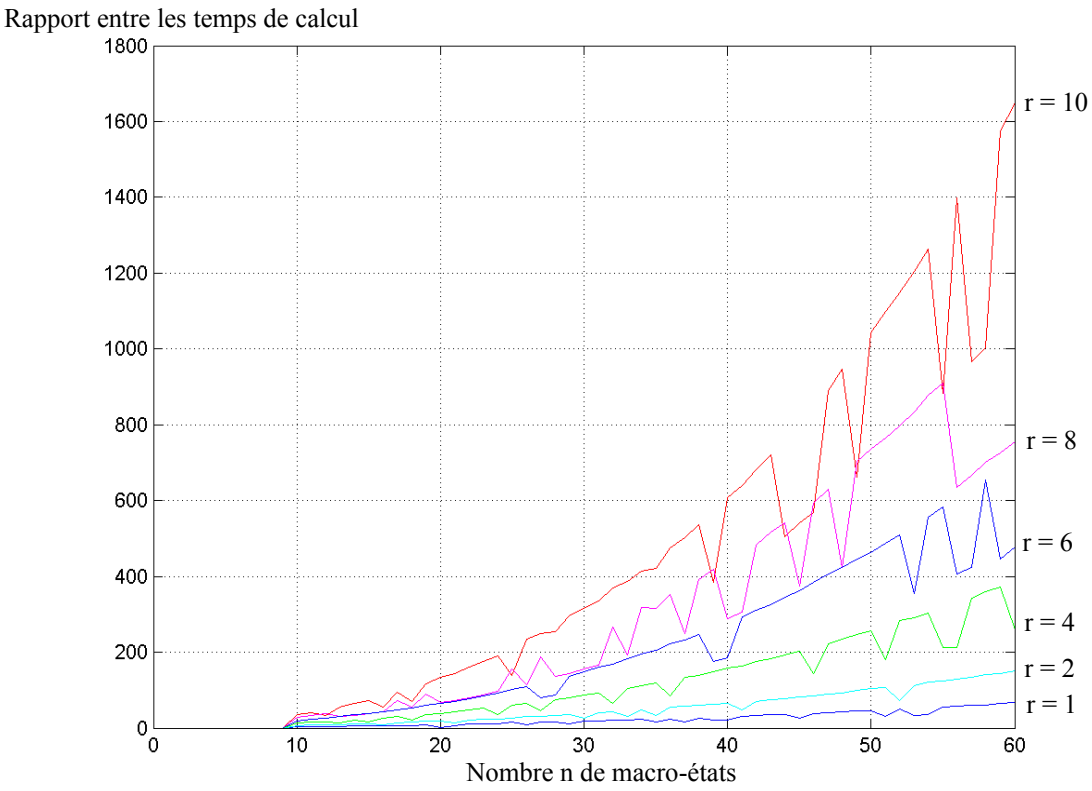


Figure 50 : Evolution du gain en temps de calcul en fonction de n

Les scripts ayant permis de mener ces analyses sont présentés en annexe 1.

Tous les calculs d'équations différentielles sont effectués avec un solveur à pas variable de Matlab Ode45 basé sur l'algorithme de Runge-Kutta.
Précisons que les mesures ont été effectuées sans justification de la validité des résultats pour le modèle réduit. Il est certain que pour des valeurs de r trop faibles, la technique des perturbations singulières ne peut plus être appliquée et le découplage des variables lentes et rapides ne peut plus être justifié sans commettre d'erreurs sur les probabilités.
Nous traiterons ultérieurement cette problématique appliquée à ces familles de processus : l'évaluation des erreurs de probabilités entre un processus à deux dynamiques et son équivalent réduit sera réalisée en fonction du couple (n,r).
Les résultats montrent nettement le gain en temps de calcul lorsque le nombre de macro-états augmente et/ou lorsque les dynamiques sont de plus en plus différentes. Il a été également constaté que le temps de calcul pour les processus non réduits est beaucoup plus sensible à la taille du graphe alors qu'il reste relativement constant pour les processus réduits associés.
Les gains sont d'autant plus importants que le rapport entre les dynamiques est élevé. En effet, le processus réduit ne possède qu'une dynamique et n'est donc pas sensible à r. Le processus initial possède deux dynamiques, le solveur devant prendre alors un pas de calcul suffisamment petit pour calculer précisément les variables rapides. La présence de variables lentes augmente considérablement le temps de calcul, puisqu'elles sont nécessairement évaluées à l'aide d'un pas de calcul très petit. C'est la même problématique que celle rencontrée lors de la simulation d'un système à deux échelles de temps.
Les valeurs de r qui ont servi à cette étude restent néanmoins raisonnables en comparaison de la différence des échelles de temps existant dans un modèle réel, et plus spécifiquement dans le cadre d'une étude de sûreté de fonctionnement. Les constantes de temps « rapides » propres

au système sont de l'ordre de la milliseconde (top d'horloge, …), de la minute (régulation, événements extérieurs,…) ou de l'heure (événements résultants d'actions du conducteur…), alors que les variables lentes sont associées à la dynamique des défaillances, de l'ordre de l'année. Le rapport des dynamiques est donc beaucoup plus grand, engendrant des temps de calcul beaucoup plus importants.

Estimation de l'erreur en fonction de (n,r) :

L'objectif de cette analyse est d'évaluer l'erreur absolue maximale de la probabilité d'arriver dans le macro-état final M_n entre le processus initial ($\dot{X} = A^T X$) et le processus réduit associé ($\dot{X}_r = A_r^T X_r$) en fonction des paramètres r et n. Les équations différentielles de l'exemple précédent sont résolues numériquement sur un intervalle de temps dépendant du couple (n, r). La largeur de cet intervalle varie de façon à ce que les régimes permanents puissent être visualisés. Les scripts Matlab sont détaillés en annexe 1.
La figure suivante montre que l'erreur maximale absolue entre les deux courbes de probabilité $Pr(M_n)$ diminue rapidement lorsque la différence des deux dynamiques augmente. Cette constatation est conforme aux résultats attendus : la méthode des perturbations singulières s'applique d'autant mieux que le découplage des variables rapides et lentes est marqué. On remarque que l'erreur diminue également lorsque le nombre de macro-états augmente. La décroissance est rapide pour n<5. Cependant, pour n ≥5, l'erreur mesurée devient moins sensible à n, voire quasiment indépendante.
Globalement, même pour des valeurs (n, r) faibles, l'erreur absolue reste relativement faible.

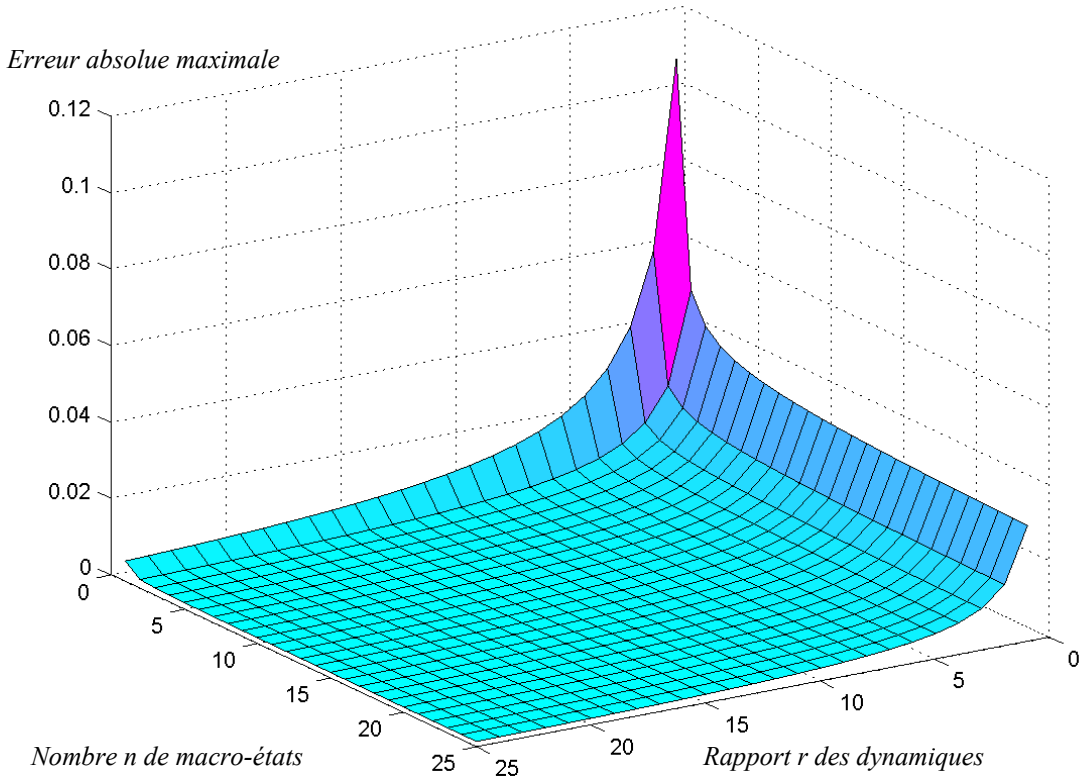


Figure 51 : Mesure de l'erreur absolue maximale de $Pr(M_n)$ en fonction de (n, r)

Etude de la séparabilité des dynamiques :

Cette nouvelle analyse a pour objectif de visualiser le taux de séparation des dynamiques en calculant le terme $\left\|A_r^{-1}\right\|.\left\|A_1\right\|$ en fonction de λ et du nombre n de macro-états transitoires. Les dynamiques lentes et rapides sont d'autant mieux découplées que le terme est inférieur à 1.

Cette fois-ci, pour des raisons de commodités, les taux rapides sont fixés et prennent les valeurs suivantes : α_1=100, α_2=1000, α_3=100.

La première étape consiste à rechercher une matrice de passage pour laquelle la matrice A est sous forme bloc-diagonale (par une résolution des équations de Riccati). Les scripts Matlab associés sont présentés en annexe 1.

On constate sur les figures 52 et 53 une croissance assez rapide du critère en fonction de λ sur une première partie de la courbe. Cette évolution traduit une diminution de l'écart entre les deux dynamiques, puisque tous les taux de transitions « rapides » et « lents » se rapprochent jusqu'à atteindre le même ordre de grandeur.
Puis à partir d'une certaine valeur de λ, le critère croît plus lentement, voire décroît pour de faibles valeurs de n.
Cependant, quelle que soit la valeur de n, il atteint globalement le même ordre de grandeur que l'unité. Le découplage des dynamiques n'est plus très marqué, voire inexistant. L'approximation d'ordre 0 des perturbations singulières n'est alors plus applicable sans commettre des erreurs de simplification.

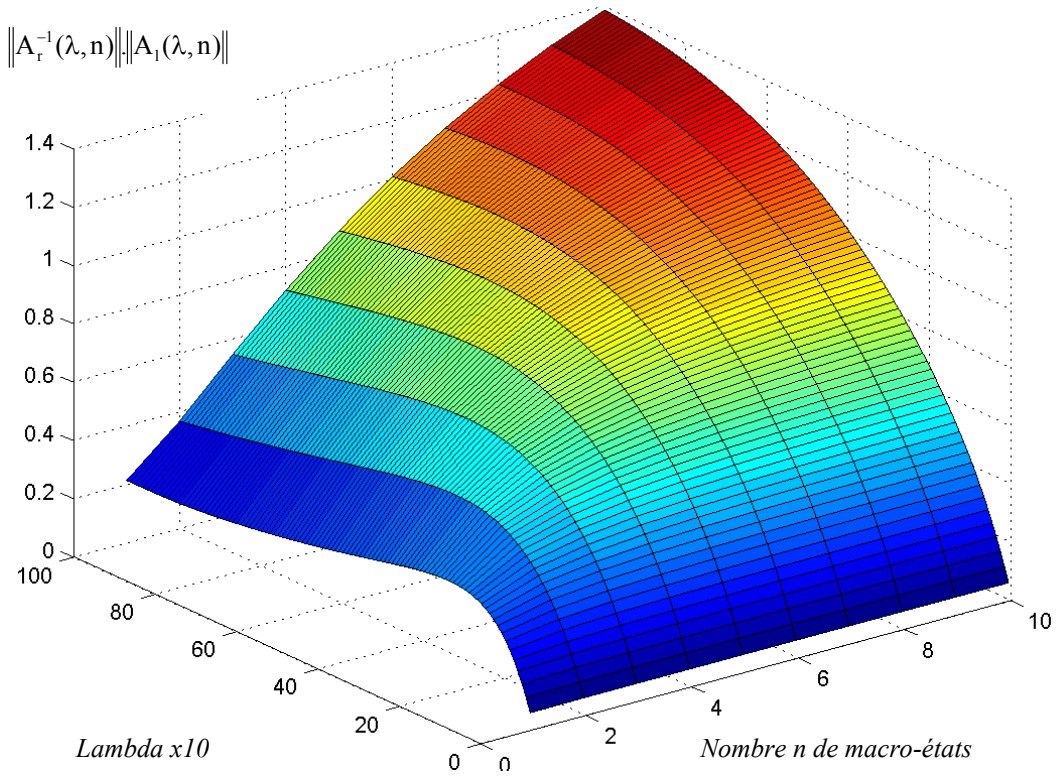


Figure 52 : Evolution du critère de séparabilité

$\left\|A_r^{-1}(\lambda,n)\right\|.\left\|A_l(\lambda,n)\right\|$

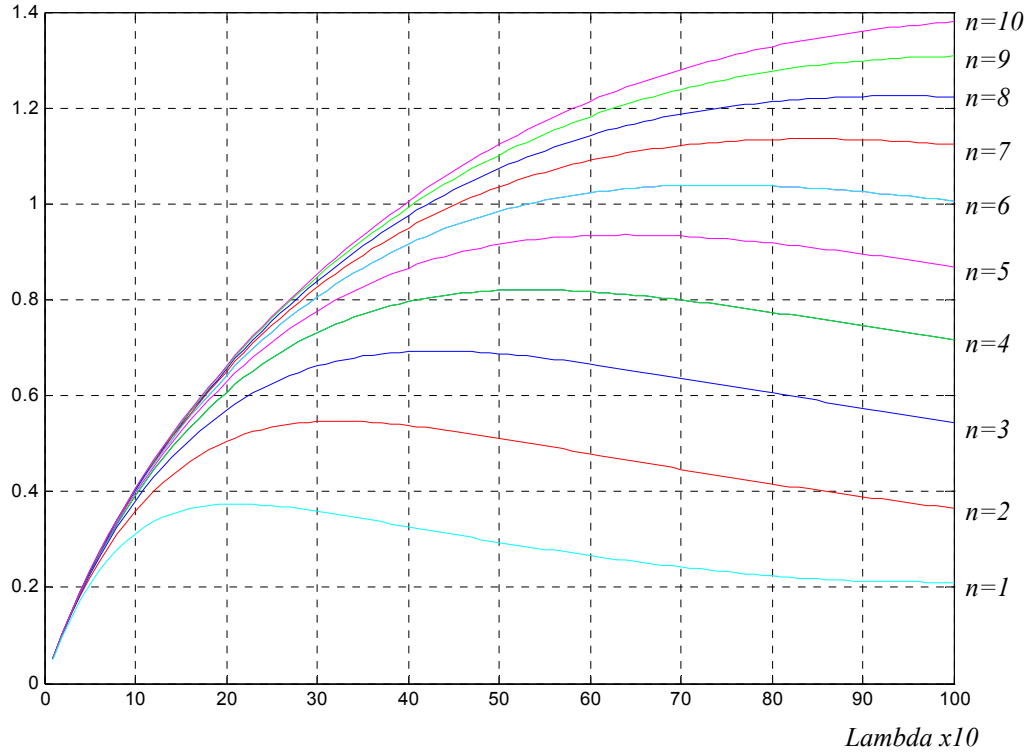


Figure 53 : Evolution du critère de séparabilité pour différentes valeurs de n

Le tracé des cercles de Gershgorin calibrés donne également une indication sur le taux de séparation des dynamiques. Par exemple :

- <u>Pour n=10 et λ=10</u> :

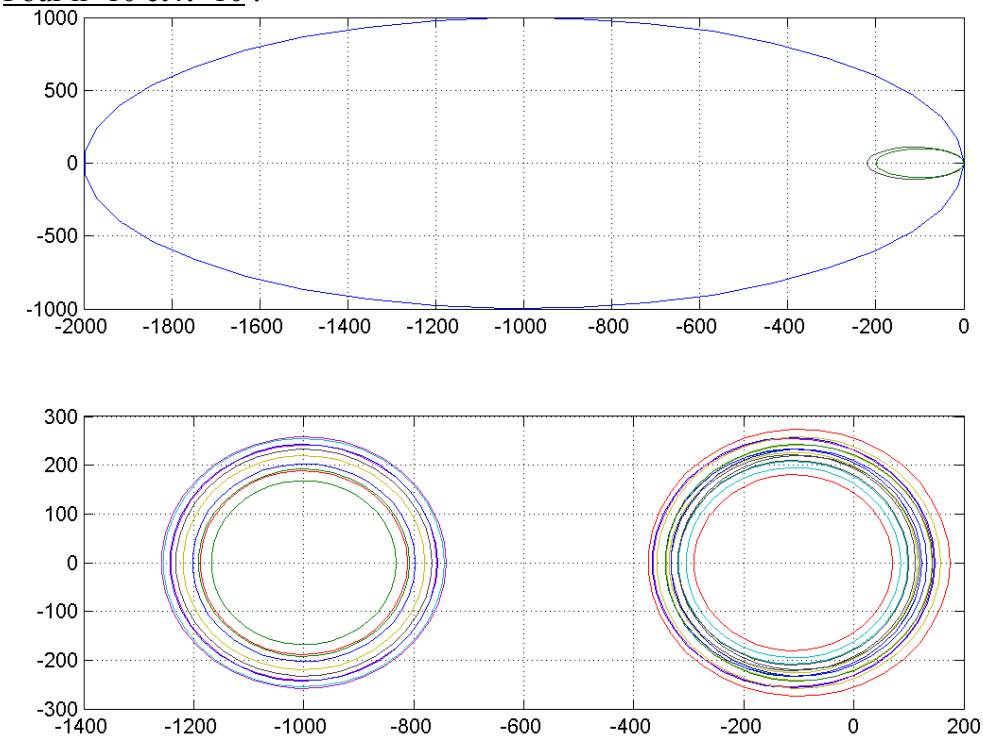


Figure 54 : Cercles de Gershgorin calibrés (n=10, λ=10)

La figure 54 présente les cercles associés à la matrice A(n=10, λ=10) avant et après calibrage. Deux classes de cercles nettement séparées montrent que les dynamiques lentes et rapides sont bien découplées. Pour A(n=10, λ=600), le tracé des cercles donne :

- Pour n=10, λ=600 :

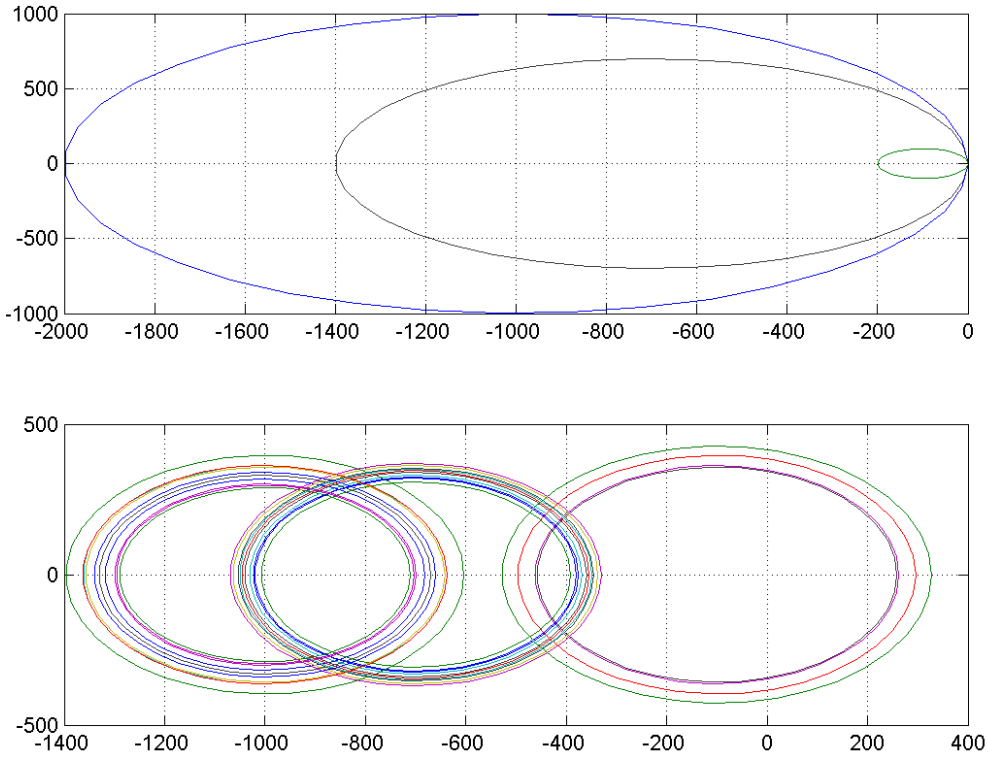


Figure 55 : Cercles de Gershgorin calibrés (n=10, λ=600)

Manifestement dans ce dernier cas de figure, le découplage des dynamiques n'est pas suffisamment marqué pour légitimer l'application des perturbations singulières. En effet, les taux de transitions sont tous du même ordre de grandeur.
Dans cette étude de cas, le rapport des dynamiques a intentionnellement été affaibli afin d'illustrer les différentes configurations pouvant se présenter. Dans le domaine de la modélisation des systèmes et de la sûreté de fonctionnement, l'écart est beaucoup plus marqué.
Néanmoins, l'évaluation d'un critère de séparabilité reste fastidieux, voire difficile à effectuer. En effet, elle nécessite d'un part de connaître le graphe de Markov exhaustif, et d'autre part d'effectuer un changement de base adéquat à partir de la matrice de transition afin d'identifier les composantes lentes et rapides.
Cependant, la principale difficulté reste encore de construire un graphe d'état entièrement probabilisé, puisque les systèmes modélisés sont par nature mixtes : déterministe, pour la partie commande et stochastique puisque les phénomènes aléatoires sont gouvernés par l'occurrence de défaillances.
Nous allons à présent améliorer l'applicabilité de la méthode d'agrégation en construisant un graphe de Markov réduit **directement à partir d'une représentation du système dans un formalisme de plus haut-niveau** ; cette phase doit permettre d'une part de se baser sur un modèle beaucoup plus parlant pour le concepteur-analyste grâce à un meilleur pouvoir d'expression et d'autre part de s'affranchir de certaines limites propres aux méthodes markoviennes.

5. Exemple élémentaire d'une architecture fonctionnelle/ matérielle

L'exemple proposé [SCH03c], d'ordre didactique, représente un système élémentaire de contrôle-commande. Un réseau de Petri interprété de commande, composé des places P_1, P_2, P_3 et P_4, modélise l'architecture fonctionnelle du système en fonctionnement nominal (cf. annexe 3). A ces places sont associées des actions ou opérations non décrites ici. Des événements de nature quelconque sont associés aux transitions T_1, T_2, T_5, T_4 et T_7 dont les franchissements permettent à la fonction de passer d'une opération à une autre.
Cette architecture fonctionnelle est indépendante de l'architecture matérielle sur lequel sera implémenté l'algorithme de commande. La fonction évolue à son propre rythme, échangeant avec l'environnement des informations via des capteurs et des actionneurs.
La fonction est supposée être implémentée sur un calculateur principal C_1.
Après avoir spécifié le comportement du système en réseau de Petri, le concepteur souhaiterait évaluer la fiabilité de son système de commande sur le calculateur et, éventuellement, apporter des solutions visant à l'améliorer si les exigences du cahier des charges ne sont pas respectées. Ces solutions peuvent être d'ordre fonctionnel ou matériel [AUB87].
L'approche que nous proposons consiste dans un premier temps à créer un modèle global en réseau de Petri sur lequel s'appuiera l'analyse quantitative. Il est composé d'une partie fonctionnelle spécifiant le comportement du système et d'une partie dysfonctionnelle. Cette dernière consiste à formaliser l'occurrence des défaillances de composants susceptibles de tomber en panne. Ce sont typiquement les éléments physiques de l'architecture matérielle : actionneurs, capteurs, calculateurs.
La seconde étape consiste à intégrer dans le modèle l'interaction entre les parties fonctionnelle et dysfonctionnelle afin de reproduire le comportement global du système en présence de défaillances.

Modélisation des défaillances :

Les réseaux de Petri stochastiques répondent à nos besoins de modélisation. Le franchissement d'une transition stochastique permet de représenter le changement de l'état d'un composant vers un état de défaillance. Nous supposons que des distributions de loi exponentielle sont associées à ces transitions stochastiques.
L'interaction entre les modèles fonctionnel et dysfonctionnel est représentée à l'aide de transitions immédiates. Un exemple est donné en figure 56 :

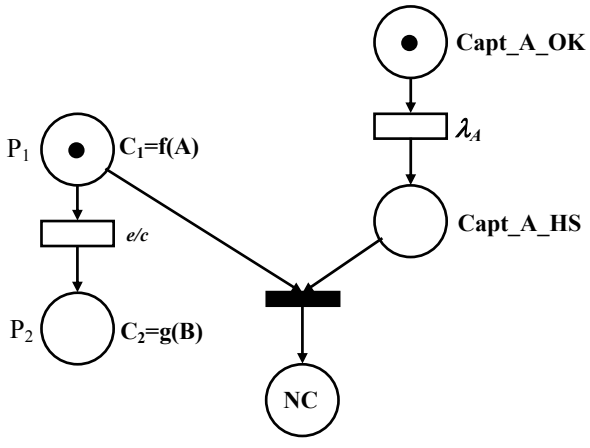


Figure 56 : Modélisation d'une défaillance

L'opération affectée à la place P_1 élabore une consigne C_1 (consigne actionneur par exemple) à partir d'une information mesurée par un capteur A. Son état est formalisé par les places Capt_A_OK et Capt_A_HS représentant respectivement l'état de marche et de défaillance.

L'occurrence d'une défaillance par le franchissement de la transition stochastique amène le système dans l'état de *Non-Conformité NC* si la fonction exploite à cet instant précis l'information de A. Formellement, le marquage simultané des places P_1 et Capt_A_HS sensibilise la transition immédiate, qui se traduit par un passage du système dans l'état de dysfonctionnement.

Une étude de fiabilité, à partir d'une telle représentation, repose sur une mesure quantitative de la probabilité d'être à un instant t dans la place NC.

Etude de différentes solutions fonctionnelles/matérielles :

Quatre variantes du modèle initial sont construites afin de prendre en compte différentes possibilités de mécanismes de tolérance aux fautes. Les modélisations en réseau de Petri sont présentées en annexe 3.

- **Modèle 1** : c'est le modèle initial. Aucune redondance fonctionnelle n'est implémentée et aucune redondance matérielle n'est envisagée.

- **Modèle 2** : un calculateur de secours C2 est employé. Cette redondance matérielle permet de tolérer une panne due au calculateur C1. Le même algorithme de commande est implanté sur les deux calculateurs. En cas de défaillance de C1, le calculateur C2 est activé (redondance passive) et reprend l'opération en cours grâce à une mémorisation de l'état de la commande au moment de la panne de C1.

- **Modèle 3** : une reconfiguration logicielle est étudiée. Elle permet de tolérer une panne du capteur A. En cas de défaillance, le système passe dans un mode dégradé, modélisé par la branche {T3, P5, T6}. Le fonctionnement normal {T4, P4, T7} est alors inhibé. Toutefois, si l'occurrence de la défaillance de A a lieu pendant que le système utilise l'information de ce capteur (jeton dans la place P4), alors le système passe dans un état non conforme (jeton dans NC).

- **Modèle 4** : la redondance du calculateur C1 et la reconfiguration logicielle sont prises en compte.

La figure suivante montre une représentation du modèle 4 ; deux parties sont à distinguer :

- **la partie fonctionnelle** : celle-ci décrit l'algorithme de contrôle-commande du système. Les opérations associées aux places ne sont pas détaillées.

- **la partie dysfonctionnelle** : elle est construite à l'aide des réseaux de Petri stochastiques pour représenter l'ensemble des défaillances de composants, et permet d'effectuer les mesures de performance en termes de fiabilité (ou de disponibilité).

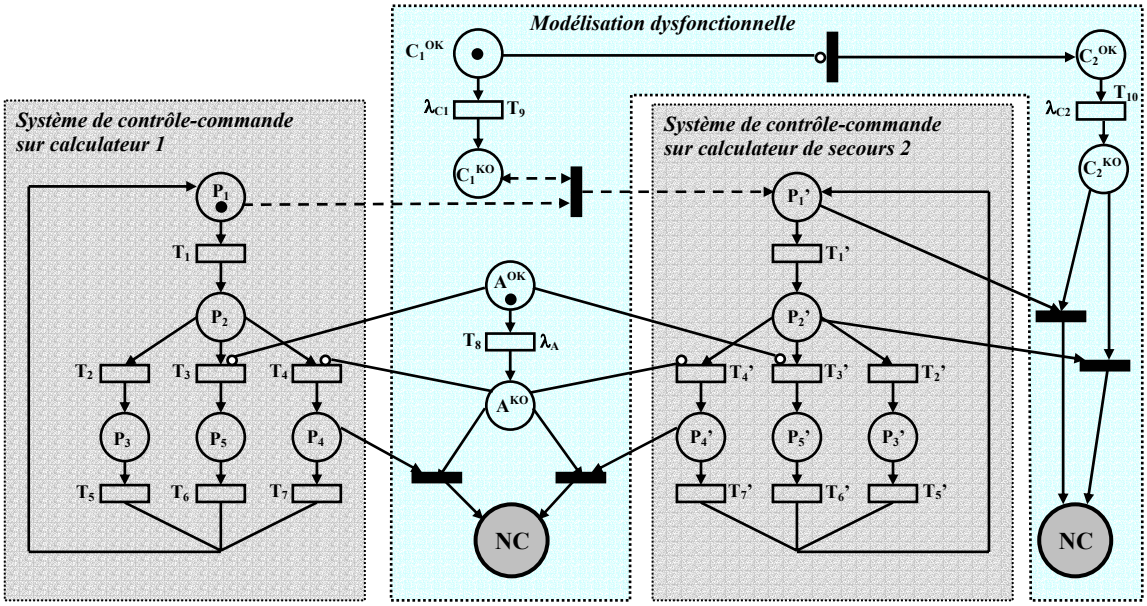


Figure 57 : Modèle 4 du système de contrôle-commande

Nous faisons l'hypothèse que seuls le capteur A et le(s) calculateur(s) sont susceptibles de tomber en panne.
Une étude analytique classique de l'évaluation des grandeurs de sûreté implique la construction du graphe des marquages afin d'en déduire un processus markovien. Dans l'exemple présenté, les transitions de la partie fonctionnelle ne sont pas stochastiques, puisque l'évolution est liée à des phénomènes déterministes. Ces transitions permettent de synchroniser le système de contrôle-commande avec le processus commandé. Pour simplifier dans un premier temps l'explication de la démarche, nous associons à ces transitions T_i un taux de franchissement exponentiel α_i comme le montre la figure suivante :

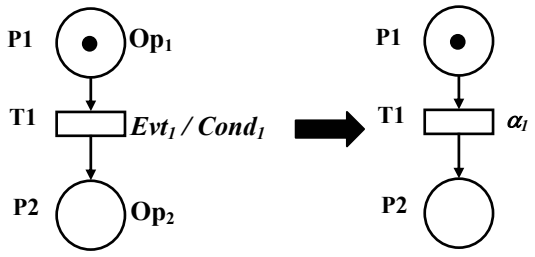


Figure 58 : « Stochastisation » des transitions

Le fonctionnement de la partie contrôle-commande étant régi par une dynamique beaucoup plus rapide que la dynamique des défaillances, nous faisons l'hypothèse que :

$$\alpha_i >> \lambda_j \qquad \forall\ i,j$$

La connaissance du graphe des marquages pour chaque modèle permet d'établir le graphe de Markov associé. Celui du modèle 4 est présenté en annexe 3. Bien que la taille de ce graphe reste raisonnable (17 états), une augmentation de la complexité ou du niveau de détails aura pour effet d'accroître sensiblement le nombre d'états, voire de générer une explosion combinatoire, d'où l'intérêt d'une démarche d'agrégation des états du graphe.

Sur le graphe de Markov exhaustif présenté en annexe 3, deux dynamiques coexistent. Ces dynamiques sont induites par la présence des taux α_i rapides et des taux λ_j lents. Nous allons découpler ces dynamiques et réduire la taille de l'équation d'état en identifiant l'ensemble des classes d'états constituant des composantes fortement connexes.
La première étape consiste à éliminer l'ensemble des arcs relatifs aux taux de défaillance. Il apparaît alors quatre composantes fortement connexes :

- composante 1 : états {1, 2, 3, 4}
- composante 2 : états {5, 6, 7, 8}
- composante 3 : états {9, 10, 11, 12}
- composante 4 : états {13, 14, 15, 16}

En plus de la composante triviale, formée de l'état 17, relatif à NC (état absorbant).

Manifestement, une interprétation physique se dégage de ces sous-graphes : chaque composante s'identifie à un mode de fonctionnement nominal ou dégradé. Nous avons ainsi :

- **composante 1** ≡ mode nominal MN (aucune défaillance),
- **composante 2** ≡ mode dégradé MD1 (reconfiguration matérielle sur le calculateur C_2 suite à une défaillance de C_1),
- **composante 3** ≡ mode dégradé MD2 (reconfiguration logicielle suite à une défaillance du capteur A),
- **composante 4** ≡ mode dégradé MD3 (reconfigurations matérielle et logicielle).

Chaque sous-graphe contient uniquement des taux de transitions rapides, relatifs à des modes de fonctionnement issus de l'algorithme de contrôle-commande. Les sous-graphes sont reliés entre eux par des taux de défaillance, amenant le système d'un état nominal ou dégradé, à un état davantage dégradé ou de non-conformité.
Nous pouvons alors établir le graphe de Markov réduit à partir de ces considérations. Les macro-états (dont nous cherchons les probabilités) correspondent aux modes de fonctionnement identifiés. Les taux de transition équivalents sont issus de l'équation d'état réduite. Ils sont égaux à des combinaisons linéaires des taux de défaillance entre les macro-états pondérés par les probabilités conditionnelles (asymptotiques) d'être dans l'état élémentaire en amont des taux de défaillance considérés.

Le graphe de Markov agrégé est composé de cinq macro-états relatifs aux modes de fonctionnement nominal, dégradés et de dysfonctionnement. Celui-ci est présenté sur la figure 59 suivante :

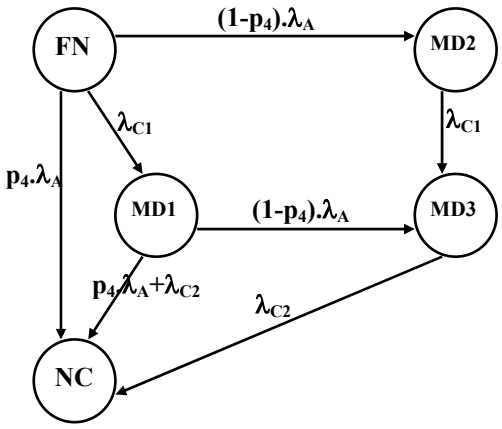


Figure 59 : Graphe de Markov agrégé

Le paramètre p_4 pondérant le taux de défaillance λ_A est la probabilité conditionnelle asymptotique que le système se trouve dans l'état 4, sachant qu'il est en fonctionnement nominal. En effet, une défaillance du capteur A peut avoir des conséquences différentes en fonction de l'instant d'occurrence de la défaillance :

- si la défaillance a lieu pendant que le système exploite l'information du capteur, c'est-à-dire si la place P_4 est marquée, alors le système passa dans l'état de Non Conformité.
- sinon, un mécanisme de reconfiguration logicielle fait passer le système dans le mode dégradé MD2.

Ainsi, la probabilité que le système arrive en non conformité sur défaillance du capteur est directement liée à la probabilité que la place P_4 soit marquée. Ce terme de pondération se retrouve également sur le taux de défaillance λ_A entre les macro-états MD1 et NC. Inversement, la probabilité que le système bascule dans la reconfiguration logicielle MD2 ou MD3 dépend du terme de pondération $(1-p_4)$.

Le calcul de l'ensemble des probabilités conditionnelles asymptotiques pour chaque composante fortement connexe nécessite d'extraire de la matrice de transition globale, les quatre sous-matrices propres à chaque composante, après avoir éliminé les taux de transition lents (approximation d'ordre 0 des perturbations singulières). La matrice globale et les sous-matrices sont présentées en annexe 3. En appliquant la méthode d'agrégation aux quatre variantes du modèle initial, nous obtenons des réductions du nombre d'états rassemblées dans le tableau suivant :

Modèle	Avant réduction	Après réduction
Modèle 1	8	3
Modèle 2	15	9
Modèle 3	9	3
Modèle 4	17	5

Figure 60 : Comparaison du nombre d'états avant et après agrégation

Après réduction, le nombre d'états se limite au nombre de modes de fonctionnement/dysfonctionnement. Pour le modèle 4, la forme agrégée est clairement plus lisible et parlante pour un concepteur, puisque les macro-états s'interprètent facilement. De plus, ce modèle constitue un bon support dysfonctionnel, à l'instar d'autres méthodes de représentation graphique permettant une visualisation de combinaisons de défaillances amenant le système vers un état redouté, comme les arbres de défaillance. L'avantage de la méthode proposée est de tenir compte de la dynamique du système par l'intermédiaire des coefficients de pondération et de l'ordre des défaillances. Cette approche vient donc combler l'insuffisance des méthodes statiques, tout en conservant une description graphique, très importante lors de la coopération de plusieurs protagonistes sur un même projet.
A titre illustratif, une comparaison de la fiabilité pour les quatre modèles est présentée en annexe 3.

Soulignons cependant que l'extraction des sous-matrices suppose que la matrice globale est connue ou encore que le graphe de Markov de tous les états élémentaires et les regroupements ont été identifiés. Le risque d'explosion combinatoire peut rendre cette tâche difficile. Néanmoins, les probabilités conditionnelles dans les états élémentaires sont directement liées aux probabilités de marquage du réseau de Petri. L'idée, pour éviter de construire le graphe de

Markov et établir directement le graphe agrégé, consiste à raisonner sur les probabilités de marquage. De plus dans la pratique, il ne doit pas être nécessaire de « stochastiser » les transitions du réseau de Petri décrivant l'architecture fonctionnelle pour en déduire les coefficients de pondération. Cette opération rend l'analyse quantitative trop lourde à mettre en oeuvre. Nous allons voir à présent comment les coefficients de pondération peuvent être facilement estimés.

5.1.1. Evaluation des coefficients de pondération à partir d'un réseau de Petri

Nous avons vu à travers l'exemple précédent que les termes de pondération issus du calcul des probabilités conditionnelles en régime permanent s'identifiaient à des probabilités de marquage. Ce rapprochement direct n'est possible que si le **réseau de Petri est sauf**, car dans le cas contraire, les taux de franchissement du graphe de Markov dépendent des taux de transition du réseau considéré et du marquage courant.
Nous supposons dans la suite que le réseau de Petri est sauf, hypothèse légitime dans le cas des réseaux de Petri de commande.

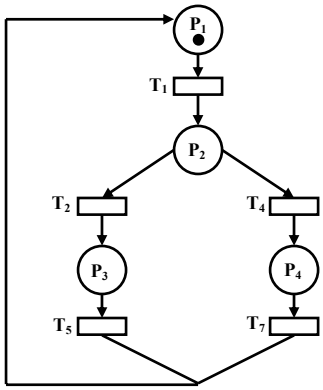
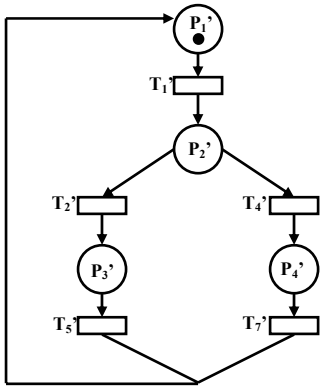
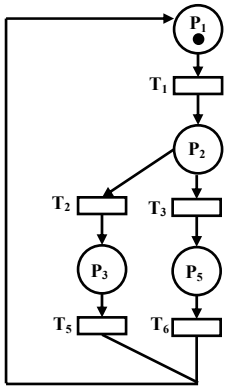
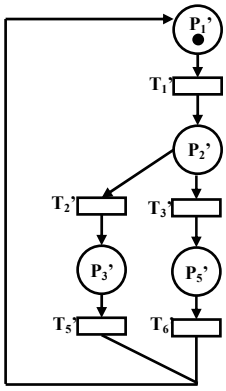
> *Définition : ergodicité du processus de marquage*
> *Le processus de marquage est ergodique s'il converge vers la même limite finie en moyenne temporelle et en moyenne probabiliste. On note cette limite M^* le vecteur de marque moyenne en régime permanent.*

Les processus de marquage relatifs à chaque mode de fonctionnement seront supposés par la suite être ergodiques. Le calcul des coefficients de pondération peut alors être effectué analytiquement dans le cas des réseaux de Petri stochastiques généralisés ; dans les autres cas, si l'analyse n'est pas envisageable, ces coefficients peuvent être évalués par **simulation**.

Revenons à l'exemple précédent. Le modèle 4 possède cinq macro-états, dont quatre caractérisent un fonctionnement nominal ou dégradé. Seuls ces derniers interviennent dans le calcul des coefficients de pondération, puisqu'ils correspondent à des classes d'états transitoires. L'approche proposée pour évaluer ces coefficients consiste à :

- construire pour chaque mode de fonctionnement nominal ou dégradé le réseau de Petri représentant uniquement le comportement dans ce mode,
- évaluer le marquage moyen dans chaque place de ces sous-réseaux, en supposant que le système y reste indéfiniment; en d'autres termes, aucune défaillance de composants ne doit pouvoir survenir. Dans le cas d'une modélisation par réseau de Petri interprété de commande, dans laquelle le comportement global du système est caractérisé par l'évolution simultanée de la partie discrète et la partie continue (partie opérative), les marquages moyens peuvent être estimés par une **simulation hybride**,
- pour chaque sous-réseau, identifier les places dont la probabilité de présence intervient dans le calcul des taux équivalents.

Le modèle 4 est donc composé des sous-réseaux suivants :

Mode de fonctionnement	**Architecture du sous-réseau associé**	**Marquage moyen à évaluer**	**Remarque**
Mode Nominal		- Place P_4	Le marquage moyen de P_4 fournit un indicateur de la durée de l'action associée. La probabilité d'arriver en NC sur occurrence d'une défaillance capteur A est d'autant plus grande que le marquage moyen en P_4 est grand
Mode Dégradé 1 (reconf. matérielle)		- Place P_4'	Idem pour P_4'
Mode Dégradé 2 (reconf. logicielle)		Aucun	L'évaluation des marquages moyens est ici inutile. En effet, quel que soit l'action en cours, la défaillance du calculateur C_1 aura la même conséquence (passage en mode dégradé 3).
Mode Dégradé (reconf. logicielle et matérielle)		Aucun	Idem que précédemment. Une défaillance de C_2 amène le système en Non Conformité quel que soit le marquage

En générant les graphes de marquage de ces sous-réseaux, on observe une correspondance avec les composantes fortement connexes du graphe initial.
Cette propriété de connexité doit être vérifiée systématiquement. Elle traduit d'un point de vue structurel, la propriété de **vivacité** du sous-réseau autonome associé. Nous supposons également que la vivacité est préservée après adjonction de l'interprétation ; en d'autres termes, le système modélisé possède toutes les « bonnes propriétés ». Cette hypothèse est légitime, si l'étude quantitative est précédée par une phase de vérification du modèle permettant notamment de s'assurer de l'absence de blocage mortel.
Les sous-graphes de Markov doivent donc posséder la propriété **d'ergodicité**. Pour chacun d'entre eux, il existe une distribution limite unique ne dépendant pas des conditions initiales.
L'analogie entre la distribution asymptotique des probabilités et le processus de marquage permet d'écrire, pour chaque mode de fonctionnement j:

$$\sum_{i\,/\,P_i \in \text{Mode j}} M^*(P_i) = 1$$

Où $M^*(P_i)$ représente le marquage moyen de la place P_i. Cette égalité est valide sous réserve bien entendu de l'unicité du jeton décrivant le comportement du système de contrôle-commande dans chaque mode.

L'utilisation de la simulation pour évaluer le marquage moyen dans chaque mode de fonctionnement ne pose pas de problème de temps de simulation. En effet, lors d'une analyse dyfonctionnelle quantitative traitée par une simulation classique de Monte-Carlo, la présence de deux dynamiques très différentes peut nécessiter des ressources en calcul importantes. La dynamique interne du système est caractérisée entre autres par l'évolution de l'ensemble des variables physiques continues et discrètes. Ces variables doivent être échantillonnées avec un pas de calcul adéquat, mais non adapté à la simulation des événements rares que sont les défaillances. De ce fait, la plupart des histoires jouées n'apportent aucune information [MON98].
Dans l'approche proposée, l'utilisation de la simulation a pour objectif de s'affranchir des hypothèses restrictives induites par les méthodes purement markoviennes. Le calcul du marquage moyen peut être effectué sur une représentation du système plus fine, ayant un pouvoir d'expression répondant mieux aux besoins de modélisation d'un concepteur. De plus, contrairement à la simulation de Monte-Carlo, seul le modèle fonctionnel est simulé, en l'absence de défaillances. L'exploitation de la technique des perturbations singulières permet de **découpler les dynamiques**. Alors que la dynamique rapide est traitée par la simulation, la dynamique lente est représentée par un modèle markovien agrégé, dont la résolution analytique de l'équation d'état associée fournit les grandeurs recherchées. Ce modèle dysfonctionnel est enrichi par des informations du système et de son environnement. Elles synthétisent de façon moyenne toutes les grandeurs ayant une influence sur l'occurrence des défaillances, autant qualitativement (en termes de conséquence) que quantitativement (par exemple la valeur du taux de défaillance variant en fonction de la température). Néanmoins, pour être évaluées, ces informations doivent être préalablement connues et intégrées dans la modélisation. Les grandeurs susceptibles d'influencer les taux de défaillance peuvent être de nature diverse : elles peuvent être continues, comme la température, ou discrètes, par exemple la position d'une vanne ou d'une pompe pouvant être en ouverture ou en fermeture. Par exemple, d'un point de vue qualitatif, le mode de défaillance de cet actionneur peut très bien dépendre de sa position courante lors de l'occurrence de la défaillance. Dans tous les cas, la grandeur influente doit pouvoir être représentée à l'aide d'un ensemble de places (dans le cas des réseaux de Petri), auquel cas le marquage d'une d'entre elles sera associé à une valeur ou

un ensemble de valeurs particulières de cette grandeur. Le graphe de Markov agrégé ne fera apparaître qu'un ou plusieurs arcs pondéré(s) par le(s) marquage(s) moyen(s) de ces places.
Pour conclure ce paragraphe, l'obtention des coefficients de pondération via une simulation comportementale de chaque mode opératoire est rapide, sous réserve d'existence d'une distribution limite du processus de marquage.

6. Conclusion

Les méthodes d'agrégation/simplification des graphes de Markov habituellement proposées dans la littérature se basent sur une formulation mathématique des regroupements d'état ; cette approche théorique est difficilement compatible avec les contraintes industrielles. De plus ces méthodes exploitent directement une représentation markovienne, ou des cas de figure pouvant se ramener à une telle représentation, sur laquelle figurent des transitions exclusivement exponentielles et un nombre d'états limité.

Or, dans le cadre d'une méthodologie de conception intégrant des analyses de sûreté de fonctionnement, le concepteur doit lui-même construire un tel graphe à partir d'un modèle comportemental de haut-niveau. La construction d'un tel graphe est une étape très délicate, puisque l'ensemble des dynamiques est loin d'être représentable avec les lois exclusivement exponentielles.

La méthode proposée dans ce chapitre a montré comment un graphe de Markov pouvait être déduit à partir d'un SDH. Les principales caractéristiques sont les suivantes :

- Chaque état est associé à un mode de fonctionnement. Leur nombre est ainsi considérablement limité. De plus leur identification est facilitée par une bonne connaissance fonctionnelle et comportementale du système.

- Les aspects relatifs à l'évaluation de la fiabilité d'un SDH sont également considérés. En effet, dans chaque mode de fonctionnement identifié, le comportement dynamique du système hybride interagit continûment avec le processus de défaillance (éventuellement différemment en fonction du mode considéré). L'évolution de l'ensemble des variables physiques suit une dynamique très rapide au regard de la dynamique des défaillances. Pour cette raison, d'après la théorie des perturbations singulières, ces variables physiques sont assimilables à leur valeur asymptotique et la dynamique globale du système est dominée par sa composante lente. L'influence de la dynamique rapide sur le processus de défaillance s'exprime ainsi sur le graphe d'états par l'intermédiaire de constantes, pondérant les taux de défaillance. Soulignons néanmoins qu'une bonne connaissance globale du comportement du système dans chaque mode est indispensable pour identifier les paramètres physiques ou fonctionnels qui interagissent sur l'évolution du processus de défaillance, et de quelle manière ces interactions se manifestent.

- La méthode proposée n'est pas tributaire d'un formalisme spécifique de représentation des SDH. Nous verrons d'ailleurs dans le chapitre 4, que d'autres formalismes (basés néanmoins sur une représentation à état/transition des systèmes de contrôle-commande) peuvent servir de support à la construction du graphe (en l'occurrence les Statecharts).

- Enfin, l'évolution dynamique du SDH est analysée dans chacun des modes de fonctionnement, afin de la caractériser par un ensemble de coefficients. On retrouve ainsi la définition de la fiabilité dynamique qui s'apparente à un *processus stochastique déterministe par partie*. Pour évaluer ces coefficients, une simple simulation est effectuée pour chaque mode, permettant ainsi de s'affranchir de la complexité des phénomènes modélisés (perturbations, non linéarités, …).

Pour résumer le principe général de la méthode, il ne s'agit ni plus ni moins d'un découplage de dynamiques. La simulation et l'analyse markovienne sont exploitées conjointement afin de contourner les limites intrinsèques à chacune de ces méthodes, et de profiter de leurs avantages.

Nous allons à présent mettre en œuvre concrètement cette méthode sur un cas d'application en utilisant les réseaux de Petri et les Statecharts comme formalismes de modélisation. Nous comparons également les performances de notre méthode avec la simulation de Monte-Carlo (temps de calcul et précision des résultats).

CHAPITRE 4

APPLICATION DE LA METHODE D'AGREGATION A UN CAS TEST

Chapitre 4

Application de la méthode d'agrégation à un cas test

1. Objectifs

L'objectif de cette partie est d'illustrer sur un cas test bien connu les différentes approches de modélisation d'un SDH et d'évaluation des probabilités d'occurrence d'événements redoutés. Ce cas d'application, largement traité dans la littérature, a été proposé par l'ISDF et a fait l'objet de publications ([DUT96], [DUT97]), puis a été traité dans la thèse de Jean-Luc Chabot [CHA98b]. Le système représente un réservoir dont on cherche à maintenir le niveau de liquide entre deux seuils à l'aide d'actionneurs appropriés et par l'intermédiaire d'une partie contrôle assurant la régulation. Cet exemple a pour principal intérêt de mettre en lumière les problématiques de sûreté de fonctionnement et de modélisation des SDH. Nous verrons en particulier l'influence du système sur l'occurrence et la nature des modes de défaillance. Réciproquement, l'occurrence de défaillances modifie le mode de fonctionnement de la régulation. Cette interaction s'effectue principalement par l'intermédiaire de la partie opérative constituée de paramètres physiques propres au procédé à commander. Il s'agit donc bien d'un problème de représentation et d'évaluation de la fiabilité dynamique d'un SDH.

Un formalisme adéquat doit être choisi afin de reproduire le comportement du SDH, et doit offrir la possibilité de d'intégrer les défaillances, support d'une analyse quantitative.

Les aspects modélisation et analyse dysfonctionnelle vont être traités sous plusieurs angles :

1. La modélisation en réseaux de Petri proposée par [CHA98b] sera reprise et adaptée à nos besoins,
2. Le système global sera ensuite formalisé par des blocs-diagrammes à l'aide de l'outil Simulink/Stateflow afin de comparer le pouvoir de modélisation des deux formalismes,
3. L'analyse quantitative sera déployée sur ces différentes représentations en comparant la simulation de Monte-Carlo et l'approche markovienne proposée dans ce mémoire. Nous soulignerons également la difficulté pour les méthodes statiques de traiter le problème sous les hypothèses considérées.

2. Présentation du système et hypothèses

2.1. Description du cas d'étude

Le processus physique à commander est un réservoir contenant du liquide. Le niveau h doit être maintenu dans un intervalle de sécurité [h_0-Δh ; h_0+Δh] avec h_0 le niveau initial.
La régulation du processus par la partie commande est assurée à l'aide de capteurs et d'actionneurs comprenant une pompe principale PO1 et une pompe de secours PO2. La vidange est effectuée à l'aide d'une vanne V.

2.2. Description du processus physique à commander

2.2.1. Caractéristiques matérielles du processus :

- La pompe principale PO1 a un débit moyen constant Q1=10 cm.mn^{-1}
- La pompe de secours PO2 a un débit moyen constant Q2=5 cm.mn^{-1}
- La vanne de vidange V a un débit moyen Q3=12 cm.mn^{-1}

La section du réservoir étant supposée constante, ces débits sont assimilés à des débits métriques.

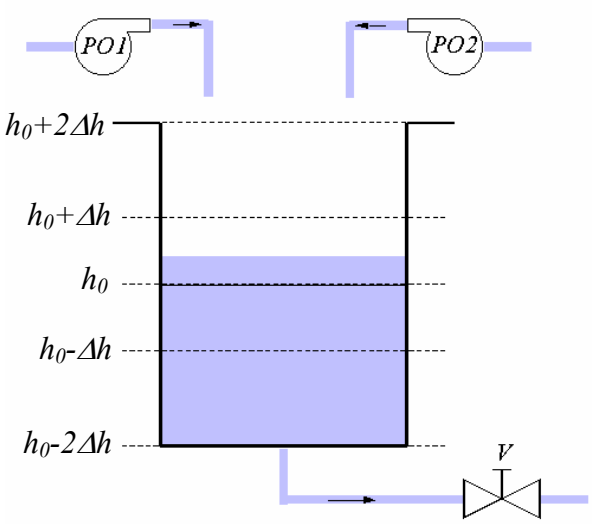


Figure 61 : Schéma du réservoir

2.2.2. Hypothèse de fonctionnement :

Le niveau initial dans le réservoir est h_0=100 cm. La valeur h permise du niveau est comprise dans l'intervalle h_0±Δh avec Δh=10 cm.

2.3. Description de la partie contrôle-commande

Initialement, le niveau h de liquide dans le réservoir est h_0. La pompe PO1 fonctionne, la vanne V est ouverte et la pompe de secours PO2 est fermée.
Des capteurs, non spécifiés ici, sont placés à différents niveaux du réservoir, dont la sollicitation provoque l'ouverture ou la fermeture des actionneurs en fonctionnement normal selon le tableau suivant :

Niveau h du liquide	Pompe 1	Pompe 2	Vanne
$h < h_0 - \Delta h$	Active	Active	Fermée
$h_0 - \Delta h \leq h \leq h_0 + \Delta h$	Active	Arrêtée	Ouverte
$h > h_0 + \Delta h$	Arrêtée	Arrêtée	Ouverte

Figure 62 : Commande des actionneurs en fonction du niveau h mesuré

2.4. Hypothèses dysfonctionnelles

Nous supposons que seuls les actionneurs (pompes et vanne) sont susceptibles de défaillir. Les défaillances des capteurs sont intégrées dans celles des pompes et de la vanne. Ces composants sont mutuellement indépendants et non réparables.
Les occurrences des défaillances des actionneurs suivent des lois exponentielles de taux constants : $\lambda_1=2{,}2831.10^{-3}$ h^{-1} pour la pompe PO1, $\lambda_2=2{,}8571.10^{-3}$ h^{-1} pour la pompe PO2, et $\lambda_3=1{,}5625.10^{-3}$ h^{-1} pour la vanne V. Pour chacun d'eux, plusieurs modes de défaillance sont envisagés. Ils sont répertoriés dans le tableau suivant :

Actionneurs	Modes de défaillance	Taux de défaillance
Pompe PO1	MdD1 : Blocage en ouverture	$\lambda_1=2{,}2831.10^{-3}$
	MdD2 : Blocage en fermeture	$\lambda_1=2{,}2831.10^{-3}$
	MdD3 : Blocage en position courante	$\lambda_1=2{,}2831.10^{-3}$
Pompe PO2	MdD1 : Blocage en ouverture	$\lambda_2=2{,}8571.10^{-3}$
	MdD2 : Blocage en fermeture	$\lambda_2=2{,}8571.10^{-3}$
	MdD3 : Blocage en position courante	$\lambda_2=2{,}8571.10^{-3}$
Vanne V	MdD3 : Blocage en position courante	$\lambda_3=1{,}5625.10^{-3}$

Figure 63 : Modes de défaillance des actionneurs

Remarquons que les modes de défaillance MdD3 dépendent de l'état du processus (en l'occurrence des actionneurs) à l'instant d'apparition de la défaillance et donc de la dynamique interne du système.

2.5. Evènements redoutés considérés

Le recensement de ces événements redoutés se fait habituellement à l'aide d'une analyse préliminaire des risques (APR). Les situations critiques dont on cherche à estimer la probabilité d'occurrence en fonction du temps sont :

- **ER1** : **Débordement du réservoir**. Nous supposons que cet Evènement Redouté se traduit par un niveau de liquide h supérieur à $h_0+2\Delta h$.
- **ER2** : **Assèchement du réservoir**. Cet événement correspond à un niveau h inférieur à $h_0-2\Delta h$.

3. Modélisation par réseaux de Petri

Le formalisme des réseaux de Petri interprétés ([DAV89]) se prête bien à la modélisation comportementale du système de contrôle-commande du réservoir et ne pose pas de problèmes majeurs. La partie contrôle-commande va permettre de réguler le niveau de liquide en envoyant des commandes d'ouverture et de fermeture aux différents actionneurs. Ces commandes sont élaborées en fonction de l'état actuel du processus à commander, c'est-à-dire en fonction du niveau h atteint par le liquide (cf. tableau de la figure 62). Cette grandeur continue est mesurée par des capteurs et exploitée par le système de commande. Le processus commandé (évolution du niveau de liquide) doit également être formalisé, ainsi que les interactions avec la partie discrète. J-L CHABOT propose dans [CHA98b] une modélisation de la partie continue à base de réseaux de Petri classique. Le principe consiste à discrétiser la variable continue h et à la représenter par une quantité de jetons dans une place. On peut par exemple imposer qu'une variation élémentaire de hauteur δh=1 cm soit modélisée par la disparition ou l'apparition d'un jeton dans une place associée au réservoir. On en déduit facilement que le débit Q1 de la pompe se traduit par l'apparition d'un jeton dans la place associée au réservoir toutes les 0,1 mn, soit toutes les 6 secondes ; le débit Q2 se traduit par l'apparition d'un jeton toutes les 3 secondes, et enfin le débit Q3 par la disparition d'un jeton toutes les 5 secondes. Initialement, le niveau du réservoir est de 100 cm. Ainsi, la place associée au réservoir est marquée de 100 jetons.
On construit alors plusieurs réseaux de Petri modélisant l'état du réservoir, l'état des actionneurs et le système de contrôle-commande. Ces différents modèles communiquent entre eux à l'aide de messages booléens. L'annexe 4 présente ces modèles.
Notons l'existence de composantes conservatives (P semi-flot) : M(V_f)+M(V_o)=1, M(PO1_o)+M(PO1_f)=1 et M(PO2_o)+M(PO2_f)=1. Ces trois composantes signifient que chaque actionneur est soit dans la position ouverte, soit dans la position fermée.
Enfin, la détection des seuils de niveau est effectuée grâce à des tests de marquage, à l'aide d'arcs pondérés et d'arcs inhibiteurs. Un nombre de marques inférieur à 80 jetons ou supérieur à 120 jetons dans la place du réservoir correspond respectivement à l'état d'assèchement et de débordement.
L'outil MOCA-RP V.10 a servi de support à la modélisation du système (cf. [MOCA]). Cet outil de modélisation et de simulation de réseaux de Petri permet notamment de faire communiquer plusieurs modèles indépendants en les synchronisant à l'aide de messages booléens reçus et envoyés sur les transitions. A ces transitions peuvent être associées des temporisations déterministes ou stochastiques. Il permet également d'évaluer des performances d'un système, notamment dans le domaine de la sûreté de fonctionnement,

grâce à la technique de simulation de Monte-Carlo. La nouvelle version MOCA-RP V.12 a un pouvoir de modélisation plus grand, grâce à la possibilité de manipuler des variables de type entier.
Le lecteur désireux de connaître les détails de la modélisation du réservoir complet pourra se référer à [CHAB0], [CHAB1], [DUT96], [SCH03b] et consulter l'annexe 4.

4. Modélisation Hybride par Simulink-Stateflow

Nous allons à présent exploiter le formalisme des Statecharts pour modéliser la partie discrète associée au système de contrôle-commande. Le processus physique et sa dynamique seront quant à eux modélisés à l'aide de blocs-fonctionnels afin de représenter l'évolution des variables continues par des équations différentielles. Dans le cas présent, l'évolution de ce processus est caractérisée par le niveau h de liquide dans le réservoir. L'équation d'évolution associée est très simple compte-tenu des hypothèses énoncées :

$$\frac{dh(t)}{dt} = \delta_1(t).Q_1(t) + \delta_2(t).Q_2(t) + \delta_3(t).Q_3(t)$$

Cette équation généralisée reflète les différents modes opératoires possibles du processus. Elle fait apparaître l'influence des phénomènes discrets sur l'évolution du processus au travers des termes δ_i . Ces derniers peuvent prendre la valeur 1 si l'actionneur associé est commandé en ouverture ou si une défaillance de cet actionneur le bloque dans la position ouverte et la valeur 0 dans le cas contraire.

L'outil Matlab/Simulink/Stateflow servira de support à la modélisation mixte. L'outil Simulink sera utilisé pour modéliser la partie continue à l'aide de blocs-diagrammes et le système discret sera formalisé par les Statecharts grâce à l'outil Stateflow. Nous allons à présent détailler les différents éléments de modélisation.

4.1. Modélisation de la partie contrôle-commande

Ce système discret est représenté par un automate Stateflow. Son rôle est d'envoyer aux actionneurs PO1, PO2 et V les commandes adéquates d'ouverture ou de fermeture en fonction de la valeur mesurée du niveau dans le réservoir.
Cet automate est composé de trois états relatifs à un fonctionnement nominal. La transition d'un état à un autre est franchie si un seuil caractéristique est atteint. Dans chacun d'eux, des actions spécifiques sont effectuées en générant des commandes aux actionneurs. Ces trois états sont :

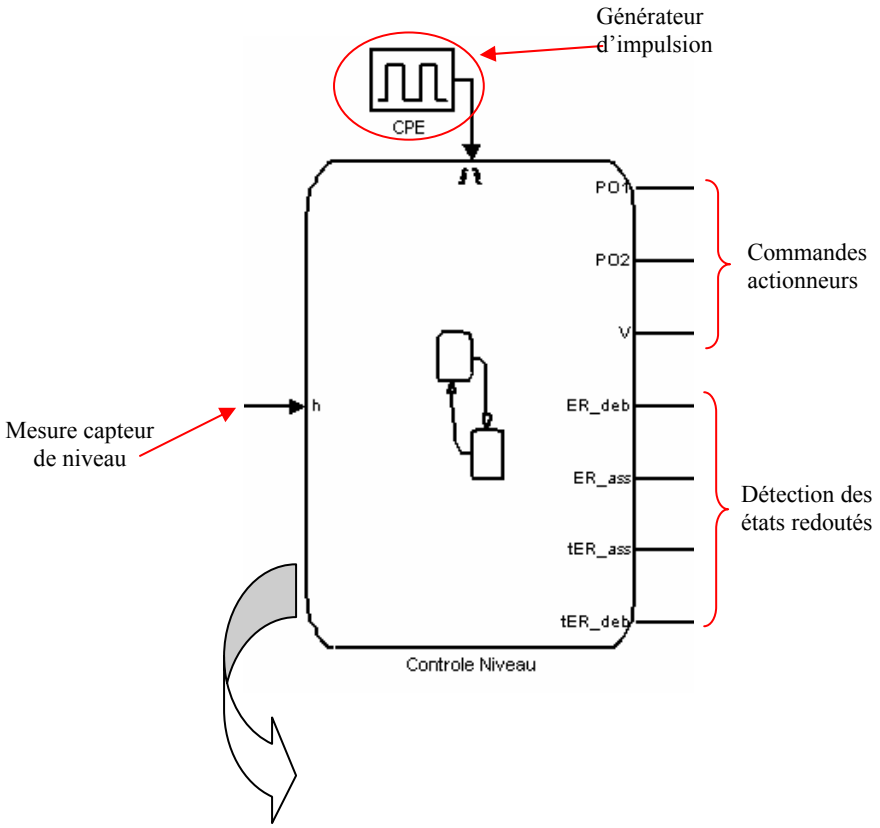
- **niveau_bas** : les deux pompes sont actives et la vanne est fermée. Cet état est atteint si le niveau est inférieur à h_0-Δh. Le système y reste tant que $h<h_0$.
- **niveau_haut** : les deux pompes sont désactivées et la vanne est ouverte. Cet état est atteint si le niveau dépasse $h_0+\Delta h$. Le système y reste tant que $h>h_0$.
- **niveau_normal** : la pompe principale est activée, la pompe de secours est arrêtée et la vanne est ouverte. Le système y reste tant que les seuils haut ou bas ne sont pas atteints.

A ces trois états, illustrés sur la figure 64, sont ajoutés deux états signifiant que le processus n'est plus maintenu dans le domaine de sécurité ou de bon fonctionnement. Ils dénotent un assèchement si le seuil critique h_0-2Δh est atteint, ou un débordement pour h> h_0+2Δh. Nous élargissons ainsi le rôle de l'automate de contrôle-commande à une fonction de « supervision ». L'intégration de ces états redoutés dans le modèle discret permettra leur détection et leur comptabilisation dans une future analyse quantitative par une simulation de Monte-Carlo. D'un point de vue strictement fonctionnel, ces états n'ont bien entendu aucune utilité dans la régulation.

Nous allons également étudier deux variantes de ce système à événements discrets, se différenciant par le traitement des informations en entrée et en sortie de l'automate.

1. **modèle triggé** : dans cette configuration, nous souhaitons que toutes les informations en provenance de la partie opérative du processus physique soient prises en compte sur des tops d'horloge. De même, les commandes élaborées par l'automate sont envoyées sur ces tops. Ce comportement synchrone par rapport à une horloge externe permet notamment de ne traiter que des variables discrètes par échantillonnage des grandeurs continues. On peut ainsi représenter explicitement la fréquence d'un processeur sur lequel sera implémenté l'algorithme de contrôle-commande. Nous utilisons un générateur d'impulsions en entrée du trigger. Les événements associées à chaque impulsion seront notés CPE (Changement de Période d'Echantillonnage, cf. [AUB87]).

2. **modèle non triggé** : cette variante n'inclut pas d'horloge externe. Les échanges d'informations ne sont pas imposés par des tops d'horloge. Le système évolue de façon asynchrone.



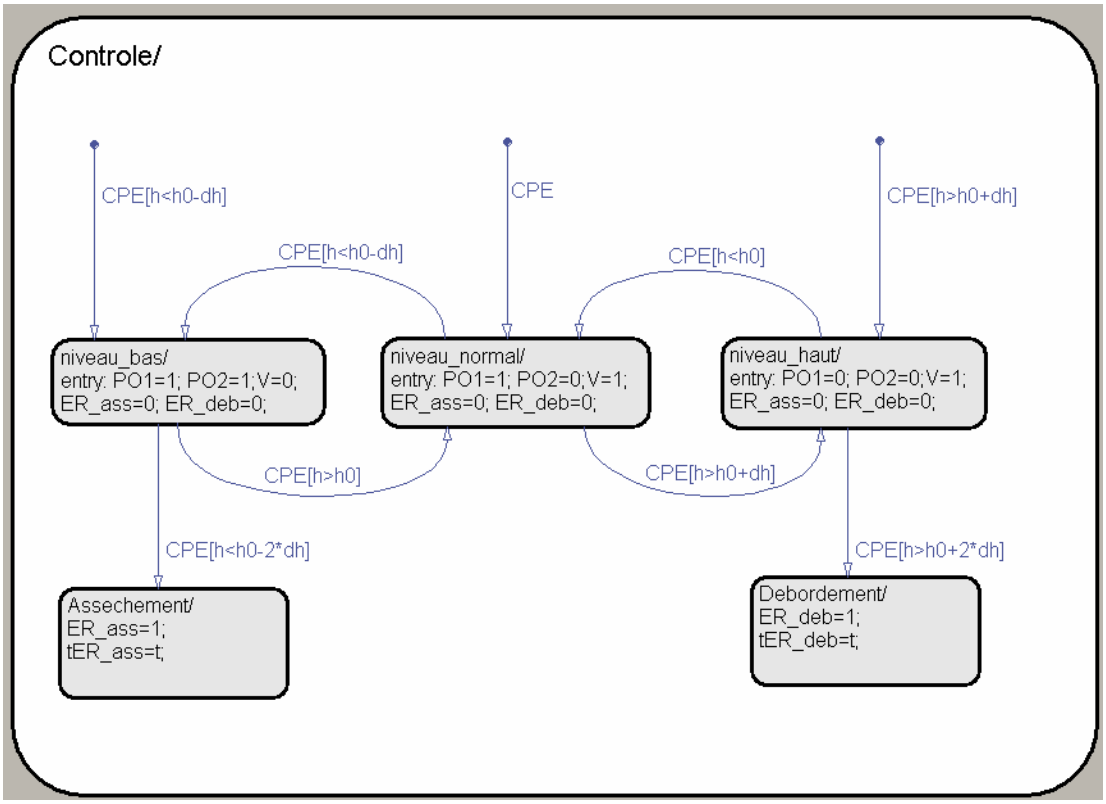


Figure 64 : Automate Stateflow de contrôle-commande et de supervision

Description des variables de l'automate de contrôle-commande :

Nom	Type	E/S automate	Description
CPE	Evènement	Entrée	Horloge externe
PO1	Variable booléenne	Sortie	Commande envoyée à la pompe PO1
PO2	Variable booléenne	Sortie	Commande envoyée à la pompe PO2
V	Variable booléenne	Sortie	Commande envoyée à la vanne V
ER_ass	Variable booléenne	Sortie	=1 si assèchement
ER_deb	Variable booléenne	Sortie	=1 si débordement
tER_ass	Variable entière	Sortie	Date d'occurrence de l'ER assèchement
tER_deb	Variable réelle	Sortie	Date d'occurrence de l'ER débordement
h	Variable réelle	Entrée	Niveau dans le réservoir

4.2. Modélisation du processus physique à commander

Cette partie détaille la modélisation du système physique composé par les actionneurs et le réservoir.

4.2.1. Modélisation du réservoir

Nous représentons à l'aide de blocs-fonctionnels Simulink l'équation d'évolution du niveau de liquide en fonction des débits des actionneurs. La figure 65 présente ce modèle :

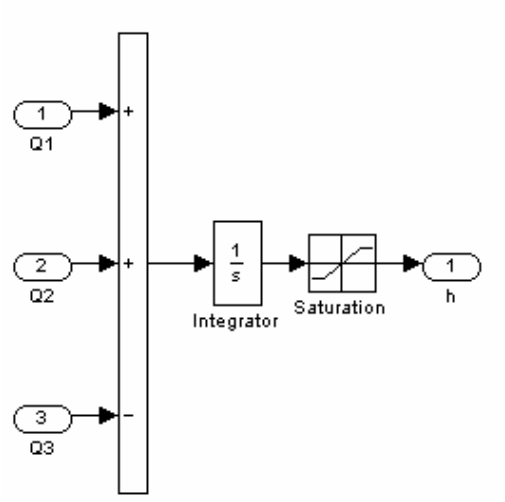


Figure 65 : Evolution du niveau de liquide dans le réservoir

La variable h est obtenue par intégration des débits fournis par les actionneurs. Le bloc « saturation » permet simplement de maintenir h entre une valeur minimale (réservoir vide) et maximale (réservoir plein).

4.2.2. Modélisation des actionneurs

Sur le modèle réseau de Petri proposé dans la partie précédente, les débits fournis par les actionneurs étaient supposés constants, par souci de simplification. Cependant, au regard de la puissance de modélisation « continue » offerte par les blocs-fonctionnels, nous souhaitons améliorer la description et le niveau de détails du comportement des actionneurs. Les débits seront de la forme suivante :

$$Q_i(t) = Q_{0i} + \delta Q_i(t)$$

où Q_{0i} représente la valeur moyenne du débit, spécifiée dans les hypothèses fonctionnelles et $\delta Q_i(t)$ la variation autour de la valeur moyenne. Ce terme correspond à une perturbation pouvant être d'origine mécanique, électromécanique ou électrique. Nous supposons néanmoins que ce terme est de moyenne nulle.

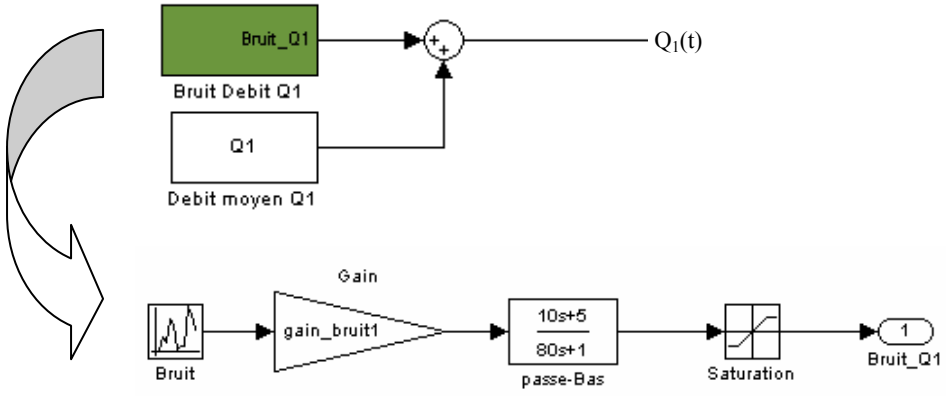
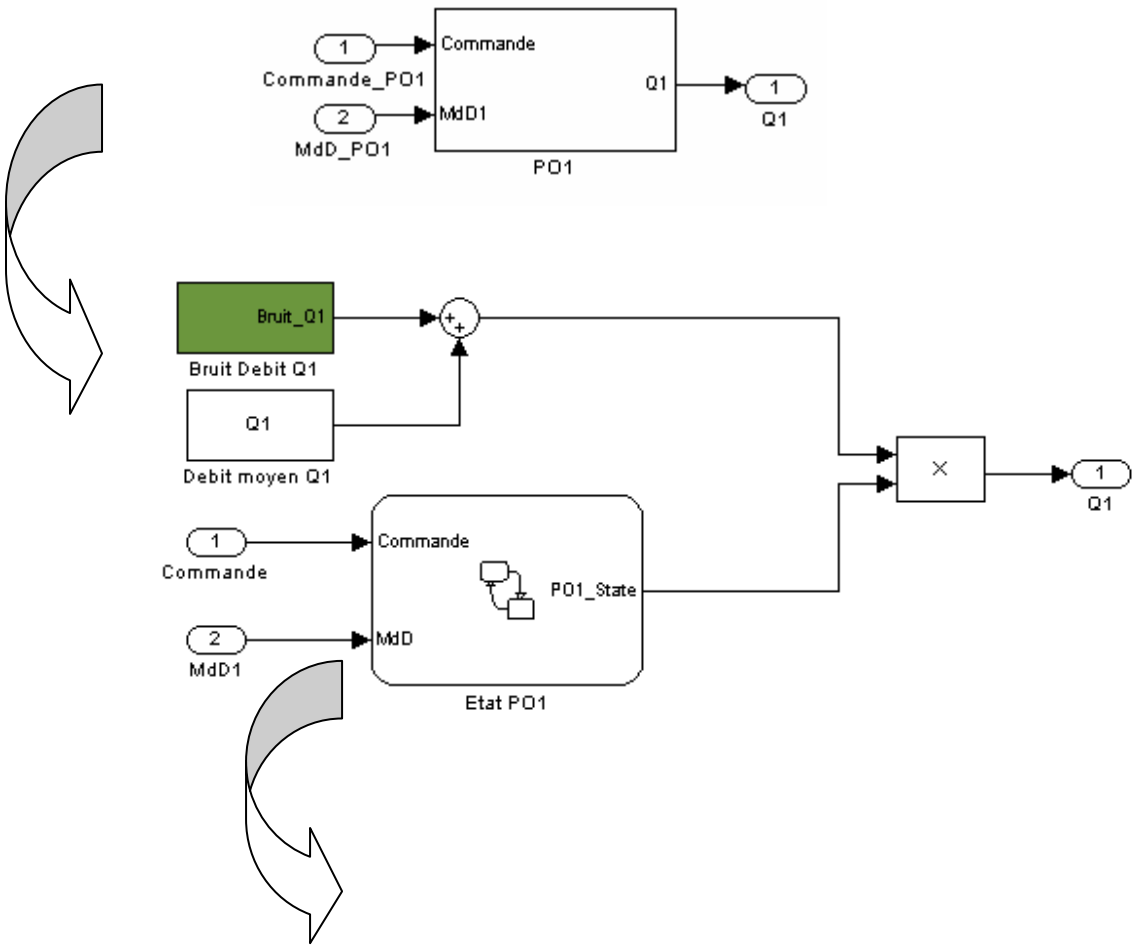


Figure 66 : Modélisation du débit Q_1 soumis à une perturbation

Cette complexification du comportement par l'ajout de phénomènes stochastiques dans l'évolution de la partie opérative est motivée par la mise en œuvre de la méthode d'analyse quantitative. En effet, nous allons voir par la suite que l'effort apporté pendant la construction d'un graphe de Markov agrégé et l'évaluation des coefficients de pondération ne sont pas proportionnels à la complexité des équations différentielles décrivant la partie continue. Le calcul des coefficients de pondération reposant sur la technique de la simulation, le niveau de détail peut être aussi fin qu'on le souhaite. Cette constatation n'est pas vérifiée pour les méthodes d'analyse basées sur une linéarisation des équations différentielles, comme celle présentée dans [MON98].

4.2.3. Modélisation des défaillances des actionneurs

Chacun des actionneurs est représenté par un bloc Simulink à deux entrées et une sortie. La première entrée est la commande reçue par la partie contrôle-commande ordonnant l'ouverture ou la fermeture. La deuxième entrée correspond à l'occurrence éventuelle d'une défaillance qui influera sur la commande par l'inhibition ou le forçage de celle-ci. La sortie représente le débit effectivement fourni par l'actionneur.
L'effet d'une défaillance de l'actionneur est modélisé par un automate à deux états : un état d'ouverture et un état de fermeture. Le passage de l'un à l'autre a lieu sur occurrence d'une commande de la partie commande, si aucune défaillance n'a lieu. Dans le cas contraire, en fonction du mode de défaillance, l'automate reste bloqué dans un des deux états. La figure 67 représente un modèle d'actionneur complet :



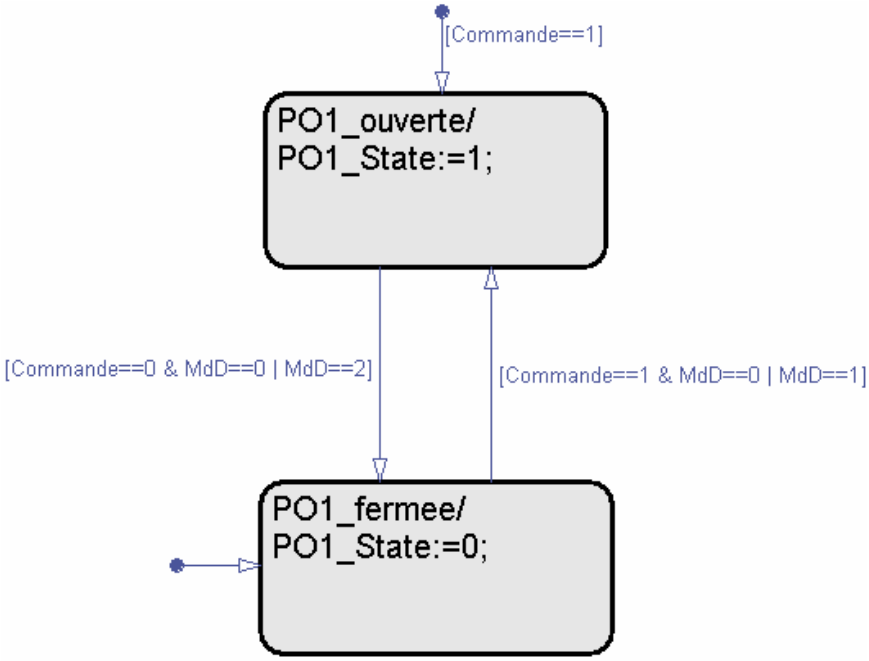


Figure 67 : Modèle complet d'un actionneur

L'état physique réel de l'actionneur est représenté par la variable **PO1_State** (pour la pompe PO1). La valeur 1 signifie que l'actionneur est dans l'état d'ouverture, et 0 dans l'état de fermeture.
L'occurrence d'une défaillance se traduit par une valeur non nulle de l'entrée **MdD** (Mode de Défaillance). Par hypothèse, trois modes de défaillance sont considérés : blocage en ouverture, en fermeture et dans la position courante. Les conventions suivantes seront choisies pour la variable MdD :

Valeur de MdD	Signification
0	Pas de défaillance, fonctionnement normal
1	Blocage en ouverture
2	Blocage en fermeture
3	Blocage dans la position courante

L'intégration des entrées MdD sur chaque actionneur permettra, d'une part, d'injecter les défaillances aléatoirement dans le modèle lors d'une simulation de Monte-Carlo et, d'autre part, de se positionner dans un mode opératoire spécifique qui permettra d'évaluer des probabilités conditionnelles asymptotiques relatives à chaque mode (simulation du comportement dans chaque macro-état identifié). Nous détaillerons ultérieurement ces différents points.

4.2.4. Automate d'injection de défaillances

Le positionnement des variables MdD_i à une valeur spécifiée dans le tableau est opéré par un automate de gestion des défaillances à partir de dates d'occurrence pré-calculées, fonctions des taux de défaillance.
La technique d'analyse fondée sur la simulation de Monte-Carlo consiste à jouer N histoires de durée T. Au début de chaque histoire, les instants de défaillance de chaque actionneur sont calculés et l'injection est réalisée via cet automate au cours de la simulation. L'automate est modélisé par un bloc Stateflow. Celui-ci est composé de trois « super-états » relatifs à chaque actionneur et fonctionnant en parallèle. Nous faisons l'hypothèse que lorsqu'un actionneur est soumis à un mode de défaillance particulier, celui-ci reste indéfiniment dans cet état. Il ne peut donc pas y avoir de basculement d'un mode de défaillance à un autre au cours d'une même histoire. La structure de l'automate reflète cette hypothèse. Par exemple, pour l'actionneur PO1, l'état initial est PO1_OK, signifie que celui-ci est opérationnel. La variable MdD est donc positionnée à 0. Trois dates sont calculées : t1_ouv, t1_fer et t1_blo. Elles correspondent à l'occurrence d'une défaillance dans l'un des trois modes de défaillance. Ces dates sont toutes différentes les unes des autres. Ainsi, l'une des trois conditions portées par les arcs [t>t1_ouv], [t>t1_fer] et [t>t1_blo] sera vérifiée en premier et la transition correspondante franchie. Le nouvel état atteint reflètera le mode de défaillance de l'actionneur et la variable MdD sera affectée de la valeur adéquate pour être injectée dans le bloc actionneur. Le calcul effectif des différentes dates et la mise en place de la simulation de Monte-Carlo seront précisés par la suite.

Date	Description
t1_ouv	date d'occurrence de la panne « blocage en ouverture de PO1 »
t1_fer	date d'occurrence de la panne « blocage en fermeture de PO1 »
t1_blo	date d'occurrence de la panne « blocage dans l'état courant de PO1 »
t2_ouv	date d'occurrence de la panne « blocage en ouverture de PO2 »
t2_fer	date d'occurrence de la panne « blocage en fermeture de PO2 »
t2_blo	date d'occurrence de la panne « blocage dans l'état courant de PO2 »
t3_blo	date d'occurrence de la panne « blocage dans l'état courant de V »

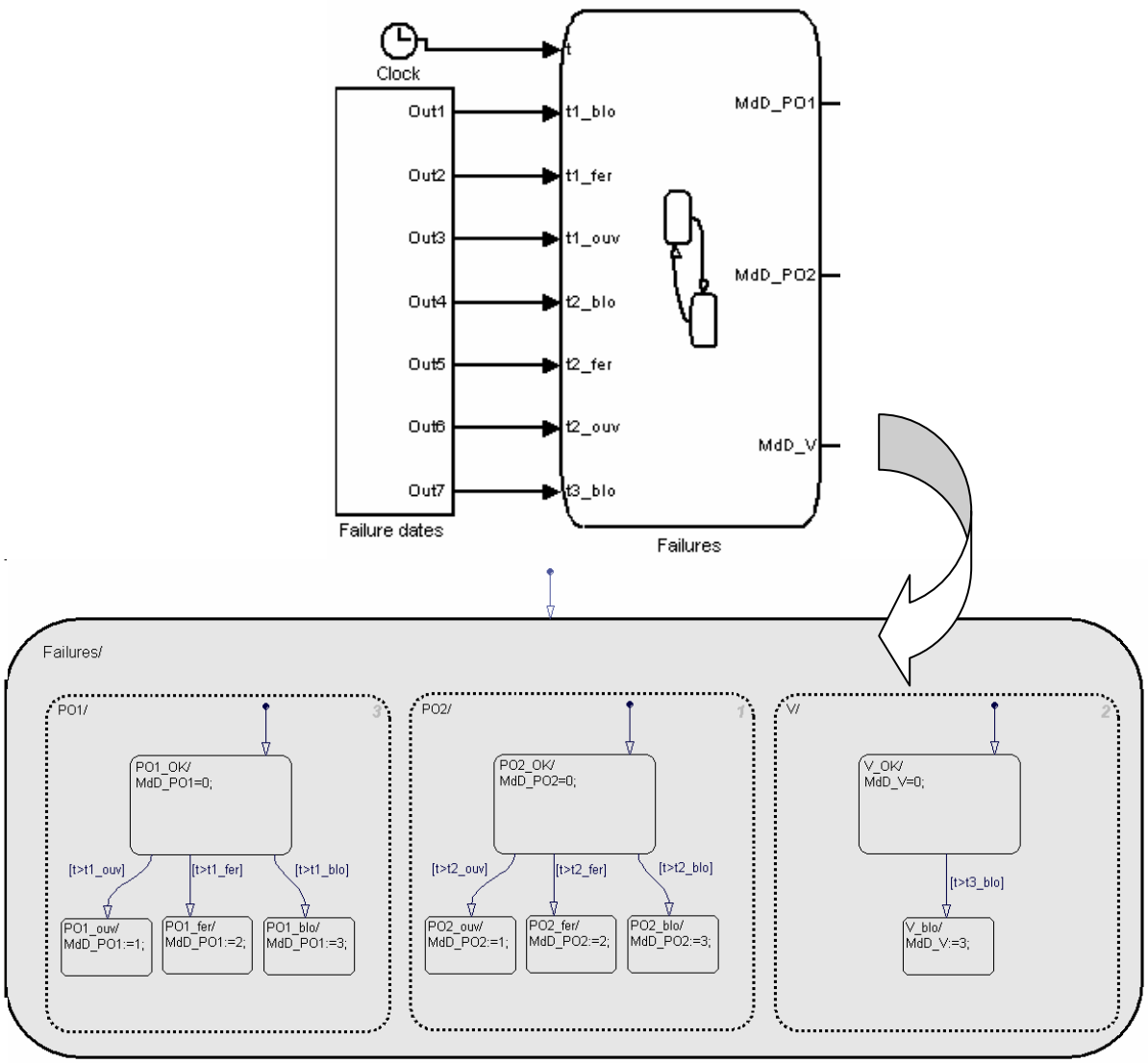


Figure 68 : Automate de gestion et d'injection des défaillances

4.2.5. Modèle complet

Le modèle complet, de nature hybride, comprenant la partie contrôle-commande, la partie opérative (actionneurs+réservoir) et l'automate de gestion des défaillances est présenté en figure 69 :

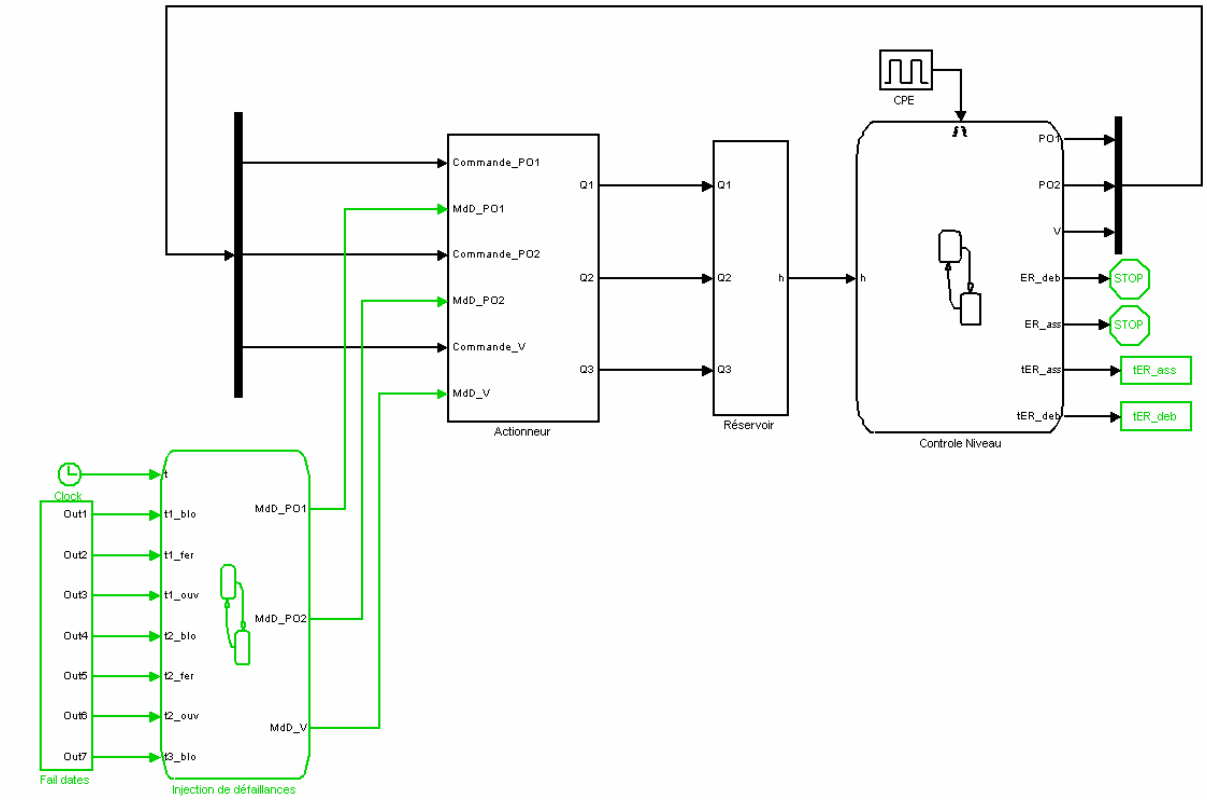


Figure 69 : Modèle complet

Nous allons à présent mener une étude quantitative des probabilités d'occurrence des événements redoutés assèchement et débordement par une simulation de Monte-Carlo supportée par les modèles « Réseau de Petri » et « Stateflow/Simulink ». Nous comparerons cette démarche classique avec la méthode proposée dans ce mémoire en construisant un modèle markovien agrégé du système complet.

5. Résolution par simulation de Monte-Carlo

C'est la méthode la plus simple à mettre en place car elle ne nécessite pas d'ajouter des contraintes ou de poser des hypothèses particulières sur le système. Le principe de la simulation de Monte-Carlo consiste à simuler N fois le modèle fonctionnel d'une durée T en y injectant aléatoirement des pannes. Pour chaque histoire jouée, le comportement du système est observé. La valeur de T doit être suffisamment grande pour voir apparaître les défaillances et observer un régime permanent après construction des courbes de probabilité. Le modèle fonctionnel (le système physique et sa commande) et le modèle dysfonctionnel sont simulés simultanément.

5.1. Simulation de Monte-Carlo à partir de la modélisation en réseau de Petri

Nous souhaitons connaître l'évolution des probabilités d'occurrence des événements redoutés sur l'intervalle de temps T=[0, 5000].
L'outil de réseau de Petri MOCA-RP V.10 permet d'évaluer statistiquement un certain nombre d'estimateurs par simulation ainsi que l'écart-type et l'intervalle de confiance à 90%. L'évaluation des probabilités s'identifie dans le cas présent par une estimation du marquage moyen des places associées aux événements assèchement et débordement. L'outil MOCA-RP fournit, pour une simulation de Monte-Carlo, le marquage moyen à la fin d'une histoire. La reconstitution des courbes de probabilités sur l'intervalle T nécessite d'effectuer plusieurs simulations de durées différentes, chacune correspondant à un point de la courbe. Chaque simulation est réalisée avec N=10 000 histoires. A titre indicatif, la machine utilisée est un PC Céleron à 2,2 GHz et à 512 Mo de RAM.

L'interface de simulation est présentée en annexe 4. Les résultats des simulations sont présentés dans le tableau suivant :

	Assèchement		**Débordement**		
Date t	Marquage moyen	Intervalle de confiance à 90%	Marquage moyen	Intervalle de confiance à 90%	Temps de calcul
200	$0{,}95.10^{-1}$	$4{,}81.10^{-3}$	$1{,}27.10^{-1}$	$5{,}55.10^{-3}$	1H 56mn
500	$2{,}37.10^{-1}$	$6{,}97.10^{-3}$	$2{,}86.10^{-1}$	$7{,}41.10^{-3}$	3H 5mn
700	$2{,}91.10^{-1}$	$7{,}45.10^{-3}$	$3{,}27.10^{-1}$	$7{,}70.10^{-3}$	3H 22mn
1000	$3{,}46.10^{-1}$	$7{,}80.10^{-3}$	$3{,}68.10^{-1}$	$7{,}91.10^{-3}$	3H 44mn
1500	$3{,}83.10^{-1}$	$7{,}97.10^{-3}$	$3{,}98.10^{-1}$	$8{,}03.10^{-3}$	3H 55mn
2000	$4{,}01.10^{-1}$	$8{,}04.10^{-3}$	$4{,}11.10^{-1}$	$8{,}07.10^{-3}$	3H 55mn
3000	$4{,}08.10^{-1}$	$8{,}06.10^{-3}$	$4{,}18.10^{-1}$	$8{,}09.10^{-3}$	4H 11mn
4000	$4{,}09.10^{-1}$	$8{,}06.10^{-3}$	$4{,}19.10^{-1}$	$8{,}09.\ 10^{-3}$	4H 11mn
5000	$4{,}11.10^{-1}$	$9{,}85.10^{-3}$	$4{,}18.10^{-1}$	$9{,}87.10^{-3}$	4H 21mn

On aboutit aux courbes de probabilités suivantes :

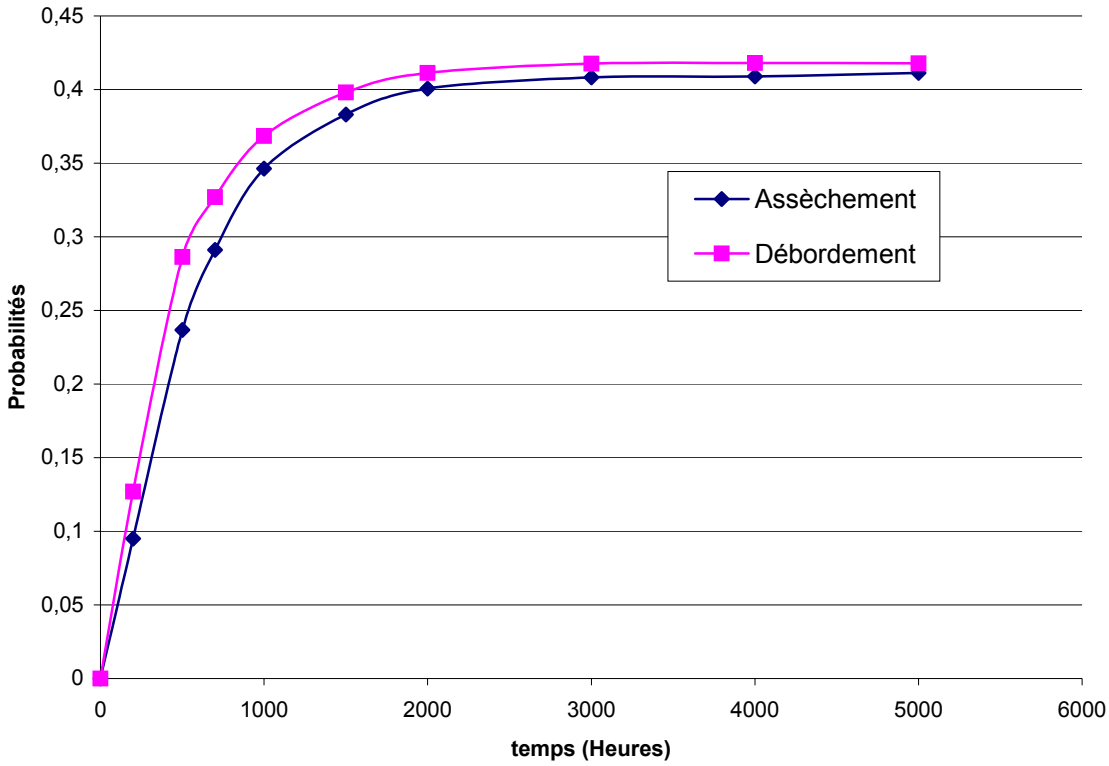


Figure 70 : Evolution des probabilités de présence dans les états redoutés « débordement » et « assèchement »

5.2. Simulation de Monte-Carlo à partir de la représentation « Simulink/Stateflow »

Contrairement à l'outil MOCA-RP intégrant une interface de gestion de la simulation et permettant de configurer tous les paramètres nécessaires (nombre d'histoires, durées, traitement des résultats…), l'outil Simulink/Stateflow nécessite de définir un algorithme de pilotage et de paramétrage de la simulation. Cet algorithme est écrit dans le langage Matlab à l'aide d'un script (fichier **.m**). Son rôle consiste précisément à tirer N histoires de durée T, c'est-à-dire de lancer N simulations du modèle en injectant les données pré-calculées (dates d'occurrence des défaillances). Les informations récoltées à l'issue de chaque simulation (dates d'arrivée éventuelles dans un état redouté) sont ensuite rassemblées dans un fichier de données **.mat**. Le script Matlab traite alors ce fichier afin d'en déduire les courbes de probabilité recherchées.
La gestion de la simulation de Monte-Carlo pour un modèle Simulink/Stateflow est résumée sur le schéma suivant :

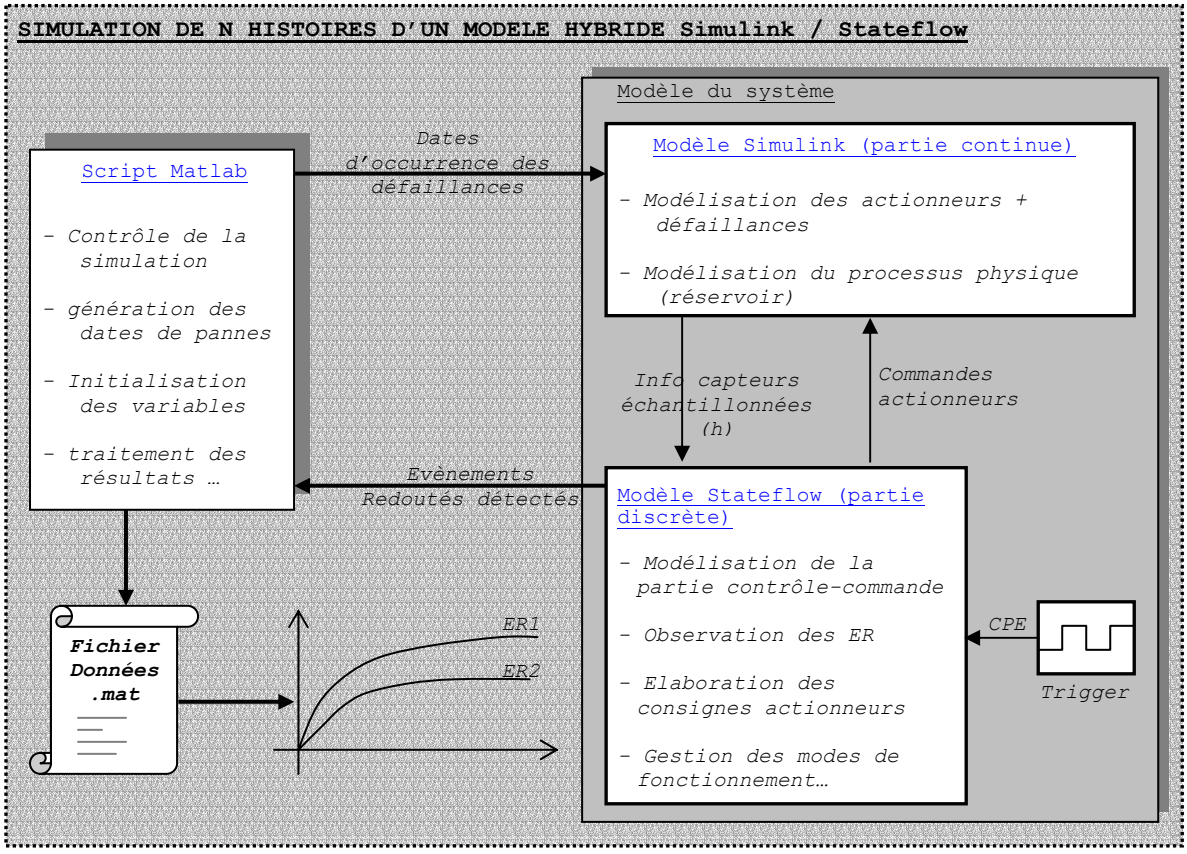


Figure 71 : Schéma de pilotage d'une simulation de Monte-Carlo

5.3. Elaboration des données d'entrée de la simulation et traitement des résultats

Le tirage au hasard des dates de défaillance est basé sur la génération de nombres aléatoires distribués uniformément. Le principe est simple : il suffit d'inverser la fonction de répartition F associée à la loi de probabilité relative à cet événement, comme illustré sur la figure suivante :

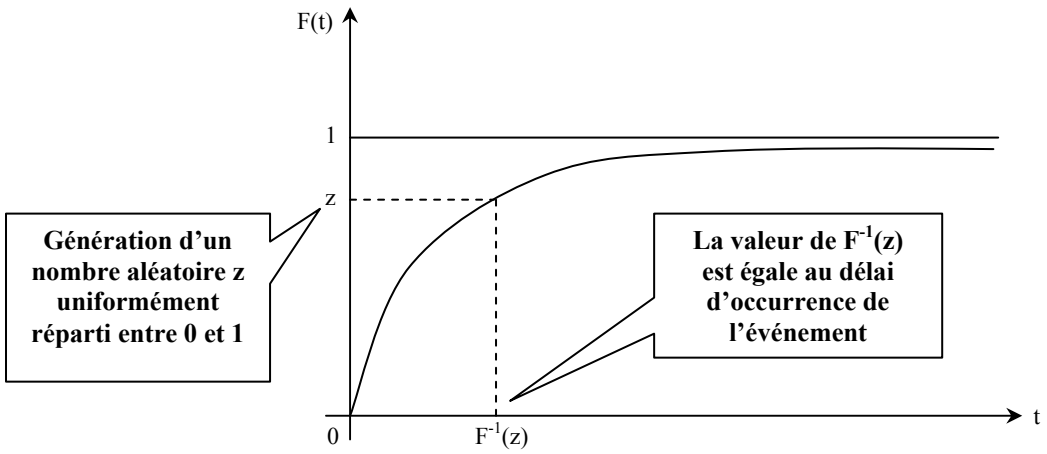


Figure 72 : Inversion de la fonction de répartition au point z

Soit t_d la date d'occurrence de la défaillance. On a donc $t_d=F^{-1}(z)$ avec $z\in[0, 1]$.
Dans le cas d'un composant dont le processus d'apparition des défaillances suit une distribution de loi exponentielle et de paramètre λ, la fonction de répartition s'écrit :

$$z = F(t) = 1-e^{-\lambda t}$$

Pour z uniformément répartie entre 0 et 1, l'instant t_d est donc :

$$t_d = -\frac{\ln(1-z)}{\lambda}$$

A chaque nouvelle histoire, une nouvelle série de tirages aléatoires des dates de défaillance est lancée et ces dernières sont injectées dans le modèle.
Inversement, à partir de la connaissance des différentes dates d'entrée dans les états redoutés pour l'ensemble des histoires jouées, il est possible de reconstruire la fonction de répartition de la loi associée au processus d'apparition de ces dits-événements. Ces fonctions de répartition fournissent directement les probabilités de présence dans ces états au cours du temps.

Posons :

- **N** : Nombre d'histoires jouées
- **T** : Durée d'une histoire
- **Pas_tps** : L'intervalle de temps [0, T] est discrétisé en **Nb_pas** de largeur **Pas_tps**. Sur chaque période $[t_i, t_{i+1}[$, la simulation de Monte-Carlo permettra d'estimer le nombre d'entrées dans l'état redouté considéré sur chacune de ces périodes.
- $\mathbf{N_{ER}}$ **:** Nombre total d'événements redoutés comptabilisés durant les N histoires.
- $\mathbf{N_{ER}(i)}$ **:** Nombre d'événements redoutés apparaissant sur la période $[t_i, t_{i+1}[$.
- **Y(i) :** probabilité d'être dans l'état redouté sur la période $[t_i, t_{i+1}[$. Cette fonction correspond à la fonction de répartition discrétisée associée à la loi de probabilité recherchée. Elle est déduite de **N** et de la variable estimée $\mathbf{N_{ER}(i)}$. Ainsi,

$$Y(i) = \frac{1}{N}\sum_{k=1}^{i} N_{ER}(k) \qquad \forall k\in[1, Nb\_pas]$$

Au terme de chaque histoire simulée, l'entrée dans un état redouté affecte une variable $\mathbf{t_{ER}}$, date d'entrée dans cet état. Cette affectation se fait via l'automate Stateflow de la partie contrôle-commande grâce à la commande **entry :** $\mathbf{t_{ER}=t}$; celle-ci est alors mémorisée. Elle reste à 0 si à la fin d'une histoire aucun événement ne s'est produit. Dans le cas contraire, les variables N_{ER} et $N_{ER}(i)$ sont incrémentées :

$N_{ER}=N_{ER}+1$
$N_{ER}(k)=N_{ER}(k)+1$ avec k tel que $t_k \le t_{ER} \le t_{k+1}$

A l'issue des N histoires, la fonction de répartition recherchée est estimée à partir de l'ensemble des observations :

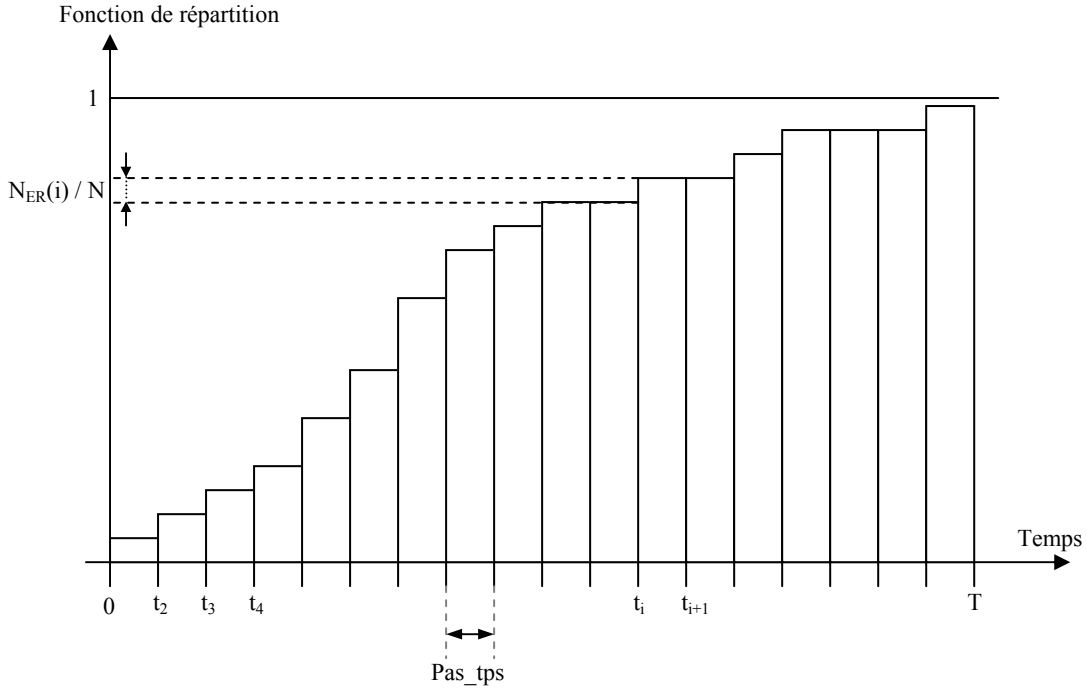


Figure 73 : Construction de la fonction de répartition à partir de la simulation des N histoires

Dans le cas du réservoir, on définit deux types de variables d'observation et deux fonctions de répartition : une associée à l'événement redouté assèchement et une autre au débordement.
La défiabilité du modèle se déduit alors très facilement, puisqu'il suffit de sommer les deux fonctions de répartition. A partir de la connaissance de la fiabilité globale (complémentaire de la défiabilité), il est également possible de calculer l'évolution dans le temps du taux de défaillance global du système à partir de la relation :

$$\lambda(t) = \frac{-\frac{dR}{dt}(t)}{R(t)}$$

Nous ne détaillerons pas davantage cet aspect. Le script Matlab de gestion de la simulation pour le système réservoir est présenté en annexe 4.

6. Résolution par approche markovienne

Nous allons à présent rechercher l'évolution des probabilités de présence dans les états redoutés débordement et assèchement en élaborant un graphe de Markov agrégé issu de la méthode proposée. Globalement, la démarche de construction est identique quel que soit le support de modélisation comportementale du système.
Il est néanmoins possible d'identifier les macro-états relatifs aux modes opératoires en construisant le graphe des marquages du système sous-jacent au modèle réseau de Petri. Même si le nombre de nœuds du réseau de Petri reste limité, le nombre de jetons introduits génère rapidement un nombre d'états élevé ; en effet, la place associée au réservoir peut contenir à elle seule entre 80 et 120 jetons. En combinant cette place aux autres places marquées (partie contrôle-commande et modélisations stochastiques), le nombre d'états possibles est de l'ordre de plusieurs centaines.
Nous choisissons de ce fait de construire directement le graphe réduit à partir d'un raisonnement logique issu des considérations structurelles et de la connaissance fonctionnelle et dysfonctionnelle du système.

La première étape consiste à identifier et recenser les macro-états. Toutes les configurations du système liées à un mode opératoire donné sont alors recherchées. A l'instar de l'exemple précédent sur le système de contrôle-commande élémentaire, ces modes opératoires sont définis par une (ou une combinaison de) défaillance(s) de telle sorte que le système continue à assurer sa fonction de régulation, mais éventuellement dans un mode dégradé. Ainsi, le passage d'un macro-état à un autre se fait uniquement sur franchissement d'une ou de plusieurs transitions stochastiques. Recensons les différentes situations possibles :

- **Mode nominal** : il regroupe tous les états élémentaires du système en l'absence de défaillances,
- **ER1** : il regroupe tous les états possibles du système en présence de l'événement redouté « débordement ». Une combinaison de défaillances des actionneurs amène le système dans ce macro-état absorbant. Par exemple, un blocage de la vanne en position de fermeture et les deux pompes bloquées en ouverture provoquent un débordement du réservoir.
- **ER2** : idem que précédemment, mais pour l'événement redouté assèchement.
- **ER3** : il s'agit également d'un macro-état absorbant, mais il ne présente pas de situation redoutée. Une combinaison de défaillance amène le système dans cette configuration qui se caractérise par un blocage du processus. Le niveau de liquide reste constant dans le réservoir, mais la régulation n'est plus assurée. Le blocage des trois actionneurs en position fermée en est un exemple.
- **Les modes dégradés** : nous emprunterons cette terminologie pour désigner tous les macro-états transitoires dont les états élémentaires n'appartiennent à aucune classe spécifiée auparavant. Ces macro-états sont caractérisés par la présence d'une ou de plusieurs pannes mais la fonction de régulation est toujours assurée par le système de contrôle-commande. Par exemple, la panne « Pompe PO1 bloquée en fermeture » n'a pas de conséquence dramatique pour le processus dont la régulation est maintenue grâce à la pompe de secours PO2. On dénombre 15 modes dégradés.

Ainsi, chaque macro-état est caractérisé par la présence au sein du système de zéro (mode nominal), d'une ou de plusieurs défaillances (modes dégradés et absorbants). Nous identifierons ces macro-états du graphe de Markov agrégé par la ou les pannes des

actionneurs. Dans le cas présent, il est relativement aisé d'associer les combinaisons à la nature des macro-états, compte-tenu des hypothèses fonctionnelles sur les valeurs des débits. Le graphe agrégé compte ainsi 1+3+15 macro-états, dont une partie est illustrée sur la figure suivante. Il est facile de vérifier par la logique combinatoire que l'ensemble de ces états forme un système complet. Par souci de lisibilité, le graphe est volontairement non exhaustif, tous les états et arcs n'étant pas représentés.

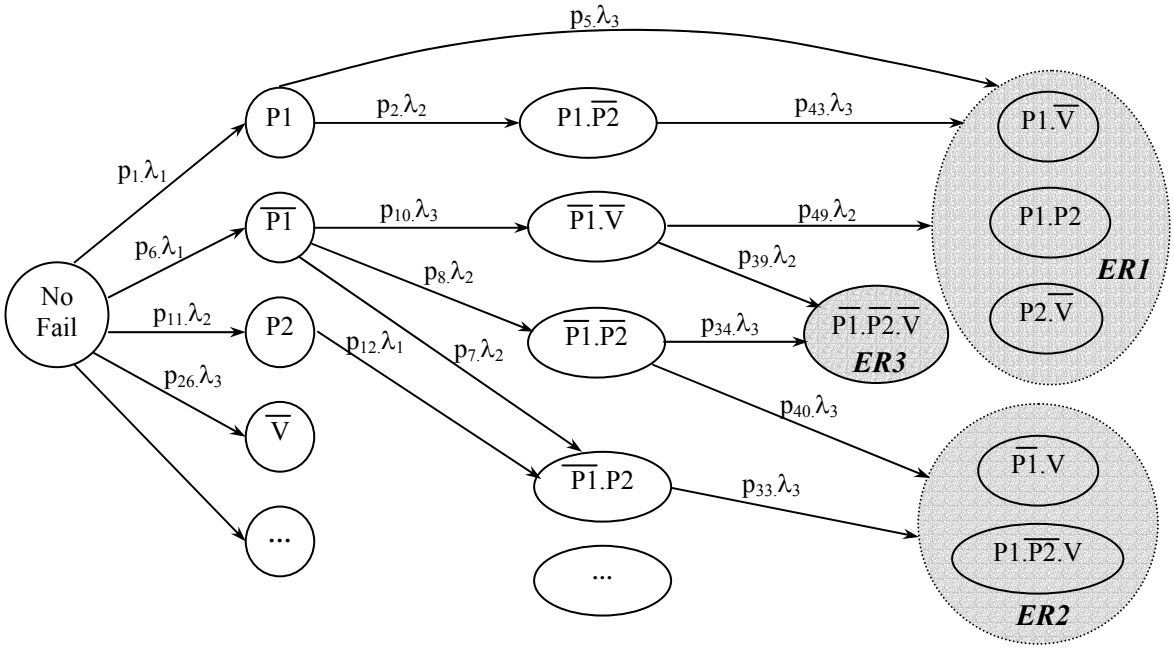


Figure 74 : Construction du graphe de Markov agrégé

Avec les notations suivantes :

P1 : Pompe 1 bloquée en ouverture	$\overline{P1}$: Pompe 1 bloquée en fermeture,
P2 : Pompe 2 bloquée en ouverture	$\overline{P2}$: Pompe 2 bloquée en fermeture,
V : Vanne bloquée en ouverture	$\overline{V}$: Vanne bloquée en fermeture.

La construction de ce graphe ne pose pas de problème. Celui-ci fait apparaître les combinaisons de défaillance qui conduisent à la réalisation des événements redoutés en passant par des modes dégradés transitoires. Encore une fois, les arbres de défaillance sont limités de par leur représentation statique de ces combinaisons et, de ce fait, fournissent des résultats pessimistes sur les indicateurs de sûreté de fonctionnement puisqu'ils ne tiennent pas compte de la dynamique interne du processus. En effet, si l'on considère une combinaison de deux défaillances, l'occurrence de la première (caractérisé par un certain taux) amène le système dans une configuration donnée. Ceci a pour conséquence de modifier le taux de la deuxième puisque l'actionneur concerné verra son fonctionnement modifié. En effet, d'après les hypothèses dysfonctionnelles, le mode de défaillance « blocage en position courante » dépend du cycle « ouverture/fermeture ». La proportion de temps passé dans chacune des positions aura donc une influence directe sur la probabilité du type de blocage. Alors que ces variations de taux de défaillance n'apparaissent pas sur les arbres de défaillance, le graphe de Markov comble cette lacune par l'intermédiaire des coefficients de pondération p_i (ou probabilités conditionnelles asymptotiques).

Seul le mode dégradé « blocage en position courante » fait intervenir explicitement la dynamique du système. Nous nous focalisons donc sur les macro-états transitoires en recherchant les composantes fortement connexes dont les probabilités conditionnelles de présence dans certains des états influent sur les taux de transitions. Dans le cas présent, ces composantes sont relatives au fonctionnement des actionneurs et à l'automate de commande. Par exemple, considérons les séquences de défaillance suivantes provoquant un passage à l'état redouté ER1:

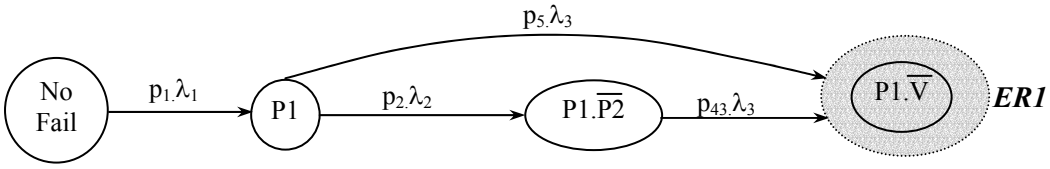


Figure 75 : Exemple de séquences élémentaires de défaillance

Supposons que le système est en présence d'une défaillance de la pompe principale, dont le mode de défaillance est blocage en position ouverte, c'est-à-dire P1. Une seconde défaillance de la vanne V dans la position fermée conduit directement à l'état redouté ER1. On recherche alors la composante conservative associée au fonctionnement de la vanne en faisant abstraction des transitions stochastiques. Celle-ci est illustrée sur la figure suivante :

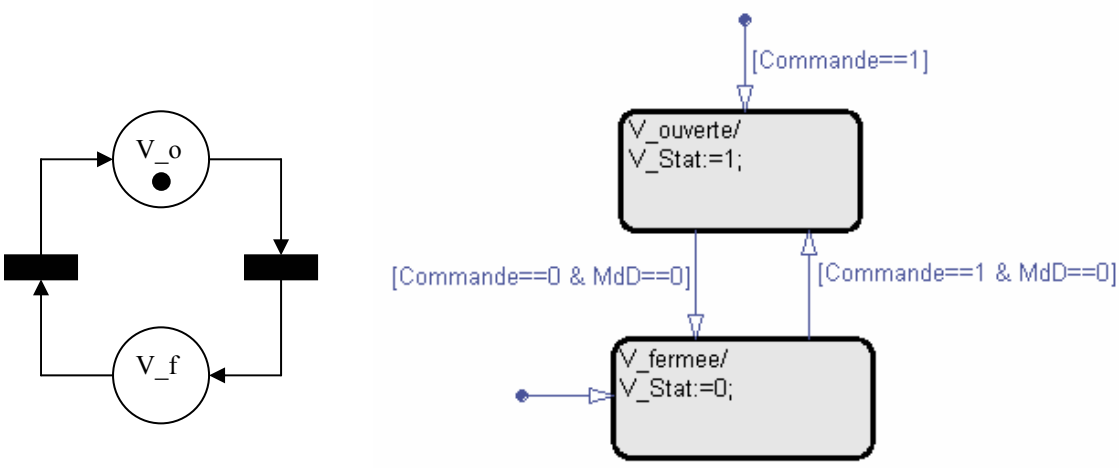


Figure 76 : Composante conservative associée au fonctionnement de la vanne, versions réseau de Petri et Stateflow

La composante conservative associée aux positions de la vanne s'exprime facilement : M(V_o)+M(V_f)=1, traduisant le fait que la vanne est toujours dans l'une des deux positions.
Les structures du réseau de Petri et de l'automate Stateflow sont similaires. Le raisonnement permettant d'obtenir les coefficients de pondération est donc le même quel que soit le formalisme (à état/transition) choisi.
Le coefficient de pondération p_5 s'identifie au marquage moyen de la place V_f du réseau de Petri, sachant que la pompe est bloquée en ouverture. Pour la version Stateflow, ce coefficient est égal au temps de séjour moyen passé dans l'état V_fermee sur la somme des temps de séjour moyen dans les états V_ouverte+V_fermee.

Le principe d'évaluation des autres coefficients est identique. La transition de l'état P1 vers l'état P1.$\overline{P2}$ est franchie sur occurrence d'une panne de la pompe de secours PO2. Cependant,

contrairement au mode de défaillance de la vanne, un blocage de la pompe PO2 en position fermée est possible selon deux modes de défaillance : blocage dans la position courante, sous réserve que celle-ci est fermée au moment du blocage, et blocage en fermeture. Deux transitions stochastiques du réseau de Petri modélisent ces possibilités. Le terme de pondération p_2 est donc la somme de deux coefficients, relatifs à chacun des modes. Le mode « blocage en position fermée » ne dépendant pas de l'état de l'actionneur, ce coefficient vaut 1. De façon analogue, le blocage dans la position courante dépend du temps de séjour passé dans l'état PO2_fermee. Nous recherchons donc la probabilité conditionnelle que la pompe PO2 soit fermée, sachant que le système est dans le macro-état transitoire P1. Or, dans cette configuration, d'après les hypothèses fonctionnelles, la pompe PO2 n'est jamais sollicitée en ouverture. La probabilité conditionnelle vaut également 1. Le terme p_2 vaut donc 2. Il est également possible d'interpréter $p_2\lambda_2$ comme étant la somme de deux termes :

$$p_2\lambda_2 = p_2^1\lambda_2 + p_2^2\lambda_2$$

en considérant la formule exprimant le taux de défaillance équivalent issue du processus d'agrégation, donnée par l'équation [19]. C'est une combinaison linéaire de l'ensemble des taux de défaillance pondérés par les probabilités asymptotiques appropriées. Les deux modes de défaillance « blocage en position courante » et « blocage en fermeture » de l'actionneur PO2 étant considérés comme deux défaillances distinctes, de même taux λ_2.
Un tableau analogue à celui de la figure suivante peut aider un concepteur/analyste à répertorier exhaustivement l'ensemble des taux équivalents portés par les arcs du graphe de Markov réduit, en identifiant les composantes fortement connexes et les états fonctionnels intervenant dans les coefficients de pondération. Ainsi, la construction d'un modèle dysfonctionnel est directement issue d'un raisonnement logique et de la bonne connaissance du système {processus, partie commande}.

Composant défaillant/ Mode de défaillance	**Taux de défaillance**	**Macro-état amont / mode de fonctionnement**	**Macro-état aval**	**Composante fortement connexe impliquée**	**Etat élémentaire fonctionnel de la composante**	**Terme de pondération**	
Blocage de PO1 en ouverture	λ_1	Mode nominal (No Fail)	Mode dégradé P1	Fonctionnement de la pompe PO1	Tous les états (PO1 ouverte et fermée)	p_1^1 (=1)	p_1
Blocage de PO1 en position courante d'ouverture	λ_1	Mode nominal (No Fail)	Mode dégradé P1	Fonctionnement de la pompe PO1	Pompe PO1 ouverte	p_1^2 (= ?)	
Blocage de la vanne en position courante de fermeture	λ_3	Mode dégradé P1	Etat redouté de débordement ER1	Fonctionnement de la vanne V	Vanne V fermée	p_5 (= ?)	
…	…	…	…	…	…	…	

Figure 77 : Aide à l'identification des taux de transition équivalents

Dans le cas présent, les composantes fortement connexes sont relatives aux différents états des actionneurs, modélisés par des automates. Dans l'exemple du système de contrôle-commande à reconfiguration logicielle et matérielle traitée dans la partie 5 du chapitre 3, les composantes connexes découlent directement de l'automate représentant la partie contrôle en réseau de Petri. Un tableau d'aide à l'identification des transitions équivalentes est construit en annexe 3.

Evaluation des coefficients de pondération :

A partir de la connaissance des composantes connexes identifiées dans le tableau précédent, l'évaluation des coefficients ne pose pas de problème majeur. Seules les défaillances des actionneurs selon le mode « blocage dans la position courante » impliquent une évaluation, les autres étant égaux à 1.

La simulation du modèle comportemental constitue alors un moyen efficace d'évaluation de ces termes. La démarche est la suivante :

1. Le système {processus+partie contrôle-commande} est positionné dans le mode de fonctionnement souhaité (macro-état amont) en forçant les variables continues et/ou discrètes à des valeurs adéquates ; par exemple, pour le macro-état P1, une défaillance de la pompe PO1 en blocage dans la position ouverte est injectée « manuellement ».
2. Toute la partie dysfonctionnelle est inhibée, c'est-à-dire que toute défaillance ultérieure est interdite.
3. L'ensemble du système est alors simulé dans **ce seul mode de fonctionnement**. Les temps de séjour moyens ou les marquages moyens (en fonction du formalisme choisi) sont alors mesurés par la simulation. L'existence de ces valeurs moyennes est assurée par la propriété d'ergodicité des composantes fortement connexes.
4. Les coefficients de pondération sont ensuite déduits des mesures précédentes. Ils s'identifient alors au marquage moyen ou au temps de séjour moyen passé dans l'état examiné sur la somme des temps de séjour dans les états de la composante.

Cette opération est répétée pour chacun des coefficients dont l'estimation n'est pas triviale.

Remarque : contrairement à une simulation classique de Monte-Carlo permettant l'estimation des grandeurs de sûreté de fonctionnement, l'emploi de la simulation dans le cas présent n'est pas contraignant. En effet, l'approche globale est basée sur un découplage des dynamiques. Seul le modèle comportemental en l'absence de défaillance est étudié. Il n'y a donc pas de simulation simultanée du modèle fonctionnel et du modèle dysfonctionnel impliquant des ressources de calcul très importantes de par la présence de deux échelles de temps très différentes (problème de simulation des événements rares). De plus, les probabilités conditionnelles (ou le processus de marquage) convergent très rapidement vers la valeur asymptotique recherchée.
Un avantage de la simulation est certainement la possibilité de représenter un fonctionnement complexe du processus continu, laissant ainsi une grande liberté de modélisation au concepteur. Contrairement à une approche markovienne classique, impliquant un conditionnement du modèle par la formulation d'hypothèses très fortes et contraignantes, l'approche proposée ne conserve l'hypothèse markovienne que dans le processus de défaillance. Tous les phénomènes déterministes relatifs au fonctionnement du système sont néanmoins intégrés dans le modèle markovien par l'intermédiaire des coefficients de pondération. Ceux-ci résument en quelque sorte l'influence réciproque de la partie

fonctionnelle (dynamique interne) sur la partie dysfonctionnelle (dynamique des défaillances). La théorie des perturbations singulières permet alors de condenser toutes ces informations en variables asymptotiques ou moyennes.

Les différents coefficients de pondération du graphe réduit sont répertoriés en annexe 4. Ils sont déduits de la simulation ou d'un raisonnement logique selon le cas. La modélisation Simulink/Stateflow sert de support au calcul des différentes inconnues.
Soulignons néanmoins que cette démarche implique d'avoir une bonne connaissance du système global, tant au niveau fonctionnel que dysfonctionnel.

Une fois l'ensemble de ces coefficients connu, la matrice de transition associée au graphe de Markov réduit s'obtient facilement. Celle-ci est également présentée en annexe 4. Une résolution analytique de l'équation de Chapman-Kolmogorov fournit alors l'évolution des probabilités :

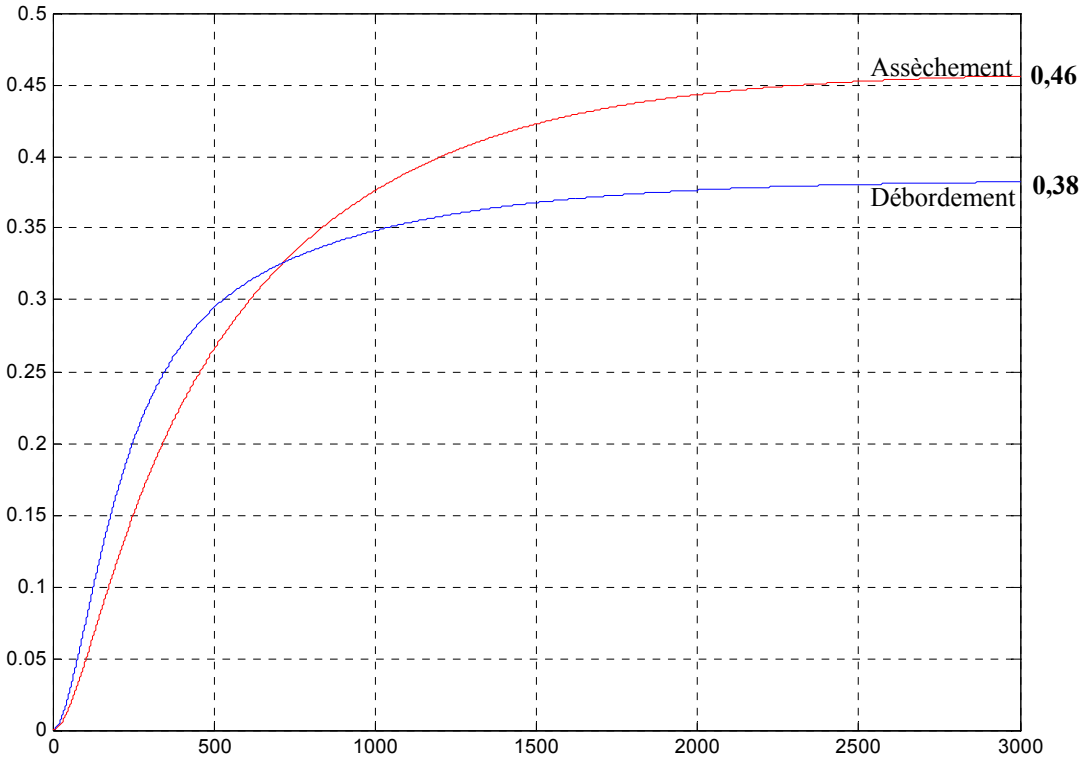


Figure 78 : Evolution des probabilités de présence dans les deux états redoutés

A l'instar du schéma de mise en oeuvre de la simulation de Monte-Carlo pour un modèle Simulink/Stateflow, nous illustrons les différentes étapes de construction d'un graphe de Makov agrégé pour un modèle quelconque sur la figure suivante. Celle-ci fait apparaître le découplage entre l'analyse du processus de défaillance (élaboration de la structure du graphe agrégée) et l'analyse de la dynamique interne (évaluation des probabilités conditionnelles asymptotiques).

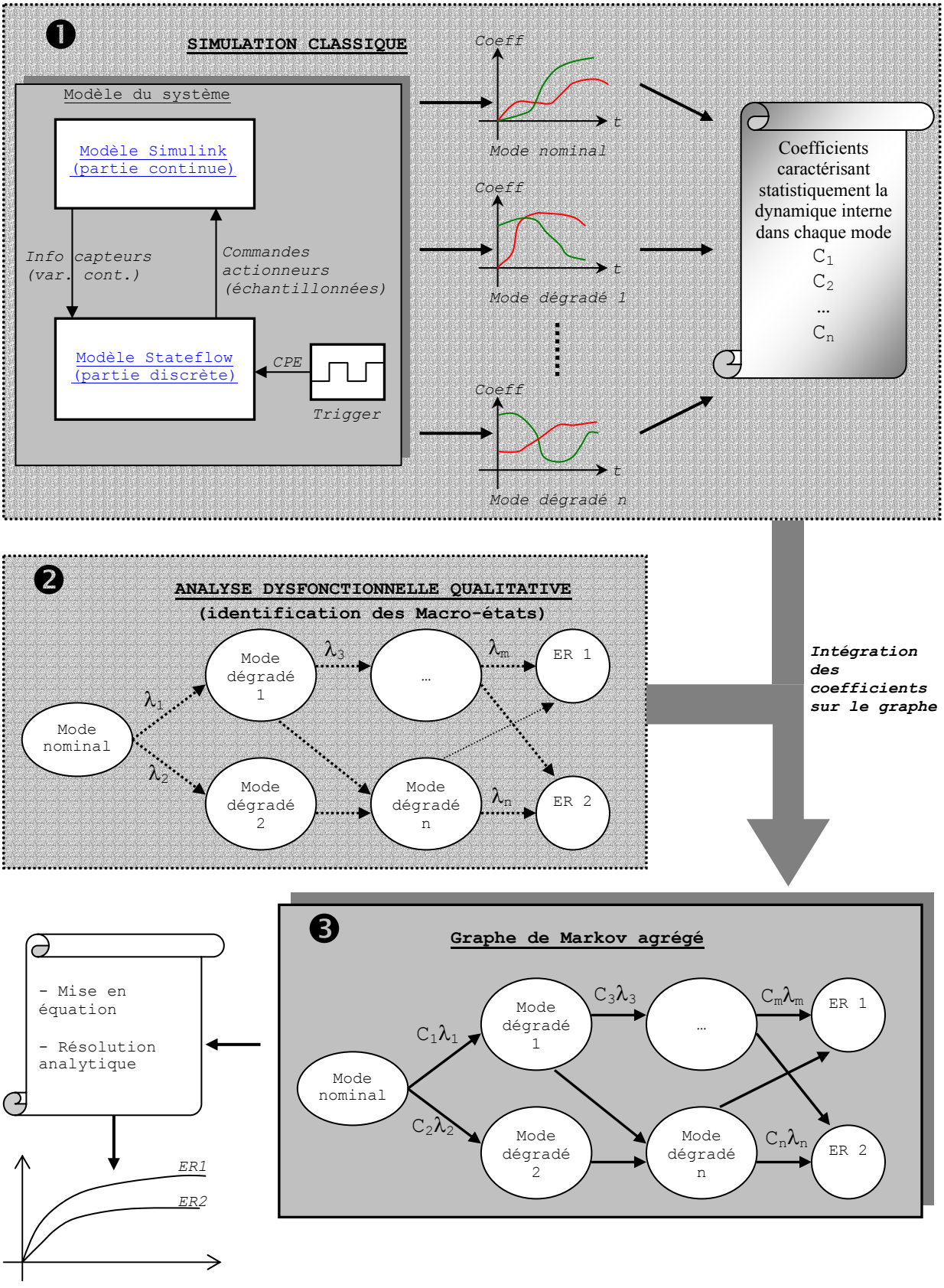


Figure 79 : Démarche globale pour la construction d'un graphe de Markov agrégé

7. Comparaison de l'approche analytique avec la simulation de Monte-Carlo

Nous allons à présent mettre en œuvre une série de simulations de Monte-Carlo du modèle Simulink/Stateflow afin de mettre en lumière les principales difficultés propres à cette technique. Outre le temps de calcul important nécessaire à l'apparition des événements rares, un certain nombre de contraintes doit être contourné afin de recueillir des résultats pertinents. Les choix du solveur et éventuellement du pas de calcul, de la fenêtre temporelle simulée et du nombre d'histoires jouées doivent être judicieusement examinés. Nous allons également faire varier l'ordre de grandeur des taux de défaillance afin d'analyser les conséquences en termes de coûts de calcul.
Plusieurs points seront traités dans cette partie :

- estimer le temps de calcul de chaque configuration de simulation,
- calculer les erreurs relative et absolue pour les courbes de probabilités des événements redoutés entre la méthode markovienne et la simulation.

7.1. Configuration matérielle et logicielle

La machine utilisée pour la simulation est un Pentium IV tournant à 2,6GHz avec 512Mo de RAM.
La version de Matlab est la V6.5 R13, Simulink V5.0.2, Stateflow V5.1.

7.2. Cas étudiés

Les différentes configurations de simulations sont rassemblées dans le tableau suivant. Deux types de modèles différents sont étudiés, se différenciant par la présence d'un trigger sur l'automate de contrôle-commande. Ce trigger (signal de type créneau), de période une seconde, génère des impulsions toutes les 0,5s.

Simul. N°	Modèle		T	N_{max}	Dynam. des défaill. (ordre de grandeur)	Rapport des dynam. (ordre de grandeur)	Solveur SK	
	Non triggé	Triggé					Pas	Type
1	✓		3000	10000	10^{-3}	10^{-5}	Fixe (0,1)	Ode 1
2	✓		3000	10000	10^{-3}	?	Variable	Ode 45
3	✓		3000	10000	10^{-3}	10^{-6}	Fixe (0,01)	Ode 1
4	✓		15000	10000	10^{-4}	?	Variable	Ode 45
5	✓		15000	10000	10^{-4}	10^{-6}	Fixe (0,1)	Ode 1
6		✓	3000	10000	10^{-3}	10^{-4}	Fixe (0.1)	Ode 1
7		✓	3000	10000	10^{-1}	1	Fixe (0.1)	Ode 1
8		✓	150000	17000	10^{-5}	10^{-6}	Variable	Ode 45

Figure 80 : Les différentes configurations étudiées

Dans ce tableau est également précisé l'ordre de grandeur relatif au rapport entre la dynamique interne du système et la dynamique des pannes, lorsque cela est possible. Nous

nous basons sur la période du générateur d'impulsions pour caractériser la dynamique du système .
Le choix de la période simulée varie également puisque celle-ci est tributaire de la dynamique des défaillances. En effet, pour chaque cas étudié nous souhaitons visualiser l'évolution des probabilités jusqu'à l'obtention de la limite asymptotique.

7.3. Examen des courbes de probabilités

Les courbes de probabilité des événements redoutés « assèchement » et « débordement » sont recherchées par chacune des deux méthodes d'évaluation dynamique ; soulignons que la technique markovienne proposée ne nécessite que très peu d'adaptation entre les différentes configurations ; en effet, une fois l'ensemble des coefficients de pondération obtenus, la matrice de transition s'obtient immédiatement puisque sa structure est identique pour les 8 configurations. Seuls sont modifiés les taux de défaillance par un rapport multiplicatif. Ainsi, l'essentiel du travail réside dans la construction de la première matrice.

Pour chaque cas de figure, nous nous intéresserons au calcul de l'erreur relative et de l'erreur absolue moyenne (en temps) entre la solution expérimentale et la solution analytique en fonction du nombre d'histoires jouées $N \in [1000; 10000]$. L'ensemble des résultats est synthétisé en annexe 4.

Quel que soit le modèle (triggé ou non), le solveur donnant les meilleurs résultats en termes de minimisation d'erreur est de type ODE 1 à pas fixe, au détriment du temps de calcul.
Concernant le modèle non triggé, on constate aisément que seul ce solveur à pas fixe convient. En comparant les cas N°1 et N°3, les erreurs sont globalement du même ordre de grandeur, néanmoins légèrement inférieures pour le N°3 simulé avec un pas de calcul plus fin. Les erreurs relatives moyennes sont de l'ordre de 1%. La qualité des courbes obtenues (pour N=10000 histoires) peut s'interpréter par le fait que le comportement du modèle est reproduit fidèlement par la simulation, tant sur le plan fonctionnel (dynamique rapide) que dysfonctionnel (dynamique lente). Cette remarque doit cependant être relativisée : en effectuant une simulation individuelle (N=1), le comportement du système peut différer en fonction de la valeur du pas (0,1 ou 0,01). En effet, l'ajout de phénomènes très rapides tels que les perturbations ajoutées aux débits des actionneurs peut modifier légèrement les instants de basculement d'une position à l'autre. Un pas trop grossier peut engendrer des erreurs de comportement par la non prise ne compte d'événements discrets entre deux pas (par exemple un double basculement d'un actionneur entre deux pas). Ce phénomène a été constaté pour la simulation N°1. Mais pour un nombre d'histoire N beaucoup plus élevé, le comportement global « moyen » est statistiquement identique pour les cas N°1 et 3 (la perte d'information reste marginale pour ces valeurs de pas).
On en conclut qu'un solveur à pas fixe de valeur 0,1 reproduit assez fidèlement les comportements rapide et lent pour le modèle non triggé. Cette conclusion est sensiblement identique pour le cas de figure N°5.
Il n'en est pas de même pour le solveur à pas variable ODE 45 (basé sur l'algorithme de Runge Kutta). Les courbes de probabilité ne sont pas du tout conformes à celles attendues. Précisons tout d'abord que toutes les options de l'interface permettant le paramétrage de ce solveur ont été laissées par défaut (pas maximal, minimal, …). Une simulation individuelle du modèle avec un solveur ODE 1 de pas égal = 0,01 et un solveur ODE 45 donne un comportement différent des variables pour chacun des solveurs :

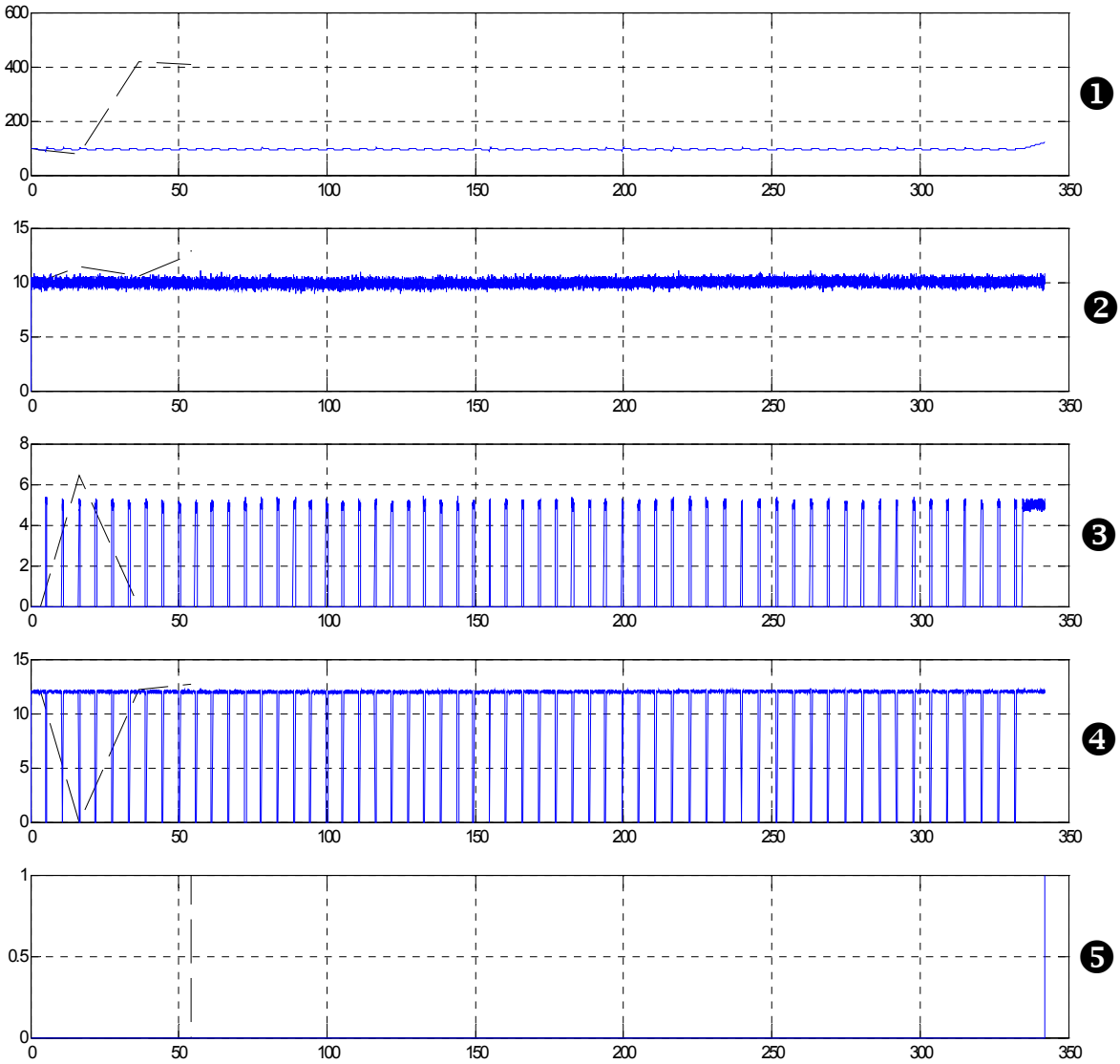


Figure 81 : Simulation du comportement du modèle avec ODE 1 et ODE 45

(1) niveau de liquide h
(2) débit de la pompe PO1
(3) débit de la pompe PO2
(4) débit de la vanne V
(5) date d'occurrence de l'événement redouté « débordement »

——— Solveur ODE 1 à pas fixe (0,01)
------- Solveur ODE 45 à pas variable

Il apparaît nettement que le nombre de pas générés par ODE 45 est insuffisant. Il n'est que de 12, alors que le solveur à pas fixe en a généré plus de 34000. La détection de la date de débordement est alors erronée, d'où la conséquence sur les courbes de probabilité recherchées. Il semblerait que le solveur à pas variable ne permette pas de reproduire et visualiser la dynamique rapide du système avec les options par défaut. Il est donc nécessaire d'affiner le paramétrage de cet algorithme.

Le système triggé est a priori plus conforme à la réalité puisque l'horloge modélisée permet de représenter l'échantillonnage des entrées du système de contrôle-commande (qui sera implanté sur un calculateur), ainsi que les consignes envoyées par celui-ci aux actionneurs à des instants discrets (définis par les tops d'horloge). Pour ce système, la dynamique des perturbations rapides est supposée n'avoir qu'un effet négligeable sur le comportement dysfonctionnel du système, il suffit de choisir le pas du solveur fixe égal à la durée entre deux tops d'horloge. Dans le cas présent, cette durée est égale à 0,5. Nous avons préféré conserver une granularité de la simulation assez fine en choisissant un pas de solveur égal à 0,1. L'erreur relative moyenne est du même ordre de grandeur que dans le cas non triggé (entre 2 et 4 %).
L'objectif de la configuration 7 était de diminuer sensiblement la différence entre la dynamique lente et rapide (non réaliste physiquement !) en augmentant les taux de défaillance. La conséquence immédiate de ce rapprochement de dynamiques est que l'erreur relative moyenne entre les deux méthodes dépasse les 10 %. Cette constatation n'est pas surprenante puisque la validité de la méthode analytique repose principalement sur un critère de séparation des dynamiques qui doit être suffisamment importante pour que la théorie des perturbations singulières puisse s'appliquer. Pratiquement, l'hypothèse de séparation est normalement vérifiée.
Enfin le dernier cas de figure (simulation N°8) est sans doute la plus pertinente puisque le but était de diminuer les taux de défaillance afin de se rapprocher d'une situation plus réaliste, tout en optant pour un solveur a priori optimal d'un point de vue rapidité de calcul (solveur à pas variable). Il apparaît que les erreurs relatives moyennes sont acceptables (autour de 2 %). Le solveur choisi est donc pertinent et ne présente pas les mêmes difficultés que pour le modèle non triggé. Il semble en effet que l'ajout du générateur d'impulsion force la détection de la dynamique rapide pour ce solveur.

Nous pouvons conclure que pour le modèle non triggé, il est nécessaire d'utiliser un solveur à pas fixe. Un pas égal à 0,1 est suffisant. Pour le modèle triggé, les solveurs à pas fixe et variable conviennent.

7.4. Examen du temps de calcul

Pour chaque configuration, la durée de calcul est directement proportionnelle au nombre d'histoires jouées (cf. figure 83). Afin de pouvoir comparer les performances temporelles, nous calculons la durée d'une histoire pour chaque cas de figure. Cette durée est ensuite normalisée en divisant le temps de calcul par l'intervalle de temps simulé T. On obtient le temps moyen par histoire et par unité de temps simulée. Ces valeurs sont synthétisées dans le tableau suivant :

N° simulation	Modèle	Durée par histoire et par unité de temps simulées
1	Non triggé	$1{,}13 \cdot 10^{-4}$
2	Non triggé	$3{,}87 \cdot 10^{-7}$
3	Non triggé	$2{,}38 \cdot 10^{-3}$
4	Non triggé	$8{,}49 \cdot 10^{-6}$
5	Non triggé	$2{,}19 \cdot 10^{-4}$
6	Triggé	$2{,}19 \cdot 10^{-4}$
7	Triggé	$2{,}14 \cdot 10^{-4}$
8	Triggé	$8{,}85 \cdot 10^{-5}$

Figure 82 : Comparaison des performances en temps de calcul

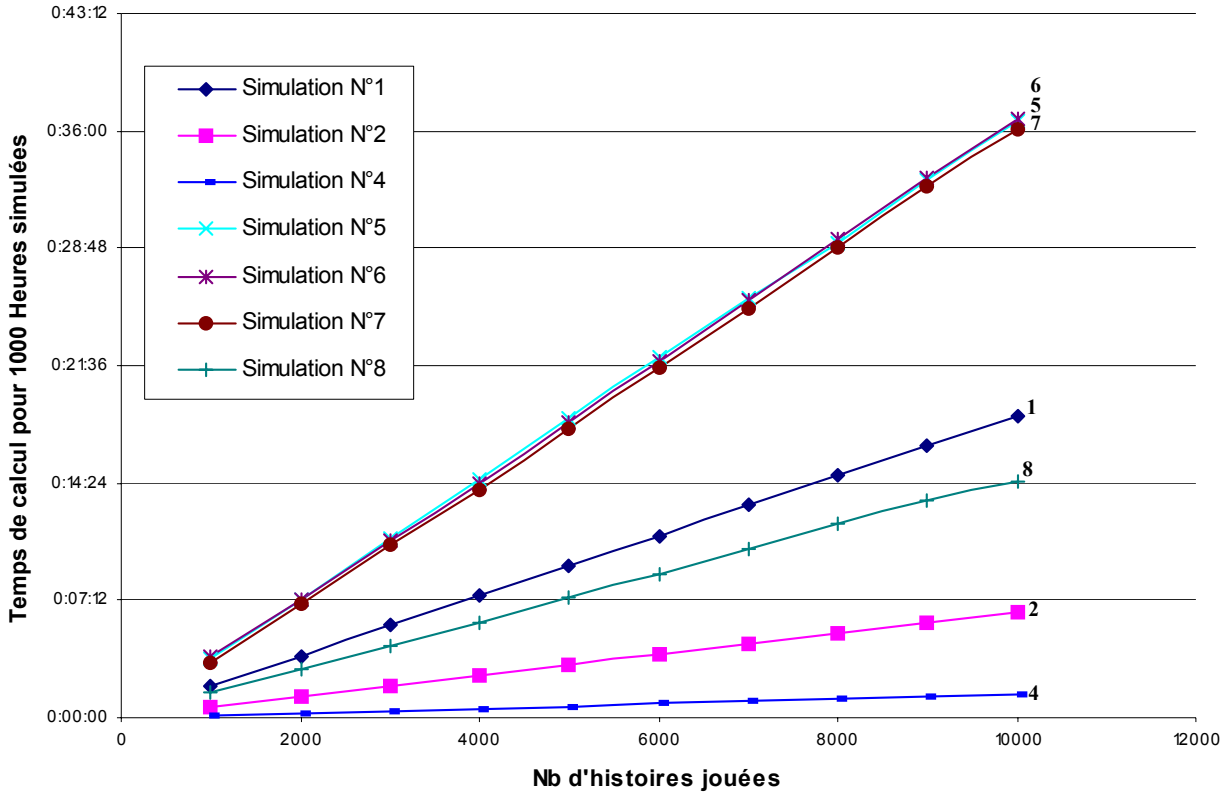


Figure 83 : Evolution des temps de calcul en fonction du nombre d'histoires de durée T=1000H

Quel que soit le modèle traité, le solveur le plus rapide est à pas variable (Ode45). Cependant, les courbes de probabilité associées révèlent une erreur relative et absolue trop importante par rapport aux résultats analytiques dans le cas du modèle non triggé. Une optimisation dans le réglage des options peut améliorer le compromis temps de simulation/erreur commise.
Néanmoins, nous constatons que pour certaines configurations, l'erreur relative est faible et la durée de calcul acceptable (de l'ordre de 2H de calcul pour 10 000 histoires simulées). Ceci est le cas pour les simulations 1 et 6. Cependant, l'ordre de grandeur des taux de défaillance n'est pas représentatif des taux réels. Une simulation avec les mêmes configurations, mais avec des taux beaucoup plus faibles engendrerait des durées de calculs largement plus grand.
La simulation 8 a exigé plus de 36 heures de calcul, pourtant avec un solveur à pas variable. Malheureusement, au terme de cette simulation, le régime permanent n'est pas atteint. La durée nécessaire peut être estimée : $8{,}85.10^{-5}$ x T x N ≈ $8{,}85.10^{-5}$ x 3.10^{5} x 10^{4} soit environ 70 heures. La diminution supplémentaire de l'ordre de grandeur des taux de défaillance par un facteur 10^{x} engendre une multiplication du temps de simulation par ce même facteur. Il devient alors difficilement envisageable d'employer la technique de simulation de Monte-Carlo pour des systèmes de taille plus importante ou de complexité accrue.

Pour conclure cette analyse, la durée de simulation de 10 000 histoires avec un rapport des dynamiques de l'ordre de 10^{8} nécessite environ 1000 heures de traitement informatique avec un solveur à pas fixe (Ode1 avec un pas = 0,1).
En comparaison, la recherche de l'ensemble des coefficients de l'annexe 4 a nécessité une heure de simulation, en omettant l'analyse préalable du problème (recherche des macro-états).

7.5. Conclusion sur la comparaison des deux approches

La simulation de Monte-Carlo est intéressante dans le sens où la mise en place de cette méthode ne nécessite aucune hypothèse restrictive sur le système, et la modélisation dysfonctionnelle est relativement simple à intégrer dans le modèle global. Cependant, un choix et un réglage judicieux des options doivent être effectués sur le solveur afin que la simulation n'engendre pas des temps de calcul prohibitifs ou l'omission d'une dynamique lors de la simulation. Néanmoins, la présence de plusieurs dynamiques au sein du système implique plusieurs dizaines d'heures de calculs au minimum.

L'application de la méthode analytique par les graphes de Markov agrégés nécessite tout de même une mise en œuvre de la simulation, mais uniquement en présence d'une dynamique (simulation fonctionnelle de chaque mode dégradé, sans possibilité de transition de défaillance). La technique de la simulation est exploitée uniquement en vue de recueillir des informations synthétisant statistiquement la dynamique interne du système ; ces informations prennent la forme de « temps de séjour » dans des états ou places, de « marquages moyens », etc, en fonction de la nature du formalisme.

Remarquons également que la modification des constantes de fonctionnement du système (débits moyens des actionneurs) a une influence sur les courbes de probabilité, puisqu'une augmentation ou une diminution de ces débits va influer sur les temps moyens dans les états ouvert et fermé. Par conséquent, les coefficients de pondération obtenus par simulation sont également modifiés et prennent en compte la dynamique interne du système.

La durée totale de ces simulations n'a pas excédé une heure, mais implique une réflexion supplémentaire dans l'élaboration d'un graphe de Markov représentatif des états dégradés du système. Cette analyse préalable inclut la recherche exhaustive des macro-états, en l'occurrence l'ensemble des modes dégradés et des états absorbants du système. Les transitions entre ces macro-états doivent également être identifiées. Une fois l'ensemble des éléments nécessaires déterminés, la résolution numérique est « immédiate » (résolution d'un simple système d'équations différentielles d'ordre 1).

En outre, ce graphe fournit une bonne représentation des séquences des défaillances qui aboutissent à un événement redouté.

CONCLUSION GENERALE

Conclusion générale

Ces dernières décennies ont vu apparaître de nouvelles méthodes théoriques et outils d'analyse fonctionnelle et de sûreté de fonctionnement. Les progrès constants de l'informatique en matière de capacité et de vitesse de traitement ont permis d'introduire et d'appliquer ces nouvelles méthodes sur des systèmes de plus en plus complexes, autrefois réservées au domaine universitaire. Cette constatation est vérifiée en particulier pour les méthodes formelles et les outils de de modélisation.
Cependant, cette multiplication de méthodes et d'outils ne peut être bénéfique que si ces derniers s'intègrent dans un cadre méthodologique bien défini. Il n'est pas envisageable de bouleverser radicalement les habitudes des concepteurs, ni d'introduire brutalement ces outils et ces méthodes sans se soucier de leur cohérence dans la démarche de conception.

Ainsi, l'objectif des travaux présentés dans ce mémoire a consisté précisément à identifier les éléments permettant d'améliorer la démarche de développement au regard de ce constat, à savoir : le choix d'un formalisme de modélisation, la vérification et la validation d'un modèle et l'évaluation de la sûreté de fonctionnement.

Il s'agissait également de prendre en considération la spécificité des systèmes embarqués. La nature hybride, c'est-à-dire mélangeant les aspects continus et discrets et la nature temps réel des systèmes de contrôle-commande ont orienté notre recherche méthodologique.

Il apparaît que les formalismes à état-transition sont les mieux adaptés pour représenter la nature discrète des systèmes de contrôle-commande et pour exploiter les différents outils d'analyse formelle et de sûreté de fonctionnement. Couplé à un formalisme de représentation de la dynamique continue, le comportement global peut être observé et analysé. Nous avons ainsi fait un état des lieux sur les outils existant permettant de modéliser ces systèmes hybrides.

Nous ne prétendons pas substituer les techniques d'analyse existantes mais, au contraire, les compléter à l'aide de nouveaux outils. Ainsi, la simulation classique reste une étape importante dans la vérification et la validation d'un modèle, en particulier lorsque celui-ci est enrichi par une quantité d'informations ne pouvant être examinée différemment. La simulation reste également incontournable pour les mesures de performance temporelle.

Nous avons également mis l'accent sur l'avantage procuré par les méthodes formelles telles que le model-checking à propos de leur capacité à éliminer les fautes de conception. Bien que cette technique ne puisse pas traiter des modèles très détaillés et nécessite de les conditionner en conséquence, leur application s'avère néanmoins être complémentaire à la simulation ou à

la génération de jeux de tests. L'utilisation d'un formalisme à état-transition pour modéliser les systèmes de contrôle-commande facilite d'autant mieux l'application de cette technique de vérification formelle dans un cycle de conception.

Le dernier point considéré dans ces travaux traite du domaine de la sûreté de fonctionnement. De nombreuses méthodes existent et s'appliquent dans différents secteurs industriels. Bien que les principes fondamentaux de construction d'un modèle dysfonctionnel et d'évaluation qualitative/quantitative soient relativement simples, il n'en demeure pas moins que celles-ci restent limitées à des architectures adaptées : nombre de composants ou de fonctions limité, structure statique, hypothèses d'indépendance des défaillances… Généralement, les méthodes sont éprouvées pour des systèmes purement mécaniques. Les méthodes statiques telles que les arbres de défaillance pour lesquelles un dysfonctionnement est la conséquence d'une combinaison statique de défaillances, conviennent assez bien. De plus, cette méthode, basée sur la logique combinatoire, est représentable graphiquement et offre aux divers protagonistes un support d'échange des études réalisées ou en cours. La compréhension des mécanismes de défaillance est améliorée.
La prise en compte du temps dans l'évaluation de la sûreté de fonctionnement n'est cependant pas possible avec les arbres « classiques ». Les méthodes graphiques telles que les processus de Markov répondent à ces besoins mais sont fortement contraints par des hypothèses lourdes. Les problèmes d'explosion combinatoire constituent également un frein à son exploitation dans le milieu industriel.

Ainsi, la solution permettant une grande souplesse de modélisation et d'évaluation de la sûreté de fonctionnement est la simulation de Monte-Carlo. Des méthodes d'accélération ont été mises au point mais impliquent souvent une adaptation du modèle. De plus, cette technique n'exploite pas un modèle dysfonctionnel synthétisant l'ensemble des combinaisons de pannes à l'instar des deux méthodes précédentes.

Au regard des limites de toutes ces méthodes évoquées, l'approche proposée tente d'exploiter les avantages de ces méthodes et d'en contourner les limites. Nous en rappelons les principaux concepts :

- La méthode exploite un **formalisme graphique** : les graphes de Markov.
- La **dynamique lente** est représentée par un graphe de Markov réduit. La structure est une abstraction des mécanismes de défaillances ordonnées dans le temps, provoquant le passage d'un mode de fonctionnement à un autre mode (reconfiguration, dégradé, état redouté).
- La **dynamique rapide**, en interaction avec ce processus stochastique, est représentée par un ensemble de constantes synthétisant statistiquement le comportement du système dans chacun des modes de fonctionnement. Elles sont fonctions de grandeurs continues (variables physiques) ou discrètes (temps de séjours dans des états élémentaires de la partie contrôle-commande…).
- Le graphe de Markov réduit est **construit directement** à partir de la connaissance des modes de fonctionnement. Contrairement à de nombreuses méthodes d'agrégation nécessitant un regroupement d'états élémentaires, l'identification de ces macro-états a priori permet d'éviter de générer une explosion combinatoire.
- Les coefficients de pondération, résumant la dynamique rapide, sont évalués par des **simulations**. Le traitement exclusif de la dynamique rapide n'engendre pas des temps de calcul prohibitifs. De plus, l'utilisation de cette technique évite à l'analyste d'adapter son modèle en fonction des contraintes de modélisation ou d'hypothèses trop fortes.

Ainsi, la philosophie générale de notre approche est basée sur un découplage des dynamiques. La taille réduite du graphe de Markov améliore la compréhension de ce modèle. De plus, l'utilisation de la simulation permet d'élargir la souplesse de modélisation, puisque la méthode n'est pas tributaire d'un formalisme particulier. L'exemple du réservoir modélisé par Simulink/Stateflow illustre la possibilité d'exploiter des représentations de haut niveau.

Soulignons néanmoins que l'application de cette méthode est fondée sur le principe de la théorie des perturbations singulières. Ainsi, la validité des résultats dépend de la séparation des dynamiques. Au regard des échelles de temps manipulées, cette hypothèse est largement vérifiée. De plus, contrairement à une simple simulation de Monte-Carlo, les acteurs chargés d'élaborer un tel graphe de Markov doivent posséder une bonne maîtrise du système : modes de reconfiguration, composants matériels pouvant défaillir, modes de défaillance, événements redoutés, … La connaissance du système ne s'en trouve alors qu'améliorée.

La méthode des graphes de Markov agrégés répond donc aux besoins du concepteur grâce à la possibilité de disposer d'un modèle graphique, et permet d'évaluer la fiabilité des systèmes dynamiques hybrides sous certaines conditions.

Afin de pérenniser la méthode, il serait intéressant de développer un atelier logiciel intégrant tous les éléments méthodologiques identifiés. Cet atelier permettrait ainsi de structurer toutes les activités d'analyse fonctionnelle et dysfonctionnelle autour d'un formalisme unique de modélisation hybride. En fonction des analyses à effectuer, ce modèle de haut-niveau serait décliné par des techniques de conditionnement ou d'abstraction afin de générer automatiquement des modèles formels ou de sûreté de fonctionnement, ou au moins d'en faciliter la construction par des règles à définir.

Le développement de systèmes automobiles de plus en plus complexes et critiques vis-à-vis de la disponibilité et de la sécurité, tels que les systèmes X-by-Wire, confortera incontestablement la nécessité de disposer d'une méthodologie globale dans laquelle les activités de sûreté de fonctionnement prendront une place importante.
A travers ce mémoire, nous avons essayé de contribuer modestement à cette quête.

ANNEXES

ANNEXE 1 : SCRIPTS MATLAB

ANNEXE 2 : QUELQUES RAPPELS SUR LES RESEAUX DE PETRI

ANNEXE 3 : EXEMPLE DU SYSTEME DE CONTROLE-COMMANDE

ANNEXE 4 : DONNEES DU CAS TEST « RESERVOIR »

Annexe 1 : Scripts Matlab

1. Tracé des cercles de Gershgorin

Les trois scripts Matlab suivants (gersh, seqa et cpltaxes) permettent de tracer les cercles de Gershgorin d'une matrice carrée A.
Ils sont issus de la toolbox « The Test Matrix Toolbox for Matlab Version 3 » disponible sur le site web de l'université de Manchester : http://www.ma.man.ac.uk/MCCM/MCCM.html
et téléchargeable à partir du lien ftp : ftp://ftp.ma.man.ac.uk/MCCM/pub/narep

Script **gersh.m** :

```
function  [G, e] = gersh(A, noplot)
%GERSH    Gershgorin disks.
%         GERSH(A) draws the Gershgorin disks for the matrix A.
%         The eigenvalues are plotted as crosses `x'.
%         Alternative usage: [G, E] = GERSH(A, 1) suppresses the plot
%         and returns the data in G, with A's eigenvalues in E.
%
%         Try GERSH(LESP(N)) and GERSH(SMOKE(N,1)).

if diff(size(A)), error('Matrix must be square.'), end

n = max(size(A));
m = 40;
G = zeros(m,n);

d = diag(A);
r = sum( abs( A.'-diag(d) ) )';
e = eig(A);

radvec = exp(i * seqa(0,2*pi,m)');

for j=1:n
    G(:,j) = d(j)*ones(size(radvec)) + r(j)*radvec;
end

if nargin < 2

   ax = cpltaxes(G(:));
   for j=1:n
       plot(real(G(:,j)), imag(G(:,j)),'-c5')      % Plot the disks.
       hold on
   end
   plot(real(e), imag(e), 'xg')    % Plot the eigenvalues too.
   axis(ax);
   axis('square');
   hold off

end
```

Script **seqa.m** :

```
function y = seqa(a, b, n)
%SEQA   Additive sequence.
%       Y = SEQA(A, B, N) produces a row vector comprising N equally
%       spaced numbers starting at A and finishing at B.
%       If N is omitted then 10 points are generated.

if nargin == 2, n = 10; end

if n <= 1
   y = a;
   return
end
y = [a+(0:n-2)*(b-a)/(n-1), b];
```

Script **cpltaxes.m** :

```
function x = cpltaxes(z)
%CPLTAXES   Determine suitable AXIS for plot of complex vector.
%           X = CPLTAXES(Z), where Z is a complex vector,
%           determines a 4-vector X such that AXIS(X) sets axes for a
plot
%           of Z that has axes of equal length and leaves a reasonable
amount
%           of space around the edge of the plot.

%           Called by FV, GERSH, PS and PSCONT.

% Set x and y axis ranges so both have the same length.

xmin = min(real(z)); xmax = max(real(z));
ymin = min(imag(z)); ymax = max(imag(z));

% Fix for rare case of `trivial data'.
if xmin == xmax, xmin = xmin - 1/2; xmax = xmax + 1/2; end
if ymin == ymax, ymin = ymin - 1/2; ymax = ymax + 1/2; end

if xmax-xmin >= ymax-ymin
   ymid = (ymin + ymax)/2;
   ymin =  ymid - (xmax-xmin)/2; ymax = ymid + (xmax-xmin)/2;
else
   xmid = (xmin + xmax)/2;
   xmin = xmid - (ymax-ymin)/2; xmax = xmid + (ymax-ymin)/2;
end
axis('square')

% Scale ranges by 1+2*alpha to give extra space around edges of plot.

alpha = 0.1;
x(1) = xmin - alpha*(xmax-xmin);
x(2) = xmax + alpha*(xmax-xmin);
x(3) = ymin - alpha*(ymax-ymin);
x(4) = ymax + alpha*(ymax-ymin);

if x(1) == x(2), x(2) = x(2) + 0.1; end
if x(3) == x(4), x(4) = x(3) + 0.1; end
```

Le script Matlab suivant décrit la fonction de calibrage d'une matrice carrée M quelconque :

Script **calibre.m** :

```
function [Acalibr, Dcalibr, Rapport, kiter]=calibre(M)
% Fonction de calibrage de la matrice M

%-----------------------------------------------------

% Initialisation des variables

n=length(M);     % Taille de la matrice
k=1;             % Nombre d'itérations = nombre de calculs de D
A=M;             % A = Matrice calibrée
D=eye(n);        % Matrice de calibrage (diagonale)
condition=0;     % Condition d'arret
e=0.05;          % Critère d'arret (precision du rapport des rayons)


% Recherche de la matrice de calibrage, tant que les rayons des
% cercles ne sont pas égaux à e près

while (condition==0),

    d(1:n)=0;              % éléments de la matrice D pour l'itération k

    for i=1:n,

        m=lignemax(A);      % ligne m dont la somme des modules
                            % des éléments non diagonaux est max


        % Recherche de la valeur de d(i):

        if sum(abs(A(i,1:n)))==sum(abs(A(m,1:n)))
            d(i)=1;
        else

            solution(1:2)=0;

            v0 = sum(abs(A(i,1:n))) - abs(A(i,i));
            v1 = sum(abs(A(m,1:n))) - abs(A(m,m)) - abs(A(m,i));
            v2 = abs(A(m,i));

            solution=roots([v2;v1;-v0]); % solution(s) de l'équation du
                                         % second degré
            if length(solution)==1
                d(i)=solution(1);
            else

                if solution(1)>0 & solution(1)<1
                    d(i)=solution(1);
                else
                    d(i)=solution(2);
                end
            end
```

```
% Nouvelle matrice de calibrage D
            D(i,i)=D(i,i)*d(i);

            % Transformation de la matrice A

             for j=1:n,
                if j~=i,
                    A(j,i)=A(j,i)*d(i);
                    A(i,j)=A(i,j)/d(i);
                end
             end

         end

    end


    k=k+1;

    % Vérification de la condition d'arret
    RayMin=min(Rg(A));
    RayMax=max(Rg(A));
    condition=(RayMin/RayMax>1-e);

end

figure(1);
subplot(2,1,1); plot(gersh(M,1))
title('Cercles de Gershgorin d''origine et après calibrage de la
matrice M')
grid;
subplot(2,1,2); plot(gersh(A,1))
grid;
disp(['Nombre d''itérations : ' num2str(k)])

Acalibr=A; Dcalibr=D; Rapport=RayMin/RayMax; kiter=k;
```

Le script suivant donne l'indice m de la ligne d'une matrice dont la somme des éléments hors diagonaux est maximale :

Script **lignemax.m** :

```
function [m]=lignemax(A)

r=length(A);      somme(1:r)=0;

for i=1:r,
    for j=1:r,
        if i~=j,
            somme(i)=somme(i)+abs(A(i,j));
        end
    end
end

[max,ligne]=max(somme);
m=ligne;
```

Le script suivant décrit la fonction Rg donnant les rayons et les centres des cercles de Gershgorin associés à une matrice carrée M :

Script **Rg.m** :

```
function [R,g]=Rg(M)

% Rayons / Centres des matrices

n=length(M);
R=zeros(n,1);

for i=1:n,
    for j=1:n,
        if (j ~= i)
            R(i)=R(i)+abs(M(i,j));
        end
        g(i)=-R(i);
    end
end
```

2. Analyse de performance de la méthode d'agrégation sur des familles de processus markoviens. Evaluation des temps de calcul

Le script suivant permet d'estimer les temps de calcul nécessaires à l'obtention des courbes de probabilités pour les processus de Markov présentés dans le paragraphe 4.4. du chapitre 3. Ces calculs sont réalisés en fonction du couple (n, r), et le rapport des temps est représenté graphiquement en fonction de (n, r).

```
% ================================================
%       EVALUATION DU GAIN EN TEMPS DE CALCUL
%       DES PROBABILITES ENTRE LE PROCESSUS
%       ORIGINAL ET LE PROCESSUS REDUIT
% ================================================
%   Etude en fonction de la taille du graphe
%   et du rapport entre les deux échelles de temps
% ================================================
%       Raphael SCHOENIG  --- 28 nov. 2003


clear all; warning off;

% Taux de transitions "rapides" normalisés:
  r1 = 1;  r2 = 10;  r3 = 1;
% Taux de transition lent:
  lamb = 1;

% Les taux de transition rapides sont obtenus en multipliant
% les taux normalisés par un facteur "r" (rapport des ordres
% de grandeur
```

```
for r=1:10,

    % Taux "rapides" réels
    a1=r*r1;   a2=r*r2;    a3=r*r3;
    % Probabilité conditionnelle (coeff de pondération)
    p=a2*a3/(a1*a2+a1*a3+a2*a3);


    % On fait varier la taille du graphe.
    % N=Nb de Macro-états transitoires
    % Taille(M)=3N+1,    Taille(Mréduite)=N+1

    for N=10:60,

        % On résout le système initial et le système réduit par des
        % équations d'état: X'=A'X et Xr'=Ar'Xr

        A=M(10*r,N)';           B=zeros(3*N+1,1);
        C=zeros(3*N+1,3*N+1);   C(3*N+1,3*N+1)=1;
        D=B;                    X0=B; X0(1,1)=1;

        Ar=Mr(p,N)';            Br=zeros(N+1,1);
        Cr=zeros(N+1,N+1);      Cr(N+1,N+1)=1;
        Dr=Br;                  X0r=Br; X0r(1,1)=1;


        % Résolution à l'aide de blocs Simulink
        disp([' r = ' num2str(r) '      N = ' num2str(N)])

        tic;        sim('Serie2', [0 20+4*N]);
        toc;        mesure1(N,r)=toc;

        tic;        sim('Serie3', [0 20+4*N]);
        toc;        mesure2(N,r)=toc;

        %figure(1); plot(t,ER(:,3*N+1),t,ERr(:,N+1)); grid;
        Rapport_temps(N,r)=mesure1(N,r)./mesure2(N,r);


    end
end

figure(1);
colormap(cool);
surf(Rapport_temps);
ylabel('Taille N de la matrice de transition ')
xlabel('Rapport r des dynamiques');
```

Ce script fait appel aux fonctions M et Mr représentant les matrices de transition du graphe initial et du graphe réduit. Ces deux fonctions ont respectivement pour argument en entrée : (r,n) et (p, n) avec n le nombre de macro-états transitoires, r le rapport des échelles de temps, et p la probabilité conditionnelle asymptotique correspondant au coefficient de pondération dans le processus réduit.

Script **M.m** :

```
function [matrice]=M(rapport,taille)
% Définition d'une matrice de transition (transposée)

% Taux de transition normalisés
r1=1; r2=10; r3=1;       % taux rapides
lamb=1;     % taux lent

% Taux réels
a1=rapport*r1;   a2=rapport*r2;   a3=rapport*r3;

% Probabilité conditionnelle (coeff. de pondération)
p=a2*a3/(a1*a2+a1*a3+a2*a3);

N=taille;

% Définition des matrices de transitions M
% Taille(Mat)=3N+1

Mat=zeros(3*N+1,3*N+1);

     for i=1:N,
         Mat(3*i-2,3*i-2)=-(a1+lamb);
         Mat(3*i-2,3*i-1)=a1;
         Mat(3*i-1,3*i-1)=-a2;
         Mat(3*i-1,3*i)=a2;
         Mat(3*i,3*i-2)=a3;
         Mat(3*i,3*i)=-a3;
         Mat(3*i-2,3*i+1)=lamb;
     end

% Cas d'une chaine bouclée
% Mat(3*N+1,1)=lamb; Mat(3*N+1,3*N+1)=-lamb;

 matrice=Mat;
```

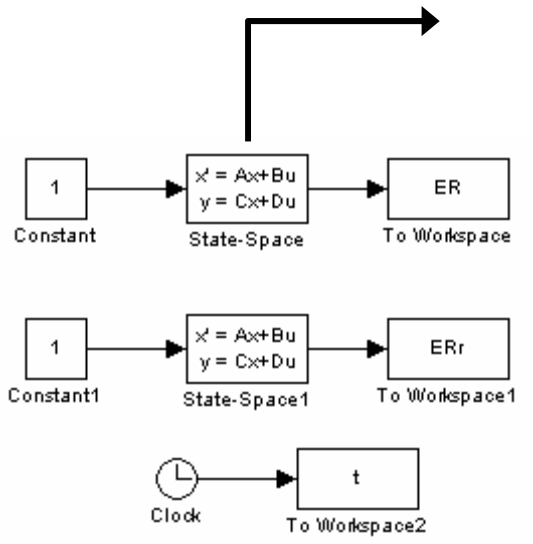
Script **Mr.m** :

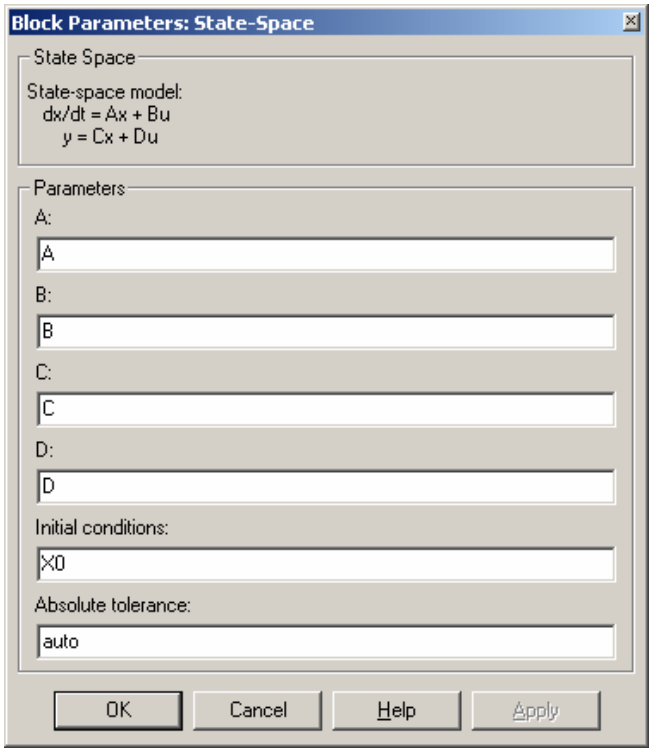
```
function [matrice_reduite]=Mr(p,N)
% Définition de la matrice de transition réduite
% assiciée à M

lamb=1;
Matrice=zeros(N+1,N+1);

for i=1:N
    Matrice(i,i)=-p*lamb;
    Matrice(i,i+1)=p*lamb;
end

% Cas d'une chaine bouclée
% Matrice(N+1,1)=p*lamb;    Matrice(N+1,N+1)=-p*lamb;

matrice_reduite=Matrice;
```

3. Analyse de performance : estimation de l'erreur absolue maximale des probabilités $Pr(M_n)$ entre le processus initial et réduit

```
% ------------------------------------------------------------------
% EVALUATION DE L'ERREUR ABSOLUE MAXIMALE ENTRE LES PROBABILITES
% D'ARRIVER DANS LE MACRO-ETAT FINAL DU PROCESSUS INITIAL ET REDUIT.
% ------------------------------------------------------------------
%             Evaluation en fonction du couple (n,r)
% ------------------------------------------------------------------
%               Raphael SCHOENIG  --- 28 nov. 2003

clear all; warning off;

    r1=1; r2=10; r3=1;
    p=r2*r3/(r1*r2+r1*r3+r2*r3);    % Coeff de pondération


for r=1:25,

    for N=1:25,

        % Equations d'états du processus initial
        A=M(r,N)';
        B=zeros(3*N+1,1);
        C=zeros(3*N+1,3*N+1); C(3*N+1,3*N+1)=1;
        D=B;
        X0=B; X0(1,1)=1;

        % Equations d'états du système réduit
        Ar=Mr(p,N)';
        Br=zeros(N+1,1);
        Cr=zeros(N+1,N+1); Cr(N+1,N+1)=1;
        Dr=Br;
        X0r=Br; X0r(1,1)=1;

        % Calcul des probabilités
        sim('Serie1', [0 20+30*N]);

        % Différence des 2 courbes de probabilités
        erreur=ER(:,end)-ERr(:,end);

        % Recherche de l'erreur maximale
        abs_err(N,r)=max(abs(erreur));
    end
end

figure(1);
surf(abs_err);
ylabel('Taille N de la matrice de transition ')
xlabel('Rapport r des dynamiques');
```

Ce script fait appel aux deux fonctions M et Mr décrites dans le paragraphe précédent. La résolution simultanée des deux systèmes d'équations différentielles se fait via Simulink en utilisant des représentations sous forme d'équations d'états (cf. figure suivante).





4. Evaluation du critère de séparabilité des dynamiques

Script **M.m** :

```
function [matrice]=M(lamb,taille)
% Définition d'une matrice de transition (transposée)

% Taux de transition normalisés
r1=1;       % taux rapide
r2=10;      % taux rapide
r3=1;       % taux rapide
rapport=100;

% Taux réels
a1=rapport*r1;
a2=rapport*r2;
a3=rapport*r3;

% Probabilité conditionnelle (coeff. de pondération)
p=a2*a3/(a1*a2+a1*a3+a2*a3);

N=taille;

% Définition des matrices de transitions M
% Taille(Mat)=3N+1

Mat=zeros(3*N+1,3*N+1);
```

```
    for i=1:N,
        Mat(3*i-2,3*i-2)=-(a1+lamb);
        Mat(3*i-2,3*i-1)=a1;
        Mat(3*i-1,3*i-1)=-a2;
        Mat(3*i-1,3*i)=a2;
        Mat(3*i,3*i-2)=a3;
        Mat(3*i,3*i)=-a3;
        Mat(3*i-2,3*i+1)=lamb;
    end

matrice=Mat;
```

Cette fonction définit la matrice de transition M de la chaîne de Markov en fonction de N (nombre de macro-états transitoires) et de lambda. Elle est légèrement différente de la fonction M définie précédemment, puisque le premier argument correspond directement au taux de défaillance, et non au rapport des dynamiques.

Script **mat_L.m** :

```
function [ Linf ] = mat_L( A11, A12, A21, A22 )

L=(inv(A22))*A21;

for i=2:30,
    L=(inv(A22))*((A21+L*A11-L*A12*L));
    norm(L);
end

Linf=L;
```

Ce script recherche la solution à l'équation de Riccati $\Re(L)=0$ permettant de bloc-trianguliser une matrice A (cf. chapitre 3). L'algorithme est basé sur la recherche de la solution limite de la suite définie dans le chapitre 3. Un nombre minimal d'itérations est suffisant.

Script **mat_M.m** :

```
function [ Minf ] = mat_( A11, A12, A21, A22,L )

M=A12*inv(A22);
%L=inv(A22)*A21;

for i=2:30,
    M=((A11-A12*L)*M-M*L*A12)*inv(A22)+(A12*inv(A22));
end

Minf=M;
```

Le principe de cet algorithme est identique au précédent.

Script **mat_P.m** :

```
function [ P ] = mat_P( M )

P=rref((null(M'))');
```

Ce script génère une matrice P dont les lignes forment une base du noyau de la transposée d'une matrice M.

Script **mat_Q.m** :

```
function [ Q ] = mat_Q( M )

Q=rref(orth(M')');
```

Ce script génère une matrice Q constituée de n lignes indépendantes de la transposée d'une matrice M.

Script de calcul du critère de séparation des dynamiques :

```
% ======================================================================
% Algorithme de vérification de la séparation des dynamiques
%                   Méthode algébrique
%
%   EVOLUTION DU TAUX DE SEPARATION EN FONCTION DE LA TAILLE
%                       DE LA MATRICE M
%           ET EN FONCION DU TAUX DE DEFAILLANCE
%
% Etablissement du critère de séparation
% Découplage des dynamiques et bloc-diagonalisation de la matrice
% ----------------------------------------------------------------------
% Raphael SCHOENIG          28/11/2003
% ----------------------------------------------------------------------
%
% ======================================================================

clear all;      warning off;


for N=1:10,

        n=size(mat_P(M(0,N)));
        n1=n(1);            % n1=nombre de variables lentes
        n2=n(2)-n1;         % n2=nombre de variables rapides

        T=[mat_P(M(0,N))    % Matrice passages
            mat_Q(M(0,N))];


        for lambda=10:10:1000,

            % Nouvelle matrice dans la base T:
              M2=T*M(lambda,N)*inv(T);
```

```
            % Bloc diagonalisation de M2 afin de découpler les parties
            % lentes et rapides.Il faut rechercher la matrice de passage

            % Recherche de la matrice L solution de l'équation de
            % Riccati :       A21 + L*A11-L*A12*L-A22*L=0

            A11=M2(1:n1,1:n1);
            A12=M2(1:n1,1+n1:n1+n2);
            A21=M2(n1+1:n1+n2,1:n1);
            A22=M2(n1+1:n1+n2,1+n1:n1+n2);

            L=mat_L(A11,A12,A21,A22); %Solution de l'équation de Riccati

            % Calcul des matrices rapide Ar et lente Al en fonction de L
            Al=A11-A12*mat_L(A11,A12,A21,A22);
            Ar=A22+mat_L(A11,A12,A21,A22)*A12;

            separation(lambda/10,N)=norm(inv(Ar))*norm(Al);

      end
end

surf(separation)
title('Critere de séparation des dynamiques ')
ylabel('lambda    x10');xlabel('Nombre de Macro-états N')
```

Annexe 2 : Quelques rappels sur les Réseaux de Petri

Cette partie est consacrée au rappel de quelques notions fondamentales de la théorie des réseaux de Petri autonomes, et de quelques extensions en réseaux de Petri non autonomes. Ces notions ne sont pas présentées formellement, mais elles sont suffisantes pour comprendre l'utilisation de ce formalisme dans le contexte des travaux de ce mémoire. Des descriptions plus formelles et les autres extensions des réseaux de Petri peuvent être trouvées dans les ouvrages [DAV89], [BRA83].
Les réseaux de Petri ont été inventés en 1962 par le mathématicien allemand Karl Adam Petri pour décrire le comportement des automates séquentiels asynchrones. Le concept de base a été ensuite largement développé par de nombreux auteurs en adaptant la théorie initiale selon leur propre besoin de modélisation et d'analyse. C'est un formalisme graphique et mathématique puissant qui permet de modéliser des phénomènes concurrents, asynchrones, distribués, parallèles, stochastiques. Outre les possibilités de modélisation, le traitement mathématique du modèle à l'aide d'équations d'états (équations algébriques) permet d'analyser qualitativement et quantitativement le système. De ce fait, ce formalisme est souvent utilisé pour modéliser les systèmes de production et étudier les flux. Les possibilités de description de phénomènes aléatoires (stochastiques), couplés aux graphes de Markov en font un outil d'analyse quantitative très puissant s'appliquant au domaine de la sûreté de fonctionnement.

1. Les réseaux de Petri autonomes

1.1. Structure d'un réseau de Petri

C'est un langage graphique composé de deux types de nœuds, les places (représentées par des cercles) et les transitions (représentées par des rectangles). Les nœuds, dont le nombre est fini et non nul, sont reliés par des arcs orientés représentés par des flèches. Un arc relie une place à une transition ou une transition à une place.

> *<u>Définition</u> [LAF91] : Un réseau de Petri R=(P, T, I, O) est un graphe biparti orienté, composé de deux ensembles distincts et finis de nœuds :*
> *Les places $p \in P$,*
> *Les transitions $t \in T$*
> *I est une application de $PxT \rightarrow \mathbb{N}$ correspondant aux arcs directs des places vers les transitions.*

- *I(p,t)>0 signifie qu'il existe un arc orienté de p vers t et I(p,t) est la valuation ou le poids de cet arc*
- *I(p,t)=0 indique l'absence d'arc orienté de la place p vers la transition t.*

O est une application de TxP→ℕ correspondant aux arcs directs reliant les transitions aux places.

- *O(t,p)>0 signifie qu'il existe un arc orienté de t vers p et O(t,p) est la valuation ou le poids de cet arc.*
- *O(t,p)=0 indique l'absence d'arc orienté entre t et p.*

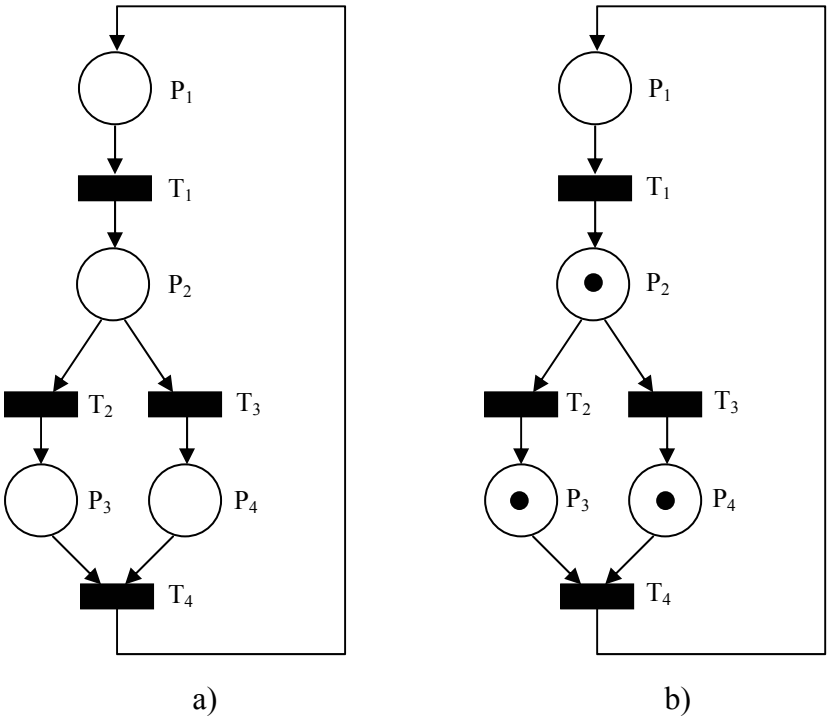
Si I(p,t)>0 (respectivement O(t,p)>0), on dira que la transition t est une transition de sortie (respectivement d'entrée) de la place p. La place p sera dite place d'entrée (respectivement place de sortie) de la transition t.
Lorsque le poids de l'arc est strictement supérieur à 1, celui-ci est indiqué sur l'arc. Si aucune indication n'est précisée, le poids est égal à 1.
On parle de réseau de Petri ordinaire lorsque tous les poids des arcs appartiennent à {0, 1}.

On utilise usuellement la notation °t pour désigner l'ensemble des places d'entrée de la transition t et t° pour l'ensemble des places de sortie de t. Ainsi :

$$°t = \{p \in P \ ; \ I(p,t)>0\}$$
$$t° = \{p \in P \ ; \ O(t,p)>0\}$$

L'ensemble des transitions de sortie de la place p est noté p° et l'ensemble des transitions d'entrée de t est noté °p :

$$°p = \{t \in T \ ; \ O(t,p)>0\}$$
$$p° = \{t \in T \ ; \ I(p,t)>0\}$$



a) b)

Figure 84 : Réseau de Petri a) non marqué et b) marquée

Tous ces éléments constituent la partie statique ou structurelle du réseau. Pour représenter le comportement du système, une structure dynamique est superposée au réseau. Cette structure est composée de jetons ou marques évoluant dynamiquement au sein du réseau.

1.2. Marquage

Le marquage M d'un réseau de Petri est une application de P dans $\mathbb{N}$:

$$M : P \rightarrow \mathbb{N}$$

M(p) représente le nombre de marques ou de jetons présents dans la place p. On les représente par des petits cercles dans les places. Les marques contenues dans les places forment un vecteur colonne dont chaque indice représente un numéro de place. La figure 84 représente deux exemples de réseau de Petri respectivement non marqué et marqué. Le marquage est dans le cas présent :

$$M=[0\ 1\ 1\ 1]^T.$$

Le marquage d'un réseau de Petri évolue dynamiquement par des franchissements de transition selon des règles détaillées par la suite. L'état du marquage représente l'état du réseau à un instant donné.

1.3. Règles de franchissement des transitions

Une transition est tirable ou franchissable si chacune de ses places d'entrée contient un nombre de marques au moins égal à la valuation de l'arc joignant cette place à la transition.
En d'autres termes :

$$\forall p \in {}^\circ t, \qquad M(p) \geq I(p,t)$$

Dans l'exemple de la figure 84, les transitions T_2, T_3 et T_4 sont franchissables. Le franchissement effectif d'une transition t a pour effet :

- de retirer I(p,t) jetons de chaque place d'entrée de la transition t,
- d'ajouter O(t,p) jetons dans chaque place de sortie p de la transition t.

Après franchissement de t, le réseau de Petri possède un nouveau marquage M' égal à :

$$\forall p \in P \qquad M'(p)=M(p)+O(t,p)-I(p,t)$$

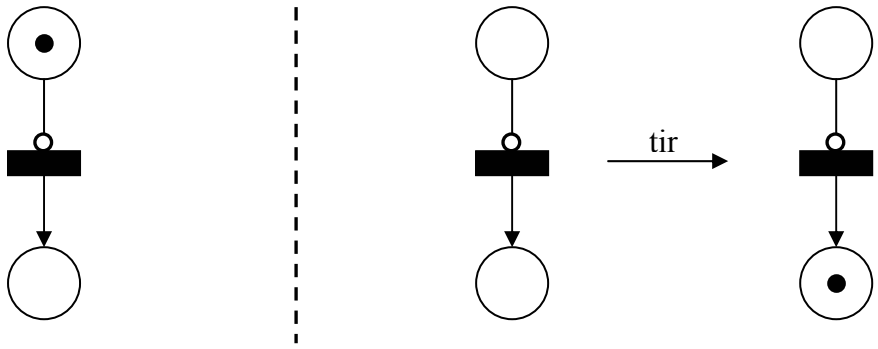
Remarquons dans l'exemple de la figure 84, que l'unicité du jeton dans la place P_2 implique que les transitions T_2 et T_3 ne peuvent pas être franchies toutes les deux. On dit que ces transitions sont en **conflit**.
Après franchissement de T_2 et T_4 le nouveau marquage du réseau est : $M'= [1\ 0\ 1\ 0]^T$.

Une suite de franchissements de transitions à partir d'un marquage donné est appelée **séquence de franchissements**, que l'on note S. Dans le cas présent, la séquence $S=T_4T_2$ fait passer le réseau du marquage M au marquage M'.

On dit que M' est un **marquage accessible** à partir de M, car il existe une séquence de franchissement faisant passer le réseau de M à M'. On note $^*M$ l'ensemble des marquages accessibles à partir de M.

Réseau de Petri avec arc inhibiteur :

Un arc inhibiteur de poids n inhibe la transition si la place à laquelle il est relié contient au moins n jetons. On les représente graphiquement par des arcs dont la flèche est remplacée par un petit cercle :



a) Tir impossible b) Tir possible

Figure 85 : Franchissement de transition avec arc inhibiteur

> *Définition [DAV89]: Réseau de Petri autonome*
> *Un réseau de Petri autonome décrit le fonctionnement d'un système qui évolue de façon autonome, c'est-à-dire dont les instants de franchissement ne sont pas connus ou pas indiqués.*

L'exemple de la figure 84 est un réseau autonome. Ce type de réseau décrit le fonctionnement uniquement de façon qualitative. Cependant, l'utilisation des réseaux de Petri dans le cadre d'un langage de spécification et de conception peut nécessiter de faire interagir le modèle par des événements extérieurs ou dépendre du temps. Des extensions aux réseaux autonomes ont été définies dans cet objectif. Ces réseaux, non-autonomes, sont *synchronisés* (par des événements) et/ou *temporisés*.

1.4. Quelques propriétés des réseaux de Petri

1.4.1. Réseau borné

Un réseau est borné à partir d'un marquage initial M_0, si pour tous les marquages accessibles à partir de M_0, le nombre de jetons dans chaque place p est fini. Si :

$$\forall p \in P,\ \forall M \in {}^*M_0 \qquad M(p) \leq k,$$

alors le réseau est k-borné.
Un réseau est sain s'il est 1-borné.

1.4.2. Vivacité

Un réseau de Petri est vivant à partir d'un marquage initial M_0, si et seulement si pour toute transition t du réseau et tout marquage M accessible, il existe une séquence de franchissement qui contient t.
Autrement dit, quelle que soit l'évolution du marquage du réseau, il existera toujours une possibilité de franchir n'importe quelle transition.

1.5. Graphe des marquages d'un réseau de Petri

Le graphe des marquages est composé de nœuds qui correspondent aux marquages accessibles à partir d'un marquage initial, et d'arcs correspondant aux franchissements de transitions faisant passer d'un marquage à l'autre.
La recherche du graphe des marquages sous-jacent à un réseau de Petri marqué, donne une représentation de l'ensemble des états accessibles. Des propriétés qualitatives et qualitatives peuvent être ensuite extraites de cette nouvelle représentation.
De nombreux outils permettent de générer automatiquement ce graphe d'accessibilité, et de le traiter mathématiquement.

L'exemple de la figure 84 b) donne le graphe des marquages suivant :

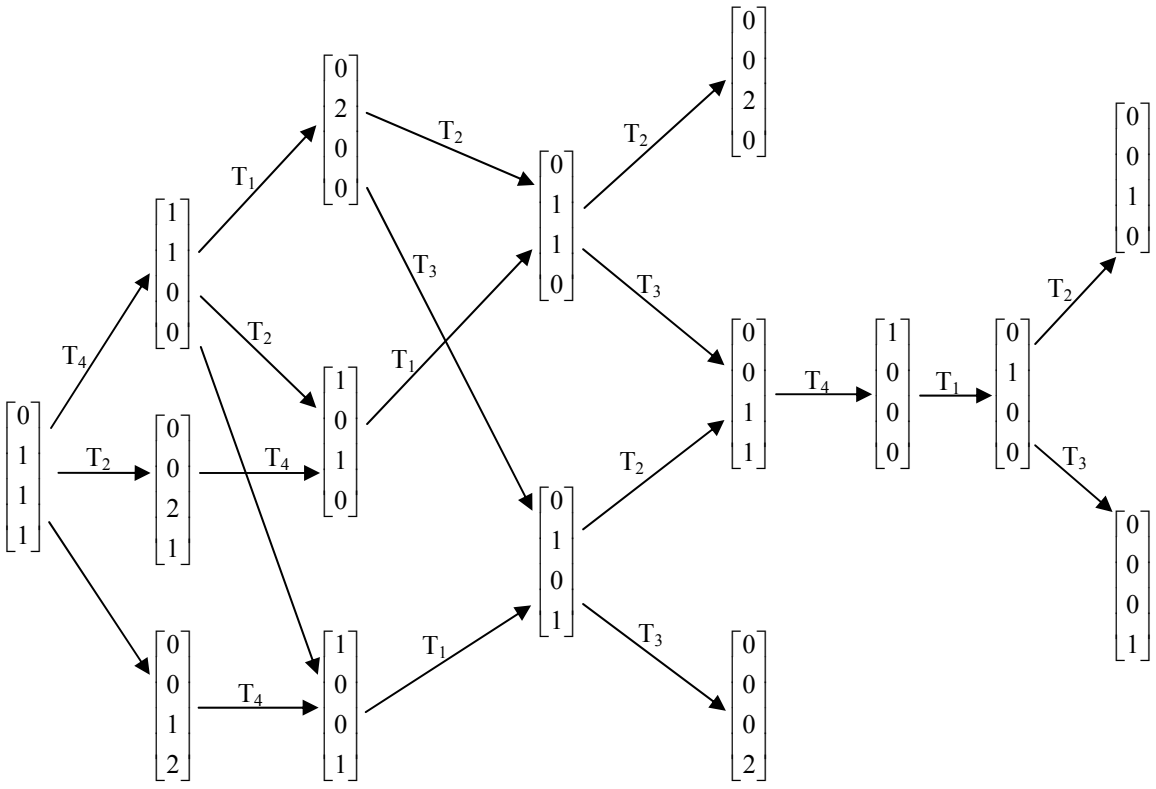


Figure 86 : Graphe des marquages du réseau de Petri marqué

2. Les réseaux de Petri non-autonomes

Afin d'améliorer le pouvoir d'expression et de modélisation des réseaux de Petri autonomes, de nombreuses extensions ont été développées ces dernières années. Ces formalismes étendus ne se limitent plus à des aspects qualitatifs, mais permettent d'enrichir la structure avec la composante temporelle. Nous allons faire un tour d'horizon des quelques extensions les plus importantes.

2.1. Réseaux de Petri synchronisés

Dans les réseaux de Petri synchronisés, à chaque transition est associée un événement, et le franchissement de cette transition s'effectuera si :

- la transition est validée,
- quand l'événement se produira.

Ces réseaux de Petri permettent de faire communiquer et interagir le système modélisé avec l'environnement ou un système extérieur. Ils permettent donc de modéliser des systèmes soumis à des contraintes externes.

2.2. Les réseaux de Petri temporisés

Cette extension se traduit par l'ajout de temporisations, et fait donc figurer explicitement le temps. On distingue les réseaux de Petri T-temporisés et P-temporisés. Dans le premier cas, une durée (constante) est ajoutée aux transitions. Après validation, une transition est franchie à l'issue de ce délai. Dans le second cas, la temporisation est ajoutée aux places.
Ces temporisations permettent notamment de représenter des durées d'opérations ou d'actions associées aux transitions ou aux places.

Les réseaux T-temporisées et P-temporisées sont équivalents. Par transformation, on peut passer d'un formalisme à l'autre.

2.3. Les réseaux de Petri interprétés

Un réseau de Petri interprété a les trois caractéristiques suivantes :

- il est synchronisé,
- il est P-temporisé,
- il comprend une partie opérative dont l'état est défini par un ensemble de variables. Cet état est modifié par des opérations O_i associées aux places.

Aux transitions sont associées des conditions C_j qui autorisent les franchissements en fonction de l'état du système, et donc de la partie opérative, et sur occurrence d'éventuels événements E_j.

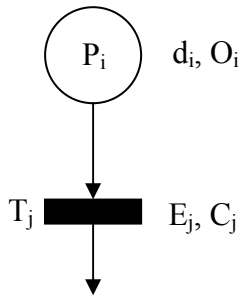


Figure 87 : Sémantique d'un réseau de Petri interprété

Ce type de modélisation permet de couvrir des besoins de modélisation et des domaines d'application assez largement. Le formalisme Grafcet en est un cas particulier, à quelques nuances près.

2.4. Les réseaux de Petri stochastiques

2.4.1. Généralités

Contrairement aux réseaux de Petri temporisés où les délais associés aux nœuds sont constants, certains phénomènes à modéliser impliquent d'intégrer la notion de « hasard ». Les réseaux de Petri stochastiques consistent précisément à introduire des délais aléatoires associées aux transitions. Ces temps sont modélisés par des variables aléatoires. La loi la plus courante est certainement la loi exponentielle qui permet alors de rapprocher le graphe des marquages sous-jacent à un processus markovien homogène.
Les réseaux de Petri stochastiques sont très utilisés dans le domaine de la sûreté de fonctionnement. L'occurrence d'une défaillance, modélisée par une distribution exponentielle, peut être représentée par le franchissement d'une transition du réseau.
Dans le cas d'utilisation de lois exponentielles, par convention le paramètre de la loi est représenté sur la transition (taux de défaillance ou de réparation).

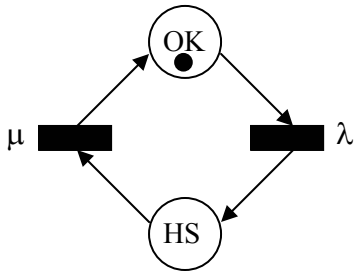


Figure 88 : Modélisation de la défaillance et réparation d'un composant

Les réseaux de Petri stochastiques ont été introduits par Natkin [NAT80] et Molloy [MOL81] afin de répondre à certains problèmes d'évaluation quantitative des systèmes informatiques industriels.

Ils sont obtenus à partir des réseaux de Petri autonomes en associant à chaque transition T_i une durée de franchissement aléatoire d_i. L'hypothèse la plus courante consiste à choisir une distribution selon une loi exponentielle de paramètre λ_i que l'on note habituellement sur la transition concernée. Ce paramètre dépend en général du marquage du réseau.
Cette durée correspond au temps qui s'écoule entre la sensibilisation et le tir effectif de la transition. On peut associer une fonction de répartition à la variable aléatoire « durée de franchissement » :

$$F_i(t) = 1 - e^{-\lambda_i(M)t}$$

où le paramètre $\lambda_i(M)$ dépend du marquage M courant. Ce paramètre est appelé taux de transition relativement au marquage M. Lorsqu'il ne dépend pas du marquage, on l'appelle simplement taux de transition λ_i. La densité de probabilité s'écrit :

$$f_i(t) = \lambda_i e^{-\lambda_i(M)t}$$

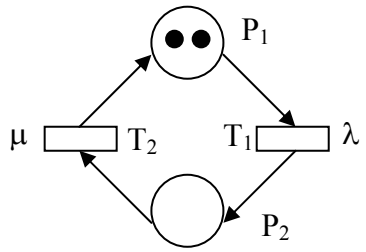


Figure 89 : Exemple de réseau de Petri stochastique

2.4.2. Politique de franchissement :

Un réseau de Petri classique représente un système non temporisé et non déterministe. L'ajout de temporisation stochastique ajoute à ce modèle un comportement temporel aléatoire, mais ne lève pas l'indéterminisme. La manière de traiter cet indéterminisme est très importante, en particulier lorsque plusieurs transitions sont sensibilisées simultanément. Il est donc fondamental d'établir clairement comment le temps interfère avec la planification des opérations liées aux places. Il existe deux façons de choisir la prochaine transition à franchir :

- **Modèle concurrentiel (ou « Race Model »)** : Lorsque plusieurs transitions sont sensibilisées en même temps, la politique du modèle concurrentiel (ou compétitif) consiste à tirer pour chacune d'entre elles le délais de franchissement aléatoire en fonction de la distribution associée à ces transitions. La transition associée à la plus petite valeur de délais sera tirée.
- **Présélection** : Le modèle de présélection consiste, dans un premier temps, à choisir, pour un marquage donné M, une transition T à franchir dans l'ensemble T(M) des transitions sensibilisées, puis à considérer sa durée de sensibilisation. Le choix de la transition est effectué en affectant un poids $\omega_T(M)$ à la transition en fonction du marquage courant. Ces poids définissent une distribution discrète de probabilité sur l'ensemble des transitions validées pour le marquage M. La probabilité de franchissement de T est alors donnée par la formule :

$$\Pr[\text{T choisie} \mid \text{M}] = \frac{\omega_T(\text{M})}{\sum_{S \in T(\text{M})} \omega_S(\text{M})}$$

Cependant cette politique est très lourde à gérer car il faut connaître toutes les configurations de T(M) pour tous les marquages possibles afin d'affecter les distributions discrètes de probabilité. L'avantage principal est de permettre une sélection explicite entre les transitions, et de lever l'indéterminisme en cas de conflit de transitions.

2.4.3. Politique de mémoire

Les classes de réseaux de Petri stochastiques diffèrent par leurs capacités à mémoriser la durée du temps qui s'est écoulée pour chaque transition parallèle. Il existe trois politiques différentes :

- **Age memory** : la durée de sensibilisation restante a_T d'une transition T, avant franchissement possible, est égale à la durée totale déterminée moins la somme des durées de sensibilisation de T déjà écoulées depuis son dernier tir ou depuis sa dernière validation. Ainsi, cette mémoire temporelle continue à agir jusqu'à ce que cette transition soit effectivement tirée et ceci même si celle-ci n'est pas sensibilisée dans le marquage courant. Cette politique est très rarement choisie car elle est très lourde à mettre en œuvre et conduit à un processus stochastique assez complexe à analyser. Cette politique peut être intéressante si l'on souhaite modéliser l'opération d'un réparateur amené à réparer un deuxième équipement. Le concept de cette politique permet au réparateur de ne pas perdre le travail déjà effectué sur le premier équipement.
- **Resampling** : on considère que la mémoire temporelle associée à une transition est remise à zéro de manière systématique après le franchissement d'une transition donnée quelle que soit la transition.
- **Enabling memory** : pour cette règle, la durée totale de sensibilisation mémorisée est remise à zéro après chaque désensibilisation de la transition. C'est un compromis entre les deux politiques précédentes.

Nous supposons que les réseaux de Petri stochastiques fonctionnent suivant cette dernière politique.

2.4.4. Types de réseau de Petri stochastique [DAV89] :

Nous distinguons deux types de réseaux de Petri stochastiques en fonction de leur comportement :

- Les réseaux de Petri stochastiques déduits des réseaux de Petri autonomes : c'est le modèle initialement défini ; les marques sensibilisant les transitions stochastiques ne sont pas réservées ou gelées.
- Les réseaux de Petri à temporisation stochastique : ils sont issus des réseaux de Petri T-temporisés ; lorsqu'une transition est sensibilisée, celle-ci réserve le ou les jetons.

Ces deux types de réseaux de Petri ont le même comportement s'il n'y a pas de conflit effectif. Nous supposons que les réseaux de Petri appartiennent à la première catégorie. En

effet, pour ce type de réseau, le marquage M(t) est un processus stochastique makovien (si les durées sont associées à des lois exponentielles), il est alors possible d'exploiter méthode d'analyse markovienne. En effet, Molloy [MOL81] démontra en 1981 que les réseaux de Petri stochastiques finis sont isomorphes à une chaîne de Markov à temps continu. La succession des marquages lors de l'évolution du réseau peut être représentée à l'aide d'une chaîne de Markov. Ainsi, un problème de représentation et d'analyse quantitative d'un système sous la forme d'un graphe de Markov peut être traité à l'aide d'un réseau de Petri, approche plus naturelle pour modéliser un système, tout en s'appuyant sur une analyse markovienne classique pour évaluer les indicateurs de sûreté de fonctionnement.
Dans le cas plus général où le réseau de Petri intègre des distributions quelconques (réseaux de Petri stochastiques étendus, stochastiques et déterministes, stochastiques markoviens régénératifs), une résolution analytique exacte n'est plus possible, et nécessite d'utiliser des techniques de simulation de Monte-Carlo. Une autre possibilité consiste à approcher, lorsque cela est possible, certaines lois par des combinaisons d'exponentielles. Par exemple les lois d'Erlang, Cox, ou des temporisations déterministes peuvent être approximées par plusieurs transitions exponentielles. L'inconvénient majeur est la diminution de la précision des grandeurs recherchées, mais surtout la multiplication du nombre d'états introduits dans le modèle, sujette à une explosion combinatoire.

2.4.5. Temps de séjour moyen

L'analyse d'un réseau de Petri markovien passe par l'étude d'une chaîne de Markov continue. Molloy et Natkin ont montré que le temps de séjour dans chaque marquage suit une distribution suivant une loi exponentielle. Soit τ_M la variable aléatoire associée au temps de séjour dans le marquage M. Soit T(M) l'ensemble des transitions sensibilisées pour ce marquage. D'après [NAT85], le temps de séjour moyen passé dans le marquage M est égal à :

$$E(\tau_M) = \frac{1}{\sum_{j/T_j \in T(M)} \lambda_j(M)}$$

On montre également que pour toute transition T_i sensibilisée dans le marquage M, la probabilité de franchissement de cette transition est égale à :

$$p_{i,M} = \frac{\lambda_i(M)}{\sum_{j/T_j \in T(M)} \lambda_j(M)}$$

2.4.6. Les réseaux de Petri stochastiques généralisés

Dans les réseaux de Petri stochastiques, toutes les transitions sont associées à une temporisation selon une distribution de loi exponentielle. Les réseaux de Petri stochastiques généralisées ont été introduits par Ajmone [AJM95] afin de limiter certaines restrictions.

Cette classe de réseau autorise deux types de transitions :
- Les transitions temporisées basées sur des distributions exponentielles,

- Les transitions déterministes à temporisation nulle (transition immédiate), basées sur une distribution de dirac. Ces transitions sont franchies immédiatement dès lors qu'elles sont sensibilisées.

Les transitions immédiates permettent notamment de modéliser les synchronisations, ou bien encore d'approximer des temporisations très faibles par rapport aux transitions stochastiques. Remarquons que ces distributions immédiates peuvent être assimilées à des distributions exponentielles de taux infini, que l'on note parfois sur les transitions. Graphiquement, les transitions immédiates sont généralement représentées par un trait noir, et les transitions temporisées par un rectangle blanc.

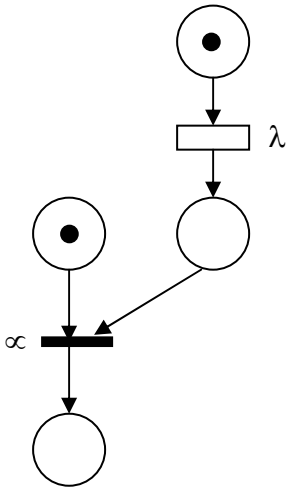


Figure 90 : Exemple de réseau de Petri stochastique généralisé

L'existence de transitions franchies immédiatement a pour conséquence de faire coexister deux types d'états :

- **les états tangibles** : les transitions sensibilisées sont des transitions temporisées. Le temps de séjour moyen est donné par la relation du paragraphe 2.4.5.

- **les états virtuels** ou instantanés: il existe au moins une transition sensibilisée qui est immédiate. Celle-ci est alors franchie immédiatement. Le temps de séjour dans cet état est donc nul.

Il a également été montré que le processus stochastique sous-jacent à un réseau de Petri stochastique généralisé est également une chaîne de Markov continue homogène après élimination des états virtuels [AJM95]. L'étude probabiliste d'un tel réseau se ramène alors à l'étude du processus de marquage lié aux états tangibles. La suppression des états virtuels permet de diminuer la taille, et donc la complexité de la chaîne de Markov analysée.

Transitions immédiates en conflit :

L'existence de transitions immédiates en conflit effectif crée un indéterminisme dans l'évolution du réseau, puisqu'à partir de cet état virtuel, deux transitions sont franchissables simultanément. Il est donc nécessaire d'affecter une distribution discrète de probabilité sur les transitions en conflit.

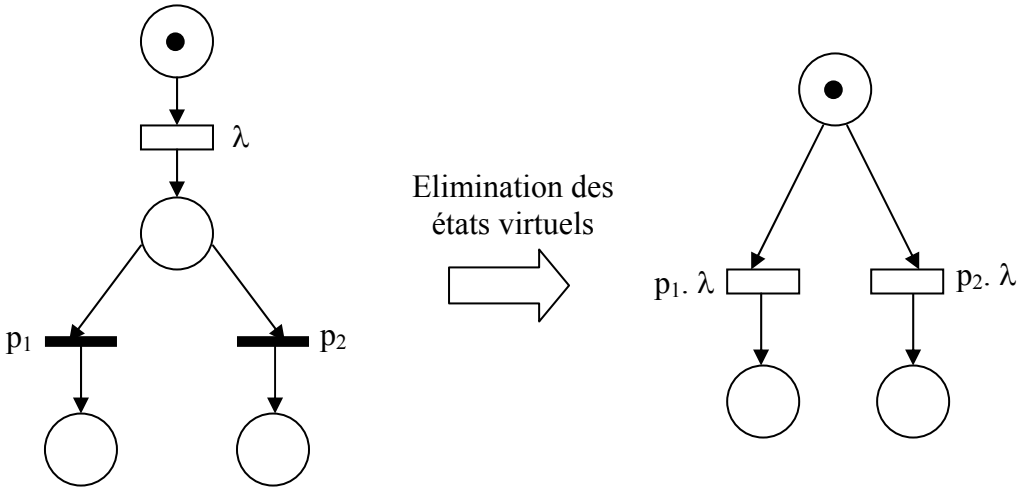


Figure 91 : Transitions immédiates en conflit et élimination d'état virtuel

Sur la figure 91, les probabilités « statiques » p_1 et p_2 sont affectées aux transitions en conflit. On a bien entendu $p_1+p_2=1$. Après élimination de l'état virtuel, le réseau ne comporte plus que deux transitions temporisées en conflit structurel, dont les taux sont pondérés par les probabilités discrètes.

2.4.7. Lien entre les processus stochastiques

L'obtention de la chaîne de Markov sous-jacente au réseau de Petri stochastique est décrite dans de nombreux ouvrages ([DAV89], [AMO99]). Celle-ci ne pose pas de problème particulier, et peut être automatisée.

Dans le cas d'un réseau de Petri stochastique généralisé, la première étape consiste à éliminer tous les états virtuels, comme illustrée sur la figure 91. A partir de ce nouveau réseau, on cherche le graphe des marquages relatif aux états tangibles. Le graphe de Markov possède alors la même structure que le graphe des marquages. Il ne reste plus qu'à étiqueter les arcs par les taux de franchissement dépendants des taux de transition du réseau de Petri et du marquage.

Soit M le marquage courant et $\lambda_i(M)$ le taux de franchissement du graphe de Markov associé au franchissement de la transition T_i. Si cette transition est n-validée pour le marquage M, alors le taux de franchissement $\lambda_i(M)$ vaut :

$$\lambda_i(M)=n.\lambda_i$$

Ainsi, l'exemple de la figure 84 aboutit au graphe d'états suivant :

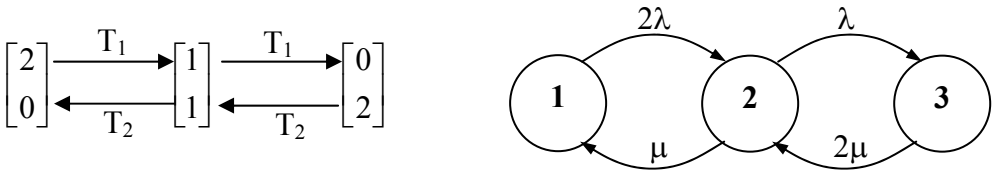


Figure 92 : Graphe des marquages et graphe de Markov associé

Il existe de nombreuses autres classes de réseaux de Petri stochastiques [AMO99] :

- **Les réseaux de Petri stochastiques étendus**, où les temporisations sont distribuées selon des lois quelconques,
- **Les réseaux de Petri stochastiques et déterministes**, composés de transitions à temporisations nulles, distribuées exponentiellement, et déterministes (constantes)
- **Les réseaux de Petri stochastiques régénérateurs markoviens**, identiques à la classe précédente, mais acceptant des transitions à temporisation distribuée selon une loi quelconque.

Cette liste n'est pas exhaustive.

2.5. Les réseaux de Petri continus

C'est un modèle dans lequel le nombre de jetons dans les places est un nombre réel. Aux transitions sont alors associées des « vitesses de transition ». Ce type de modélisation permet de représenter des phénomènes hybrides (continu/discret), par exemple des écoulements et mélanges de fluides… Ce formalisme est principalement destiné à l'évaluation de performance.

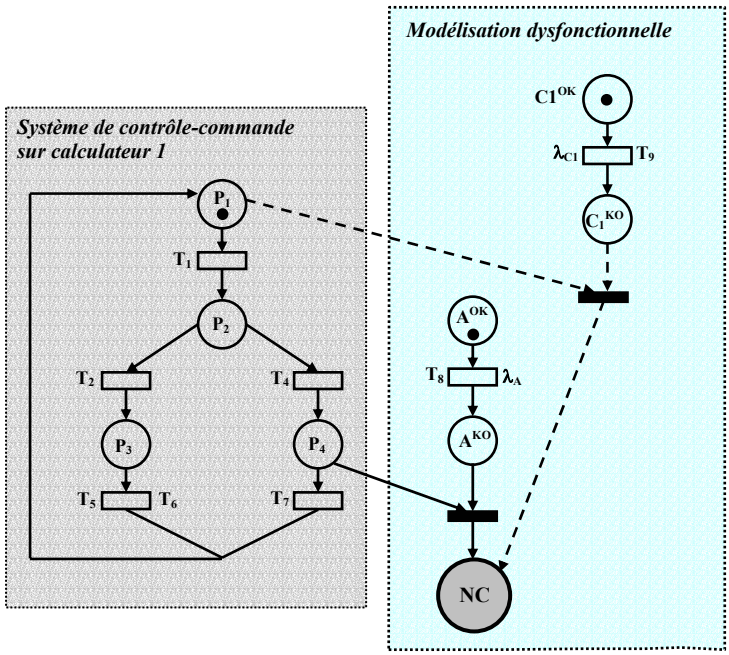
2.6. Les réseaux de Petri colorés

Les réseaux de Petri colorés facilitent la modélisation de systèmes de grande taille. Notamment lorsque certaines parties du système ont des fonctionnements similaires.
Le principe consiste à représenter l'information par les ensembles place/marque. Aux marques de chaque place sont associées un identificateur ou une « couleur ». Le franchissement de ces marques peut être effectué de plusieurs manières, en fonction des couleurs associées aux transitions. La relation entre les couleurs de franchissement et le marquage coloré est définie par des fonctions associées aux arcs.

Annexe 3 : Exemple du système de contrôle-commande

1. Description des architectures

1.1. Modèle 1 : aucune redondance fonctionnelle et matérielle



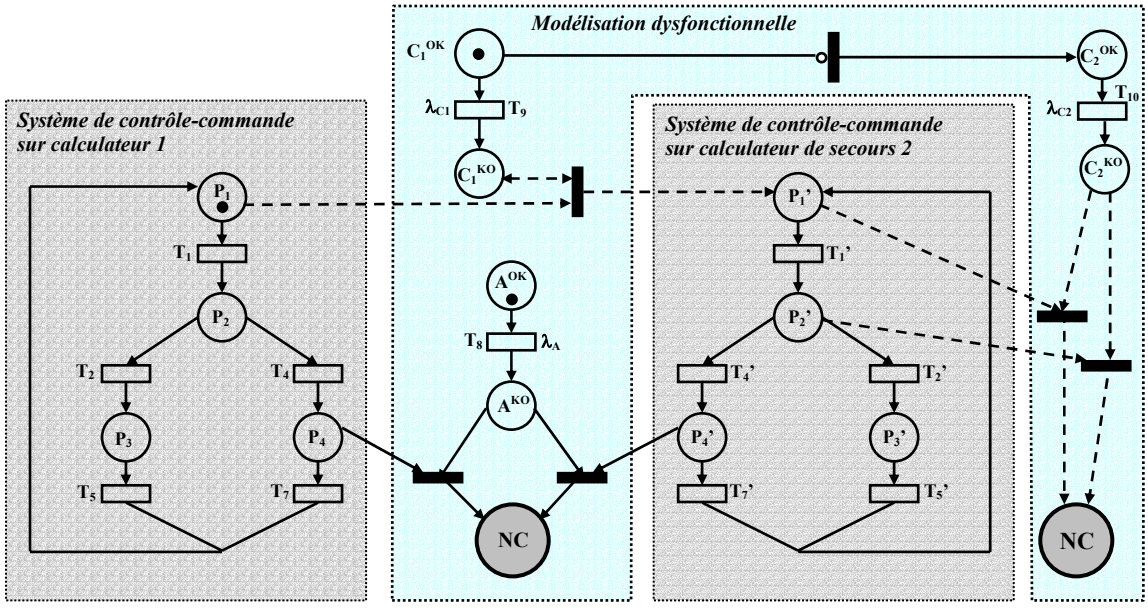
- Le réseau de Petri composé des places P_1, P_2, P_3 et P_4 modélise l'architecture fonctionnelle du système de contrôle-commande.
- Place A^{OK} et A^{KO} modélisent l'état du capteur A respectivement en état de fonctionnement, et en panne.
- Le franchissement de T_8 traduit l'occurrence d'une défaillance du capteur régie par une distribution exponentielle de taux λ_A.

- C_1^{OK} et C_1^{KO} modélisent les états de fonctionnement et de panne du calculateur C_1. Le franchissement de T_9 fait passer de calculateur à l'état de panne. Le taux de défaillance est λ_{C1}.
- La place NC (Non-Conformité) correspond à l'état dysfonctionnement du système.

Remarque :

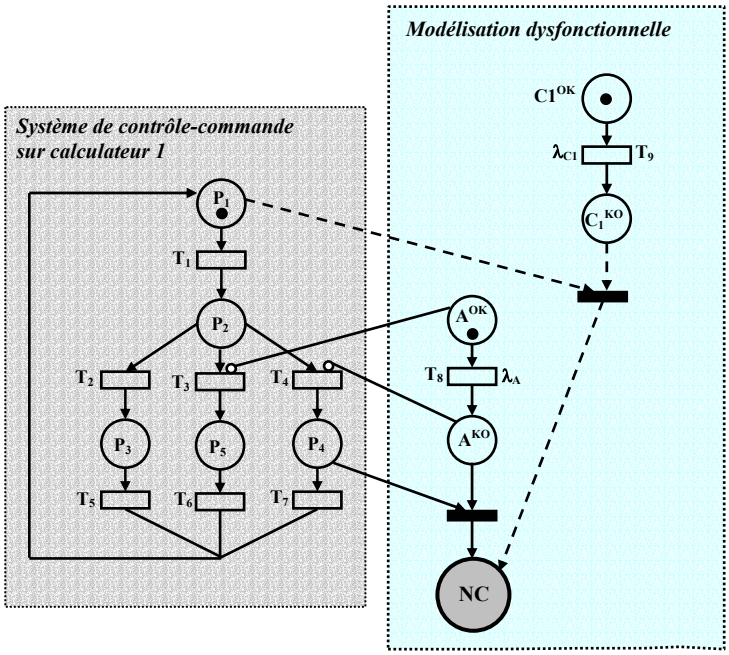
- Sur une défaillance du calculateur C_1, la marque du réseau de Petri relatif au système de contrôle-commande est récupérée afin de bloquer son évolution, et un jeton est mis dans la place NC (Non-conformité) pour traduire l'occurrence d'un état de panne du système. Les arcs en pointillés permettent de retirer la marque du réseau. Quelle que soit l'opération traitée par le système, le jeton doit être retiré. Pour des raisons de clarté, les arcs en pointillés n'ont été représentés qu'à partir de la place P_1. Le mécanisme de récupération doit être identique pour toutes les places.
- Sur défaillance du capteur A, le système arrive en état de Non-Conformité seulement si le système effectue une opération de lecture de ce capteur, associée à la place P_4.

1.2. Modèle 2 : Redondance matérielle du calculateur C_2 :



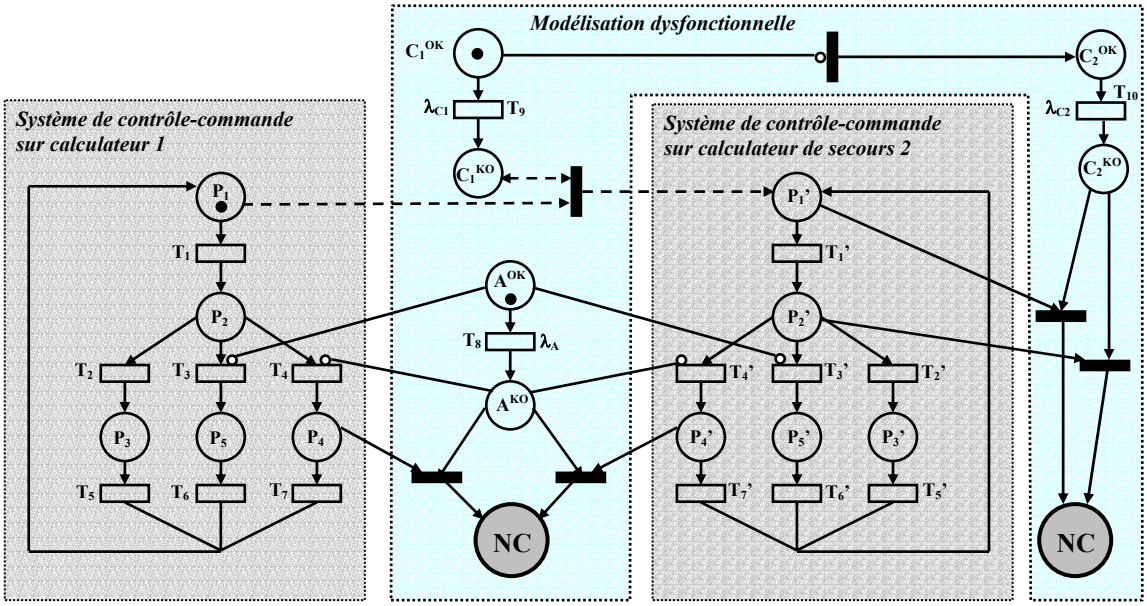
Les places C_2^{OK} et C_2^{KO} représentent les états de fonctionnement et de panne du calculateur de secours C_2. L'architecture fonctionnelle du système est répliquée sur ce calculateur de secours. L'architecture matérielle supporte une défaillance de C_1. Sur occurrence d'une défaillance du calculateur principal, le calculateur de secours, en redondance passive, est activé et reprend l'opération en cours, grâce à une mémorisation de l'état (poursuite du fonctionnement). On modélise cette reconfiguration en retirant le jeton de la place P_i de la partie commande et la place P_i' associée à la fonction sur C_2 est marquée. Tous les arcs et transitions ne sont pas représentés.
Une défaillance de C_2 entraîne le passage en Non-Conformité, à l'instar du modèle 1.

1.3. Modèle 3 : Reconfiguration logicielle supportant une défaillance du capteur A



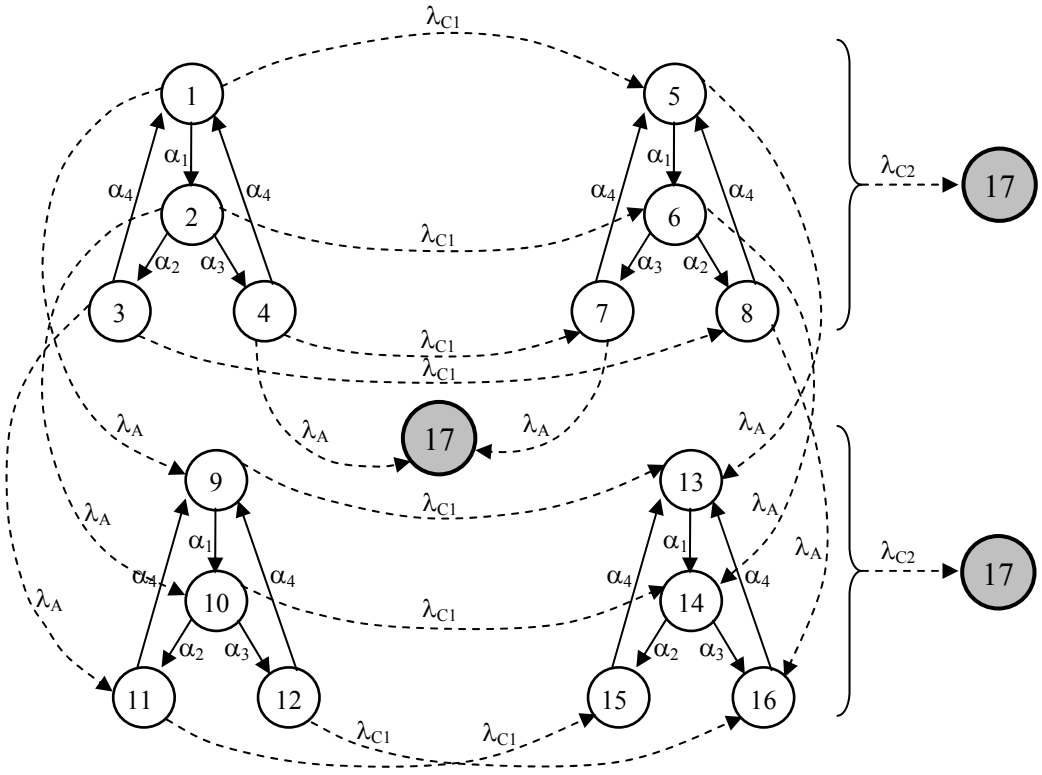
Sur ce modèle est étudiée l'implémentation d'une reconfiguration logicielle en cas de défaillance du capteur A. Sur l'architecture fonctionnelle, on associe à la place P_4 une opération de lecture et d'exploitation de la mesure du capteur. En cas de défaillance (franchissement de T_8), la transition T_4 est inhibée grâce à l'arc inhibiteur, et la place P_4 ne peut plus être marquée. Le mode dégradé est représenté par la branche $\{T_3, P_5, T_6\}$. Toutefois, si la défaillance de A a lieu pendant que l'opération de lecture de A est en cours (place P_4 marquée), alors un dysfonctionnement du système a lieu et la place NC est marquée.

1.4. Modèle 4 : Reconfiguration logicielle et redondance matérielle



Ce modèle permet d'étudier à la fois l'intégration d'une reconfiguration logicielle de l'algorithme de contrôle-commande et une redondance matérielle (association des modèles 2 et 3).

2. Graphe de Markov du modèle 4



Les états du graphe de Markov correspondent aux marquages suivants :

- état 1 ≡ $\{P_1, C_1^{OK}, A^{OK}\}$
- état 2 ≡ $\{P_2, C_1^{OK}, A^{OK}\}$
- état 3 ≡ $\{P_3, C_1^{OK}, A^{OK}\}$
- état 4 ≡ $\{P_4, C_1^{OK}, A^{OK}\}$
- état 5 ≡ $\{P_1', C_2^{OK}, A^{OK}\}$
- état 6 ≡ $\{P_2', C_2^{OK}, A^{OK}\}$
- état 7 ≡ $\{P_3', C_2^{OK}, A^{OK}\}$
- état 8 ≡ $\{P_4', C_2^{OK}, A^{OK}\}$
- état 9 ≡ $\{P_1, C_1^{OK}, A^{KO}\}$
- état 10 ≡ $\{P_2, C_1^{OK}, A^{KO}\}$
- état 11 ≡ $\{P_3, C_1^{OK}, A^{KO}\}$
- état 12 ≡ $\{P_5, C_1^{OK}, A^{KO}\}$
- état 13 ≡ $\{P_1', C_2^{OK}, A^{KO}\}$
- état 14 ≡ $\{P_2', C_2^{OK}, A^{KO}\}$
- état 15 ≡ $\{P_3', C_2^{OK}, A^{KO}\}$
- état 16 ≡ $\{P_5', C_2^{OK}, A^{KO}\}$
- état 17 ≡ {NC}

Taux de franchissement stochastiques associés aux transitions de l'architecture fonctionnelle :

$T_1, T_1' \rightarrow \alpha_1$
$T_2, T_2' \rightarrow \alpha_2$
$T_3, T_3', T_4, T_4' \rightarrow \alpha_3$
$T_5, T_5', T_6, T_6', T_7, T_7' \rightarrow \alpha_4$

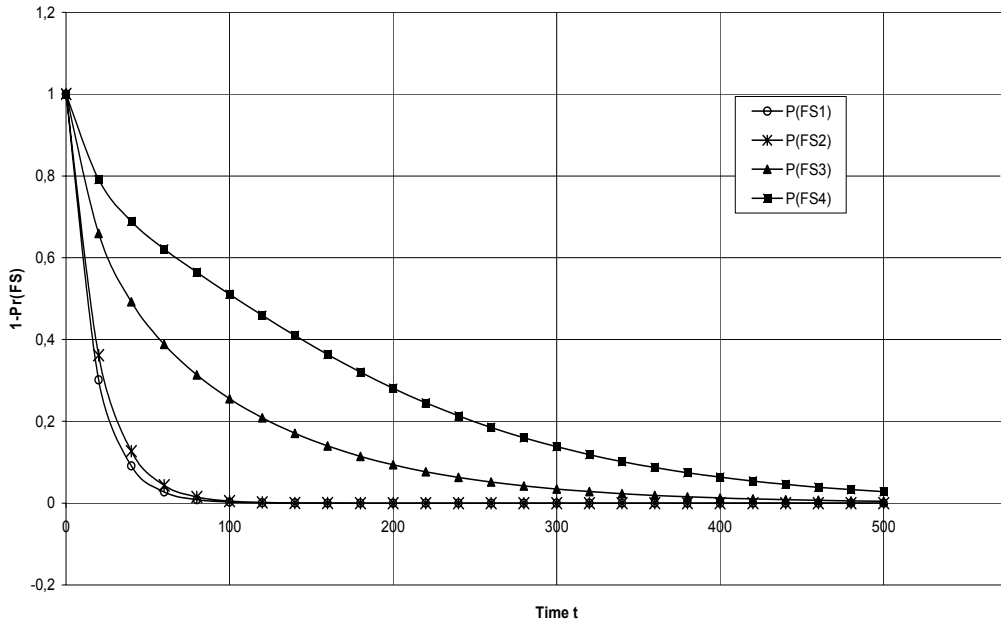
3. Matrice de transition du modèle 4

$-(\alpha_1+\lambda_{C1}+\lambda_A)$	α_1	0	0	λ_{C1}	0	0	0	λ_A	0	0	0	0	0	0	0	0
0	$-(\alpha_2+\alpha_3+\lambda_{C1}+\lambda_A)$	α_2	α_3	0	λ_{C1}	0	0	0	λ_A	0	0	0	0	0	0	0
α_4	0	$-(\alpha_4+\lambda_{C1}+\lambda_A)$	0	0	0	0	λ_{C1}	0	0	λ_A	0	0	0	0	0	0
α_4	0	0	$-(\alpha_4+\lambda_{C1}+\lambda_A)$	0	0	λ_{C1}	0	0	0	0	0	0	0	0	0	λ_A
0	0	0	0	$-(\alpha_1+\lambda_{C2}+\lambda_A)$	α_1	0	0	0	0	0	0	λ_A	0	0	0	λ_{C2}
0	0	0	0	0	$-(\alpha_2+\alpha_3+\lambda_{C1}+\lambda_A)$	α_3	α_2	0	0	0	0	0	λ_A	0	0	λ_{C2}
0	0	0	0	α_4	0	$-(\alpha_4+\lambda_{C2}+\lambda_A)$	0	0	0	0	0	0	0	0	0	$\lambda_A+\lambda_{C2}$
0	0	0	0	α_4	0	0	$-(\alpha_4+\lambda_{C2}+\lambda_A)$	0	0	0	0	0	0	0	λ_A	λ_{C2}
0	0	0	0	0	0	0	0	$-(\alpha_1+\lambda_{C1})$	α_1	0	0	λ_{C1}	0	0	0	0
0	0	0	0	0	0	0	0	0	$-(\alpha_2+\alpha_3+\lambda_{C1})$	α_2	α_3	0	λ_{C1}	0	0	0
0	0	0	0	0	0	0	0	α_4	0	$-(\alpha_4+\lambda_{C1})$	0	0	0	λ_{C1}	0	0
0	0	0	0	0	0	0	0	α_4	0	0	$-(\alpha_4+\lambda_{C1})$	0	0	0	λ_{C1}	0
0	0	0	0	0	0	0	0	0	0	0	0	$-(\alpha_1+\lambda_{C2})$	α_1	0	0	λ_{C2}
0	0	0	0	0	0	0	0	0	0	0	0	0	$-(\alpha_2+\alpha_3+\lambda_{C2})$	α_2	α_3	λ_{C2}
0	0	0	0	0	0	0	0	0	0	0	0	α_4	0	$-(\alpha_4+\lambda_{C2})$	0	λ_{C2}
0	0	0	0	0	0	0	0	0	0	0	0	α_4	0	0	$-(\alpha_4+\lambda_{C2})$	λ_{C2}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Mode Nominal (aucune défaillance) — *Reconf. matérielle (défaillance de C_1)* — *Reconf. logicielle (défaillance de A)* — *Reconf. matérielle + logicielle* — *NC*

4. Comparaison des courbes de fiabilité pour les quatre modèles

Les taux de franchissement sont choisis avec une valeur arbitraire afin d'illustrer la démarche. On aboutit alors aux courbes de Fiabilité des Systèmes FSi(t) suivantes :



Il apparaît clairement que le modèle le plus fiable, c'est-à-dire évoluant le plus lentement vers 0 est le modèle 4, puisqu'il supporte le plus de défaillances de composants.
En outre, ces courbes montrent également que la redondance matérielle du calculateur est plus efficace que la redondance fonctionnelle, avec les taux de défaillance choisis.

Cette analyse quantitative doit ensuite aider le concepteur dans son choix d'architecture fonctionnelle et matérielle, tout en mesurant le compromis entre le coût engendré par la solution retenue, et le gain en fiabilité.

5. Aide à l'identification des taux de défaillance équivalent

Composant défaillant/ Mode de défaillance	**Taux de défaillance**	**Macro-état amont / mode de fonctionnement**	**Macro-état aval**	**Composante fortement connexe impliquée**	**Etat élémentaire fonctionnel de la composante**	**Terme de pondération**
Calculateur principal C1	λ_{C1}	Mode nominal (FN)	Reconfiguration matérielle sur C2 (MD1)	Système de contrôle-commande $\{P_1, P_2, P_3, P_4\}$	Toutes les places	= 1
Capteur A	λ_A	Mode nominal (No Fail)	Non Conformité (NC)	Système de contrôle-commande $\{P_1, P_2, P_3, P_4\}$	Place P_4	p_4

Capteur A	λ_A	Mode nominal (No Fail)	Reconfiguration logicielle (MD2)	Système de contrôle-commande $\{P_1, P_2, P_3, P_4\}$	Toutes les places sauf P_4	$p_4'=1-p_4$
Calculateur principal C1	λ_{C1}	Reconfiguration logicielle (MD2)	Reconfiguration logicielle et matérielle sur C2 (MD3)	Système de contrôle-commande $\{P_1, P_2, P_3, P_5\}$	Toutes les places	= 1
Capteur A	λ_A	Reconfiguration matérielle sur C2 (MD1)	Reconfiguration logicielle et matérielle sur C2 (MD3)	Système de contrôle-commande sur C2 $\{P_1', P_2', P_3', P_4'\}$	Toutes les places sauf P_4'	$p_4'=1-p_4$
Capteur A	λ_A	Reconfiguration matérielle sur C2 (MD1)	Non Conformité (NC)	Système de contrôle-commande sur C2 $\{P_1', P_2', P_3', P_4'\}$	Place P_4'	p_4
Calculateur de secours C2	λ_{C2}	Reconfiguration matérielle sur C2 (MD1)	Non Conformité (NC)	Système de contrôle-commande sur C2 $\{P_1', P_2', P_3', P_4'\}$	Toutes les places	= 1
Calculateur de secours C2	λ_{C2}	Reconfiguration logicielle et matérielle sur C2 (MD3)	Non Conformité (NC)	Système de contrôle-commande sur C2 $\{P_1', P_2', P_3', P_5'\}$	Toutes les places	= 1

Annexe 4 : Données du cas test « Réservoir »

1. Modélisation en réseau de Petri du système « réservoir »

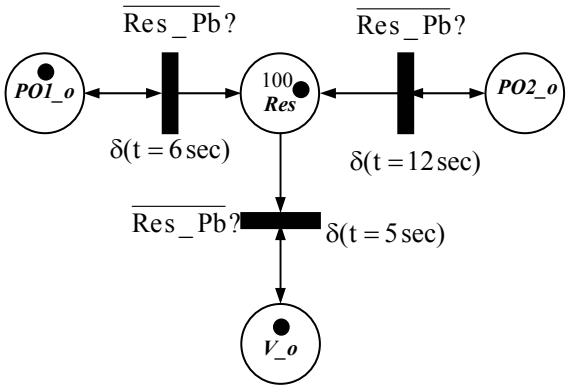


Figure 93 : Evolution du niveau h dans le réservoir

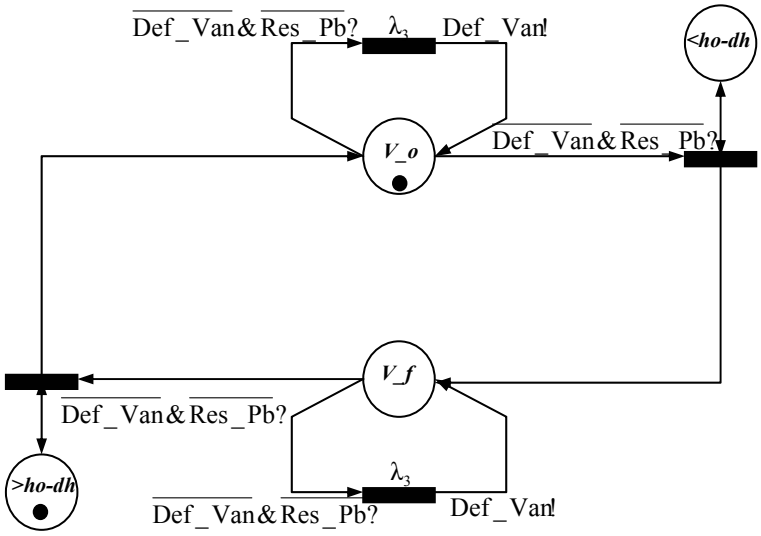


Figure 94 : Etats de la vanne V

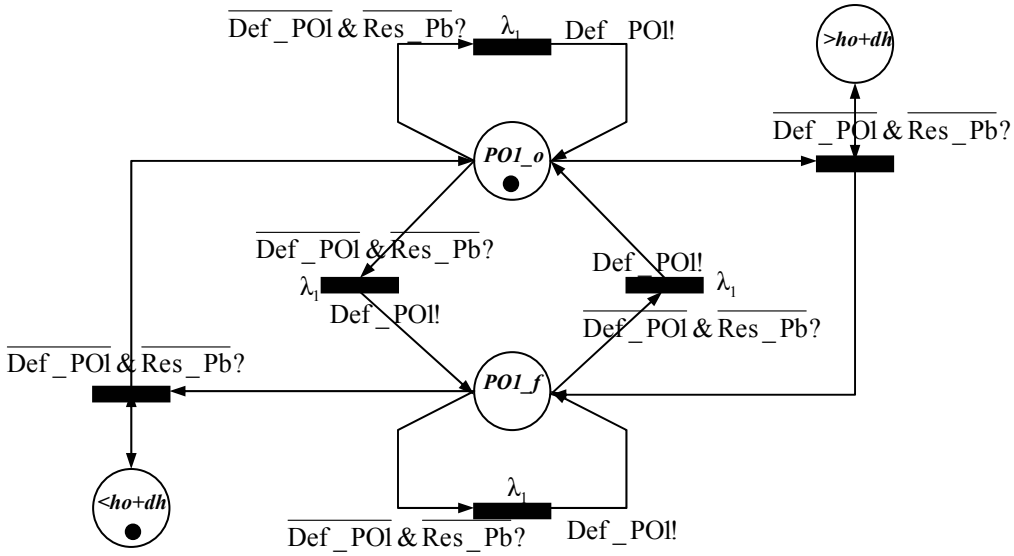


Figure 95 : Etats de la pompe PO1

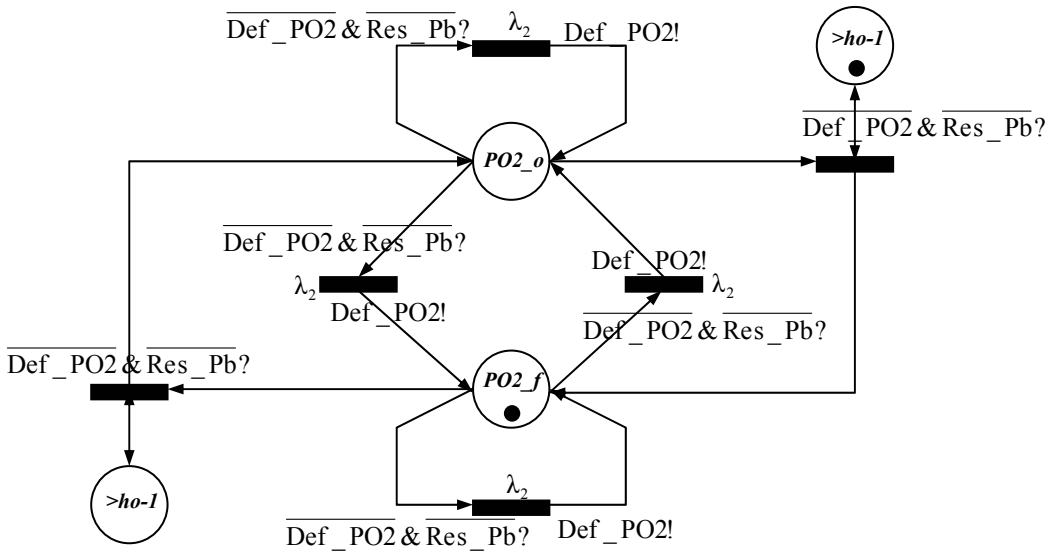


Figure 96 : Etats de la pompe PO2

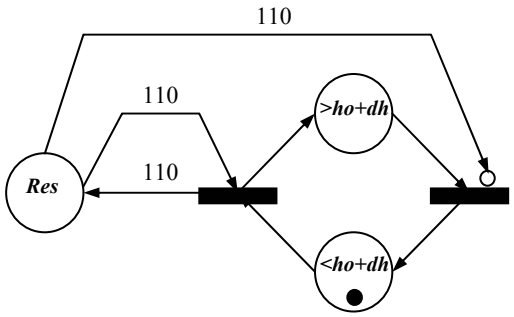


Figure 97 : Détection du niveau haut

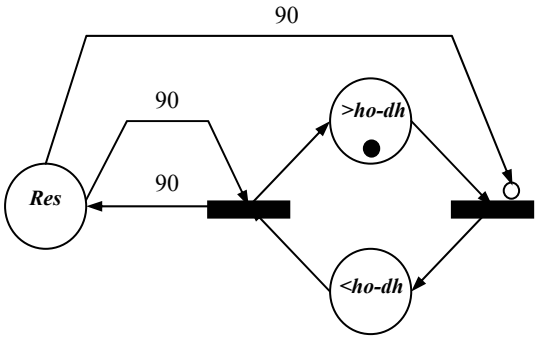


Figure 98 : Détection du niveau bas

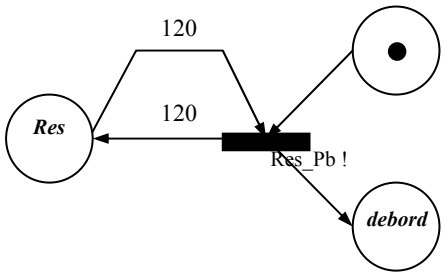


Figure 99 : Détection du débordement

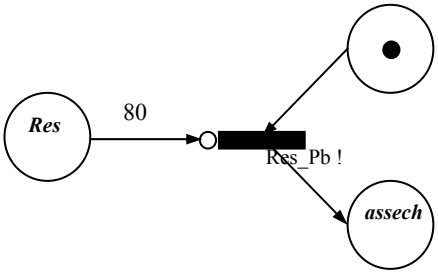


Figure 100 : Détection de l'assèchement

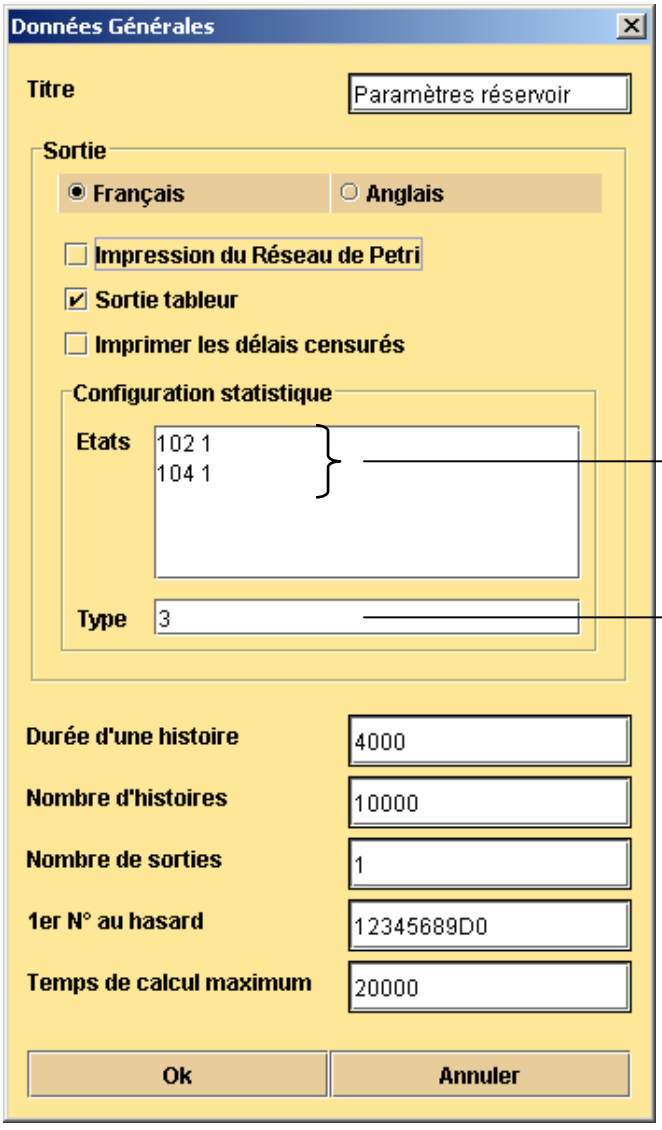
Description des places :

Place	Description
Res	Niveau dans le réservoir
PO1_o	Pompe PO1 ouverte
PO1_f	Pompe PO1 fermée
PO2_o	Pompe PO2 ouverte
PO2_f	Pompe PO2 fermée
V_o	Vanne ouverte
V_f	Vanne fermée
<h0-dh	Niveau h inférieur au seuil bas
>h0-dh	Niveau h supérieur au seuil bas
<h0+dh	Niveau inférieur au seuil haut
>h0+dh	Niveau supérieur au seuil haut
debord	Débordement du réservoir
assech	Assèchement du réservoir

Description des messages :

Message (booléen)	Etat du système	Valeur initiale
Res_Pb	Assèchement ou débordement	FAUX
Def_PO1	Défaillance de la pompe PO1	FAUX
Def_PO2	Défaillance de la pompe PO2	FAUX
Def_V	Défaillance de la vanne V	FAUX

2. Interface graphique MOCA-RP de saisie des paramètres de la simulation



Etats observés :
<Place_i> <Nb jetons place_i>

Type de statistique :
1 : Temps moyens de séjours cumulés passés dans les états
2 : Moyenne de présence d'états à la fin d'une histoire
3 : Marquage moyen de certaines places
4 : Nombre de passages moyens par les états
5 : Calcul de productivité
6 : Date moyenne de première arrivée dans les états

3. Script Matlab de gestion de la simulation de Monte-Carlo pour le système réservoir

```
%     SIMULATION DE MC   -----    MODELE RESERVOIR A 2 POMPES
% ****************************************************************
%
% 05/01/2004
% Raphaël SCHOENIG
%
% ================================================================
% INITIALISATION DES VARIABLES
% ================================================================

% Débits moyens des pompes et de la vanne :

Q1 = 10;
Q2 = 5;
Q3 = 12;

% Variations parasites sur les débits des pompes

gain_bruit1 = 2;
gain_bruit2 = 1;
gain_bruit3 = 0.5;

% Taux de défaillances des pompes et de la vanne

l1 = 2.2831 * 10^-5;  % PO1
l2 = 2.8571 * 10^-5;  % PO2
l3 = 1.5625 * 10^-5;  % V

% Déclaration des variables « Date d'occurrence des pannes
% (modes de défaillance)

global t1_ouv t1_fer t1_blo t2_ouv t2_fer t2_blo t3_blo


% Initialisation des variables nécessaires pour la simulation de MONTE-CARLO

N = 20000 ;             % Nombre d'histoires
T = 150000 ;           % Durée d'une histoire
Pas_tps = 1;            % pas de temps


% NER = nombre total d'événements redoutés détectés
% NERi() = nombre d'erreurs détectées entre le pas de temps i et i+1;
% Nbpas = découpage de T en NBpas : NBpas=T/pasSimul;

NER_ass=0;                    % Nombre d'ER assèchement
NER_deb=0;                    % Nombre d'ER débordement
NBpas=floor(T/ Pas_tps);


for i=1:NBpas+1
    NERi_ass(i)=0;
    NERi_deb(i)=0;
end

DebutSimul=clock;             % Mémorisation du début de la simul (pour
                              %L'évaluation des performances de temps de calcul)
```

```
% ================================================================
% Simulation des N histoires
% ================================================================

for hist = 1:N

    % Défaillance aléatoire de taux lambda: temps tfail d'activation

        % Astuce pour « accélérer » la simulation des N histoires :
        % Si après le calcul des dates d'occurrence des pannes, aucune
        % d'entre elles ne se situent dans l'intervalle [0, T], alors
        % on incrémente « hist » de 1 puisqu'il n'y aura pas d'ER pour
        % cette histoire. On passe à l'histoire suivante, et on recalcule
        % les dates.



        detect=0;      % Flag

        while detect==0,

        % pour l'histoire N° « hist » les dates des défaillances sont tirées
        % aléatoirement :

        t1_ouv = -log(1-rand)/l1;       % Pompe 1
        t1_fer = -log(1-rand)/l1;
        t1_blo = -log(1-rand)/l1;

        t2_ouv = -log(1-rand)/l2;       % Pompe 2
        t2_fer = -log(1-rand)/l2;
        t2_blo = -log(1-rand)/l2;

        t3_blo = -log(1-rand)/l3;       % Vanne

        if (t1_ouv>T & t1_fer>T & t1_blo>T & t2_ouv>T & t2_fer>T & t2_blo>T &
                                                                t3_blo>T),
            detect=0;          % Aucune date de défaillance n'est dans [0 T]
            hist=hist+1;       % On passe à l'histoire suivante
        else
            detect=1;          % Au moins une défaillance se produira dans [0 T]
        end

        end



        tER_ass=0;             % Date d'occurrence de l'ER assèchement
        tER_deb=0;             % Date d'occurrence de l'ER débordement

        sim('Reservoir_trig',[0 T]);         % Simulation du modèle SK / SF


        if tER_ass(end) > 0
               NER_ass=NER_ass+1;            % ER assèchement observé
               NERi_ass(floor(tER_ass(end)/ Pas_tps)+1) =
                                NERi_ass(floor(tER_ass(end)/ Pas_tps)+1)+1;
        end


        if tER_deb(end) > 0
               NER_deb=NER_deb+1;
               NERi_deb(floor(tER_deb(end)/ Pas_tps)+1) =
                                     NERi_deb(floor(tER_deb(end)/ Pas_tps)+1)+1;
        end
```

```
        % On sauvegarde les résultats toutes les 1000 histoires dans
        % des fichiers intermédiaires.

        if (hist/1000)==floor(hist/1000),
         FinSimul = clock             % Date de la fin du millier d'histoire
         fichier=[,fichier_' num2str(hist)]
         disp(hist)
         save (fichier,'T','hist','NERi_ass','NERi_deb','DebutSimul','FinSimul');
        end


end

% ================================================================
% FIN DE LA SIMULATION DES N HISTOIRES
% ================================================================


% Variable « temps »
x=[0:pasSimul:(T- Pas_tps)];


% A partir du nombre de pannes détectées à chaque pas de temps,
% on peut en déduire les fonctions de répartition, c'est-à-dire
% les probabilités d'apparition des ER à l'instant x(i).
% Ces variables sont nommées y_ass et y_deb :

for i=1:NBpas
    y_ass(i)=sum(NERi_ass(1:i))/N;
    y_deb(i)=sum(NERi_deb(1:i))/N;
end
```

4. Evaluation des coefficients de pondération par simulation

Coefficient	Macro-état en amont	N° Macro-Etat	Macro-état en aval	N° Macro-Etat	Indice Taux λi	Valeur
p_1	No_Fail	1	P1	2	1	**2**
p_2	P1	2	P1, $\overline{P2}$	8	2	**1,8836**
p_3	P1	2	P1,V	9	3	**0,8836**
p_4	P1	2	P1,P2	17	2	**1,1164**
p_5	P1	2	P1, $\overline{V}$	18	3	**0,1164**
p_6	No_Fail	1	$\overline{P1}$	3	1	**1**
p_7	$\overline{P1}$	3	$\overline{P1}$,P2	10	2	**1,7063**
p_8	$\overline{P1}$	3	$\overline{P1}$, $\overline{P2}$	11	2	**1,2937**
p_9	$\overline{P1}$	3	$\overline{P1}$,V	15	3	**0,2937**
p_{10}	$\overline{P1}$	3	$\overline{P1}$, $\overline{V}$	16	3	**0,7063**
p_{11}	No_Fail	1	P2	4	2	**1,1164**
p_{12}	P2	4	$\overline{P1}$,P2	10	1	**1,2997**
p_{13}	P2	4	P1,P2	17	1	**1,7**

$\mathbf{p_{14}}$	P2	4	P2, $\overline{V}$	19	3	**0**
$\mathbf{p_{15}}$	P2	4	P2,V	12	3	**1**
$\mathbf{p_{16}}$	No_Fail	1	$\overline{P2}$	5	2	**1,8836**
$\mathbf{p_{17}}$	$\overline{P2}$	5	P1, $\overline{P2}$	8	1	**2**
$\mathbf{p_{18}}$	$\overline{P2}$	5	$\overline{P1}$, $\overline{P2}$	11	1	**1**
$\mathbf{p_{19}}$	$\overline{P2}$	5	$\overline{P2}$,V	13	3	**0,8351**
$\mathbf{p_{20}}$	$\overline{P2}$	5	$\overline{P2}$, $\overline{V}$	14	3	**0,1649**
$\mathbf{p_{21}}$	No_Fail	1	V	6	3	**0,8836**
$\mathbf{p_{22}}$	V	6	P1,V	9	1	**2**
$\mathbf{p_{23}}$	V	6	P2,V	12	2	**1,3967**
$\mathbf{p_{24}}$	V	6	$\overline{P2}$,V	13	2	**1,6033**
$\mathbf{p_{25}}$	V	6	$\overline{P1}$,V	15	1	**1**
$\mathbf{p_{26}}$	No_Fail	1	$\overline{V}$	7	3	**0,1164**
$\mathbf{p_{27}}$	$\overline{V}$	7	P1, $\overline{V}$	18	1	**1**
$\mathbf{p_{28}}$	$\overline{V}$	7	P2, $\overline{V}$	19	2	**1**
$\mathbf{p_{29}}$	$\overline{V}$	7	$\overline{P2}$, $\overline{V}$	14	2	**2**
$\mathbf{p_{30}}$	$\overline{V}$	7	$\overline{P1}$, $\overline{V}$	16	2	**2**
$\mathbf{p_{31}}$	P1, $\overline{P2}$	8	P1, $\overline{P2}$,V	20	3	**0,8351**
$\mathbf{p_{32}}$	P1,V	9	P1, $\overline{P2}$,V	20	2	**1,6033**
$\mathbf{p_{33}}$	$\overline{P1}$,P2	10	$\overline{P1}$,P2,V	21	3	**0,4155**
$\mathbf{p_{34}}$	$\overline{P1}$, $\overline{P2}$	11	$\overline{P1}$, $\overline{P2}$, $\overline{V}$	22	3	**0,991**
$\mathbf{p_{35}}$	P2,V	12	$\overline{P1}$,P2,V	21	1	**1,2996**
$\mathbf{p_{36}}$	$\overline{P2}$,V	13	P1, $\overline{P2}$,V	20	1	**2**
$\mathbf{p_{37}}$	$\overline{P2}$, $\overline{V}$	14	$\overline{P1}$, $\overline{P2}$, $\overline{V}$	22	1	**1,999**
$\mathbf{p_{38}}$	$\overline{P1}$,V	15	$\overline{P1}$,P2,V	21	2	**1,9991**
$\mathbf{p_{39}}$	$\overline{P1}$, $\overline{V}$	16	$\overline{P1}$, $\overline{P2}$, $\overline{V}$	22	2	**2**
$\mathbf{p_{40}}$	$\overline{P1}$, $\overline{P2}$	11	$\overline{P1}$, $\overline{P2}$,V	23	3	**0,0008617**
$\mathbf{p_{41}}$	$\overline{P2}$,V	13	$\overline{P1}$, $\overline{P2}$,V	23	1	**1**
$\mathbf{p_{42}}$	$\overline{P1}$,V	15	$\overline{P1}$, $\overline{P2}$,V	23	2	**1,0009**
$\mathbf{p_{43}}$	P1, $\overline{P2}$	8	P1, $\overline{P2}$, $\overline{V}$	18	3	**0,1649**
$\mathbf{p_{44}}$	P1,V	9	P1,P2,V	17	2	**1,3967**
$\mathbf{p_{45}}$	$\overline{P1}$,P2	10	$\overline{P1}$,P2, $\overline{V}$	19	3	**0,5845**
$\mathbf{p_{47}}$	P2,V	12	P1,P2,V	17	1	**1,7004**
$\mathbf{p_{48}}$	$\overline{P2}$, $\overline{V}$	14	P1, $\overline{P2}$, $\overline{V}$	18	1	**1,000992**
$\mathbf{p_{49}}$	$\overline{P1}$, $\overline{V}$	16	$\overline{P1}$,P2, $\overline{V}$	19	2	**1**

L'indice de la sixième colonne est l'indice du taux de défaillance pondéré par le coefficient correspondant :

- λ_1 = taux de défaillance de la pompe 1
- λ_2 = taux de défaillance de la pompe 2
- λ_3 = taux de défaillance de la vanne

5. Matrice de transition du graphe de Markov réduit (les cases vides représentent une valeur nulle)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1		$p_1\lambda_1$	$p_6\lambda_1$	$p_{11}\lambda_2$	$p_{16}\lambda_2$	$p_{21}\lambda_3$	$p_{26}\lambda_3$																
2								$p_2\lambda_2$	$p_3\lambda_3$								$p_4\lambda_2$	$p_5\lambda_3$					
3										$p_7\lambda_2$	$p_8\lambda_2$				$p_9\lambda_3$	$p_{10}\lambda_3$							
4										$p_{12}\lambda_1$		$p_{15}\lambda_3$					$p_{13}\lambda_1$		$p_{14}\lambda_3$				
5								$p_{17}\lambda_1$			$p_{18}\lambda_1$		$p_{19}\lambda_3$	$p_{20}\lambda_3$									
6									$p_{22}\lambda_1$			$p_{23}\lambda_2$	$p_{24}\lambda_2$		$p_{25}\lambda_1$								
7														$p_{29}\lambda_2$		$p_{30}\lambda_2$		$p_{27}\lambda_1$	$p_{28}\lambda_2$				
8																		$p_{43}\lambda_3$		$p_{31}\lambda_3$			
9																	$p_{44}\lambda_2$			$p_{32}\lambda_2$			
10																			$p_{45}\lambda_3$		$p_{33}\lambda_3$		
11																						$p_{34}\lambda_3$	$p_{40}\lambda_3$
12																	$p_{47}\lambda_1$				$p_{35}\lambda_1$		
13																							
14																		$p_{48}\lambda_1$				$p_{37}\lambda_1$	
15																							
16																			$p_{49}\lambda_2$			$p_{39}\lambda_2$	
17																							
18																							
19																							
20																							
21																							
22																							
23																							

Légende des états absorbants :	**Débordement ER1**	**Assèchement ER2**	**Plus de régulation ER3**

6. Comparaison des résultats entre la méthode analytique et la simulation

6.1. Courbes de probabilité

Les conventions suivantes seront utilisées pour la représentation graphique des probabilités :

---------- Solution par approche markovienne
———————— Solution par simulation de Monte-Carlo du modèle Simulink/Stateflow

Simulation N°1				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Non triggé	3000	10000	λ_1=2,2831.10^{-3} λ_2=2,8571.10^{-3} λ_3=1,5625.10^{-3}	Pas fixe Ode 1 Pas = 0,1

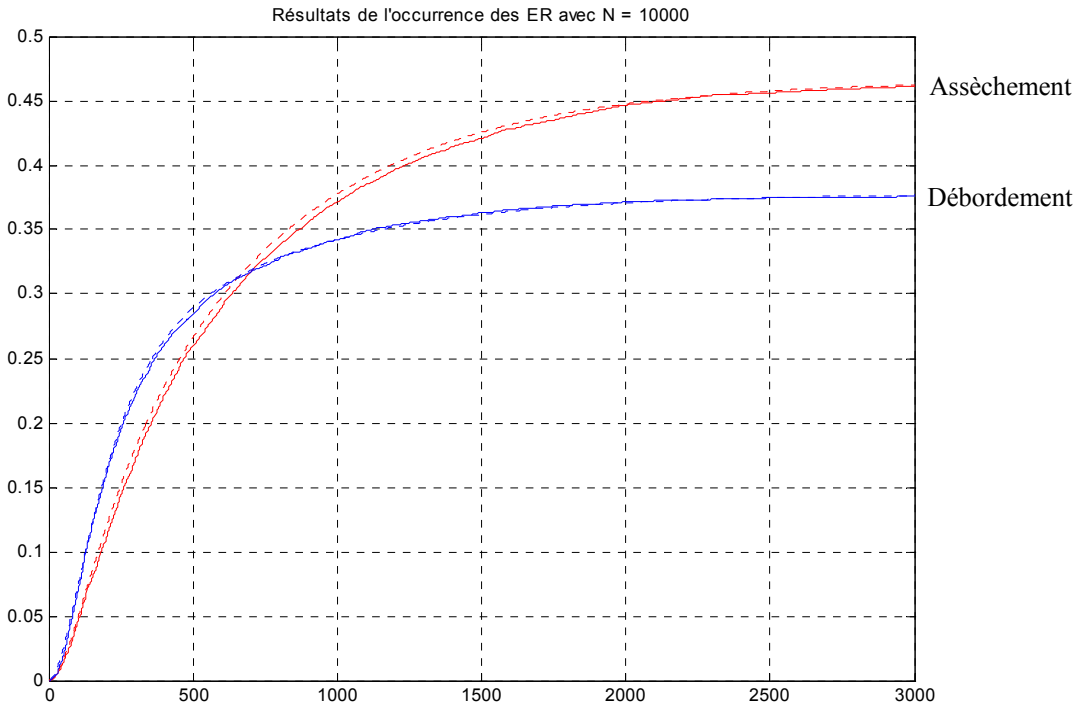


Figure 101 : Courbes obtenues après N=10000 histoires jouées

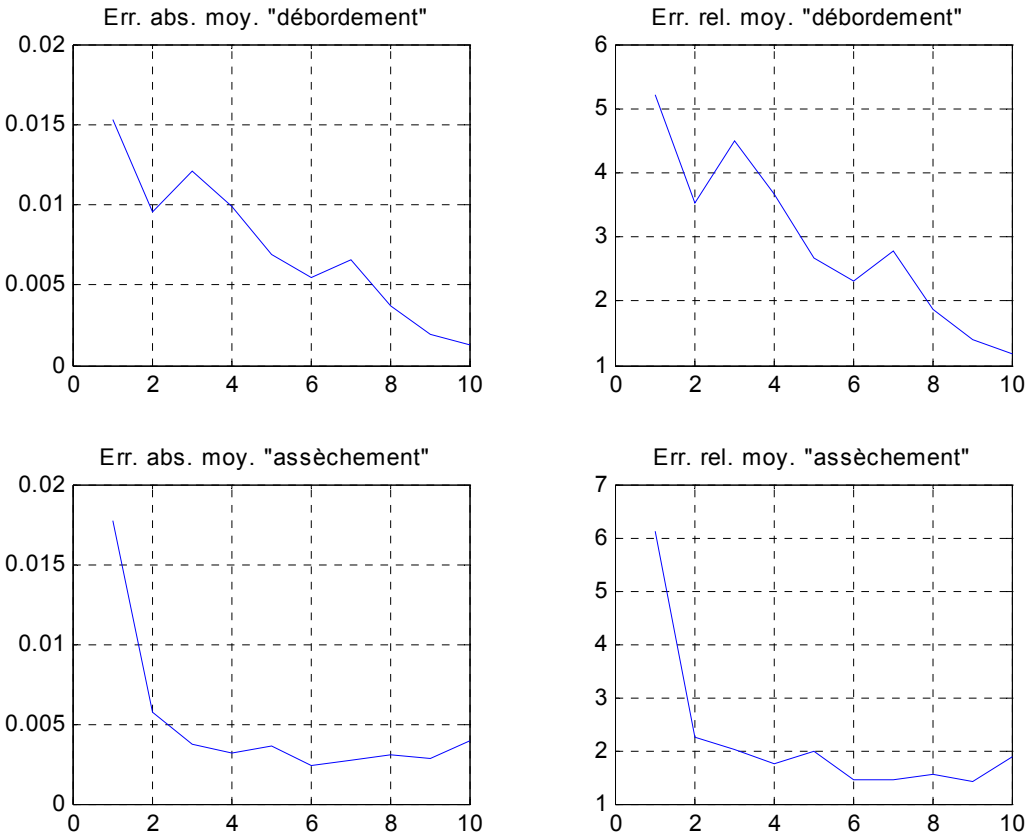


Figure 102 : Evolution des erreurs en fonction de N (x1000)

Simulation N°2				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Non triggé	3000	10000	λ_1=2,2831.10^{-3} λ_2=2,8571.10^{-3} λ_3=1,5625.10^{-3}	Pas variable Ode 45

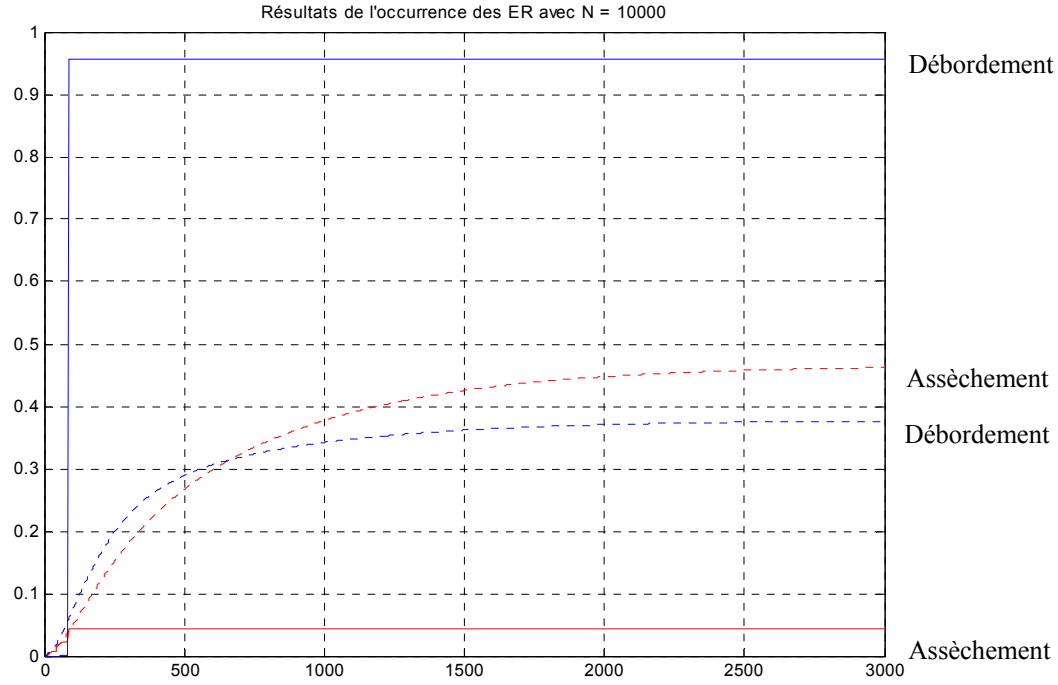


Figure 103 : Courbes obtenues après N=10000 histoires jouées

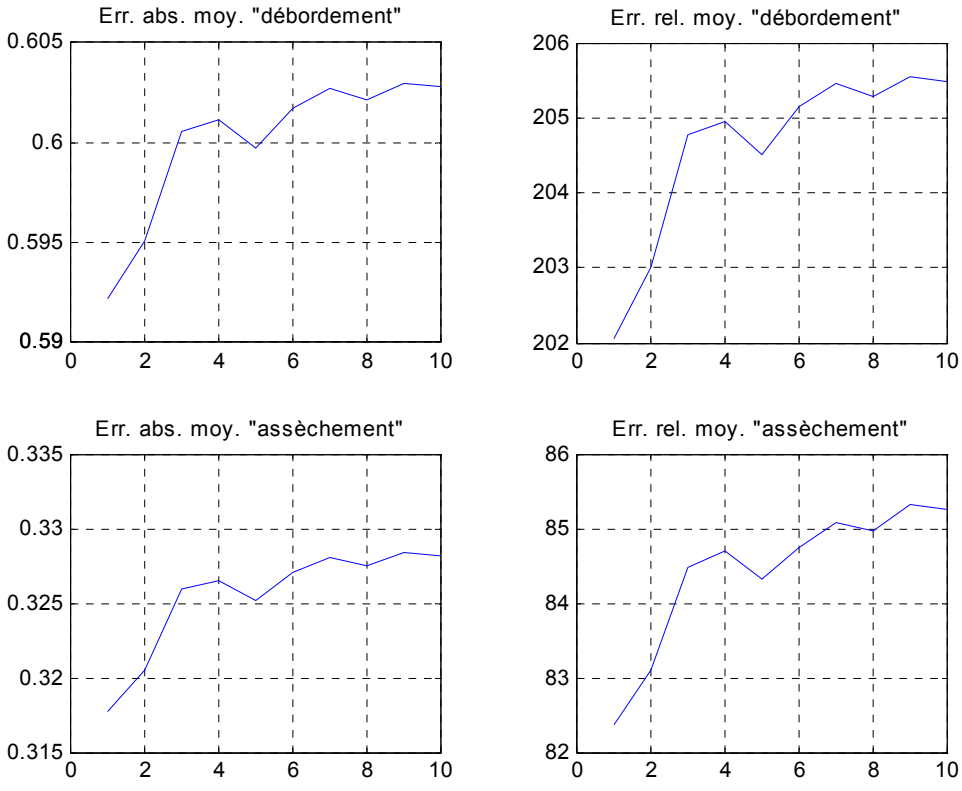


Figure 104 : Evolution des erreurs en fonction de N (x1000)

Simulation N°3				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Non triggé	3000	10000	λ_1=2,2831.10^{-3} λ_2=2,8571.10^{-3} λ_3=1,5625.10^{-3}	Pas fixe Ode 1 Pas = 0,01

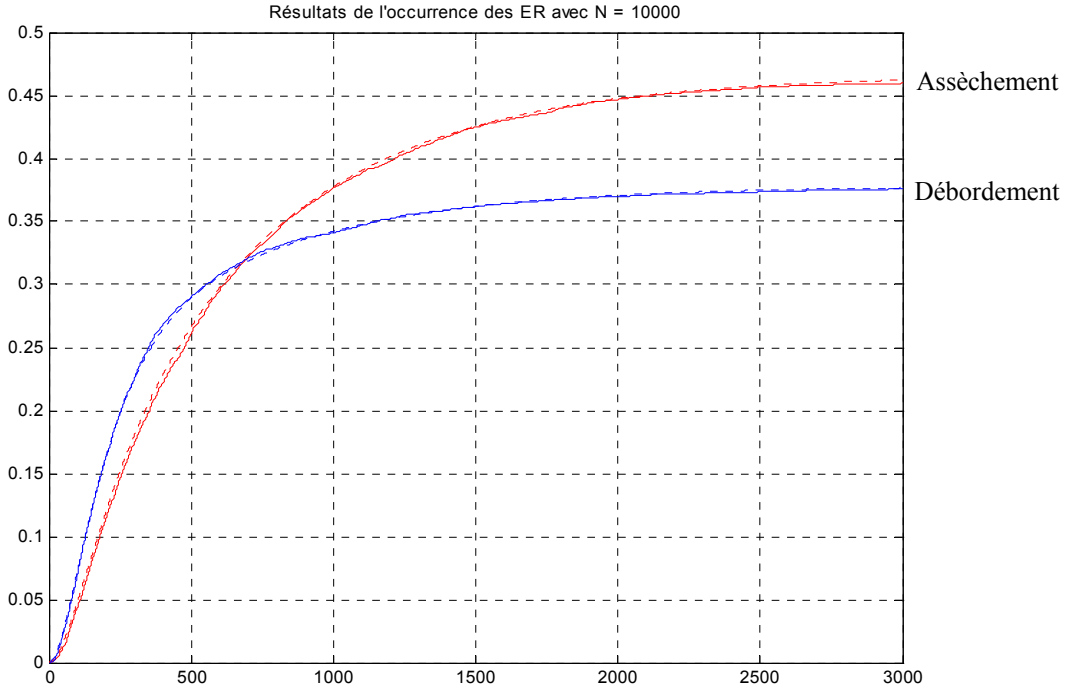


Figure 105 : Courbes obtenues après N=10000 histoires jouées

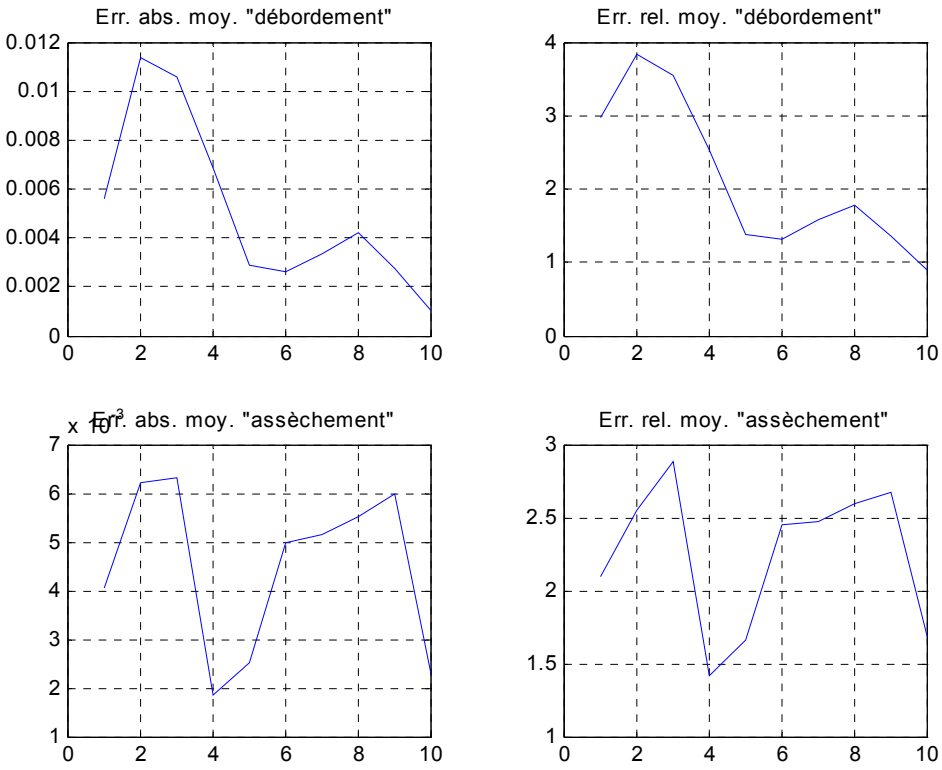


Figure 106 : Evolution des erreurs en fonction de N (x1000)

Simulation N°4				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Non triggé	15000	10000	λ_1=2,2831.10^{-4} λ_2=2,8571.10^{-4} λ_3=1,5625.10^{-4}	Pas variable Ode 45

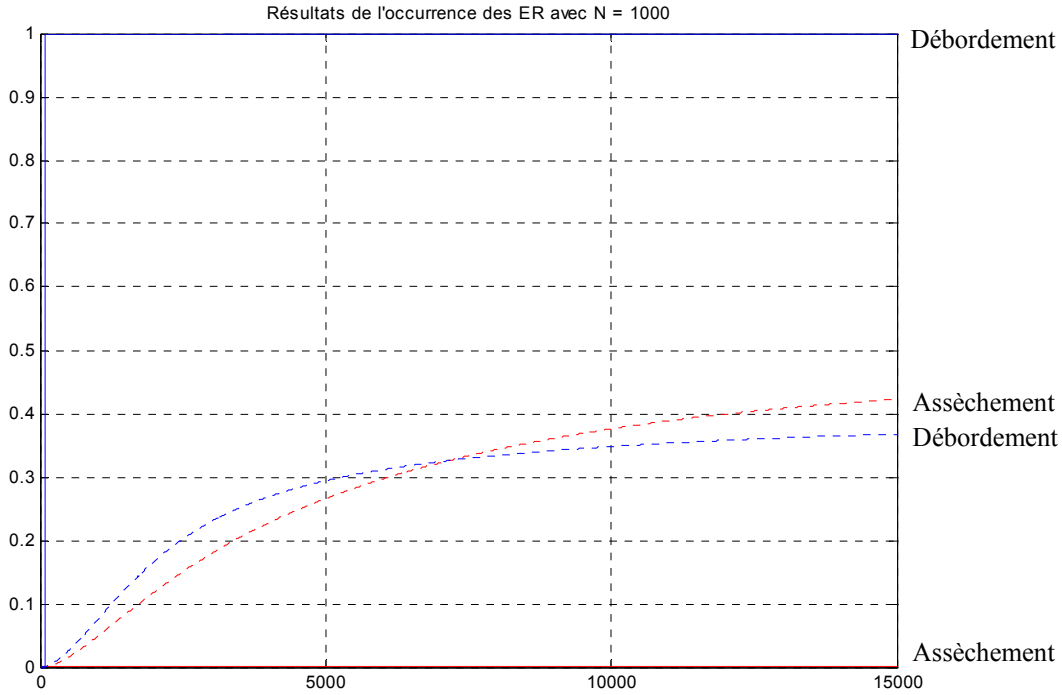


Figure 107 : Courbes obtenues après N=10000 histoires jouées

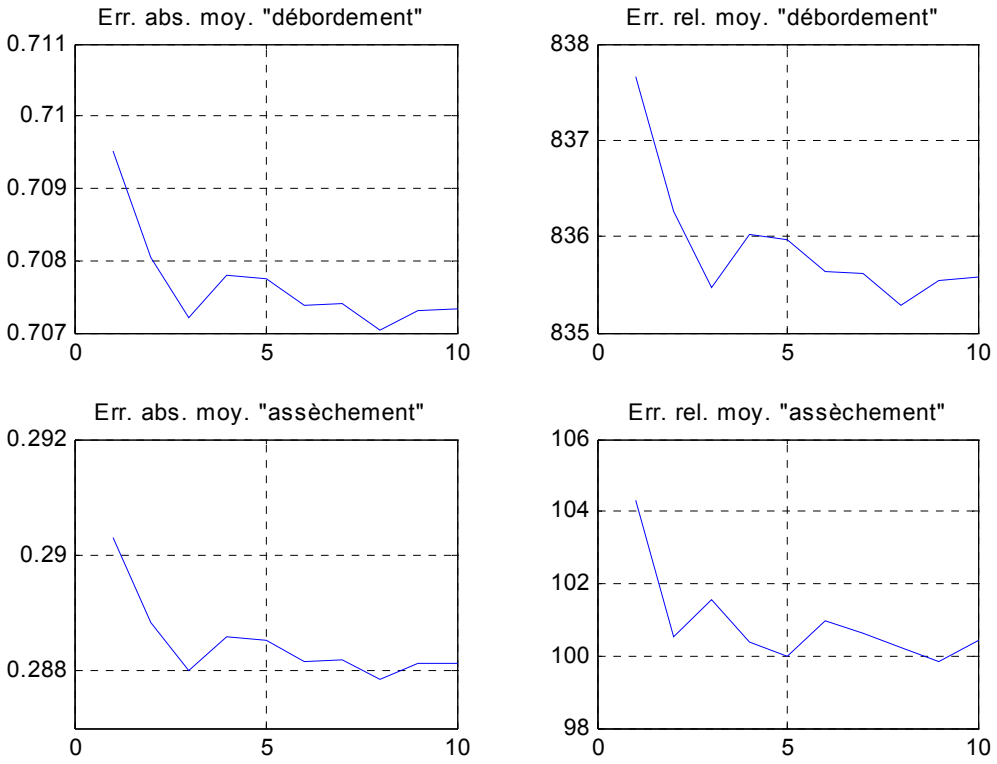


Figure 108 : Evolution des erreurs en fonction de N (x1000)

Simulation N°5				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Non triggé	15000	10000	λ_1=2,2831.10^{-4} λ_2=2,8571.10^{-4} λ_3=1,5625.10^{-4}	Pas fixe Ode 1 Pas = 0,1

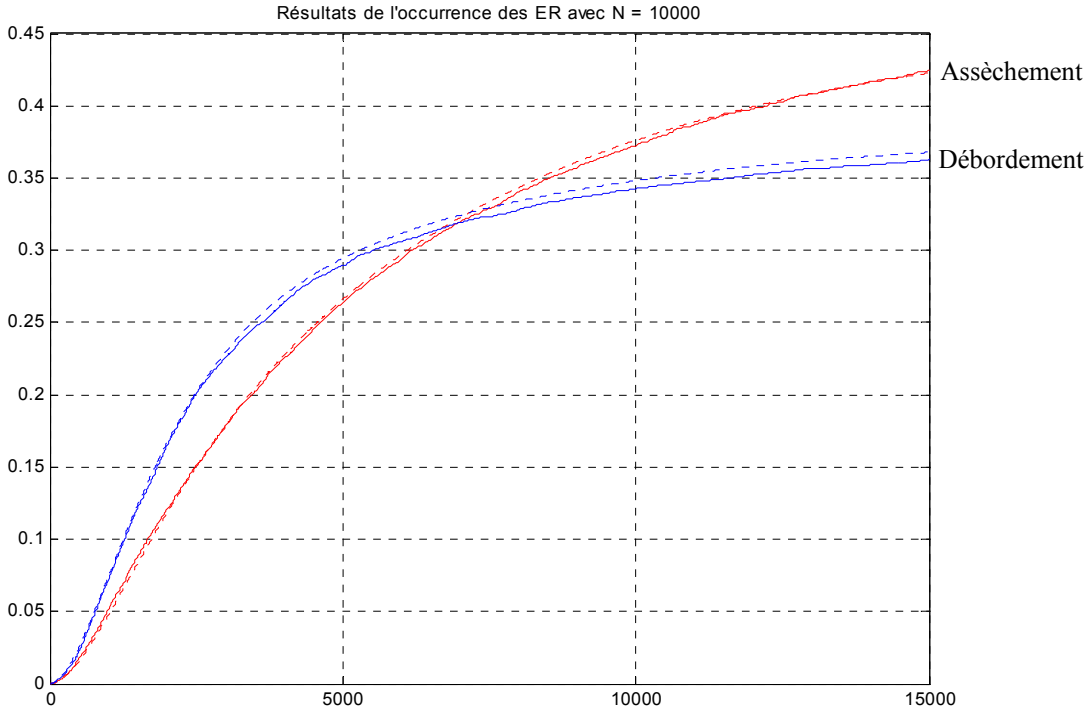


Figure 109 : Courbes obtenues après N=10000 histoires jouées

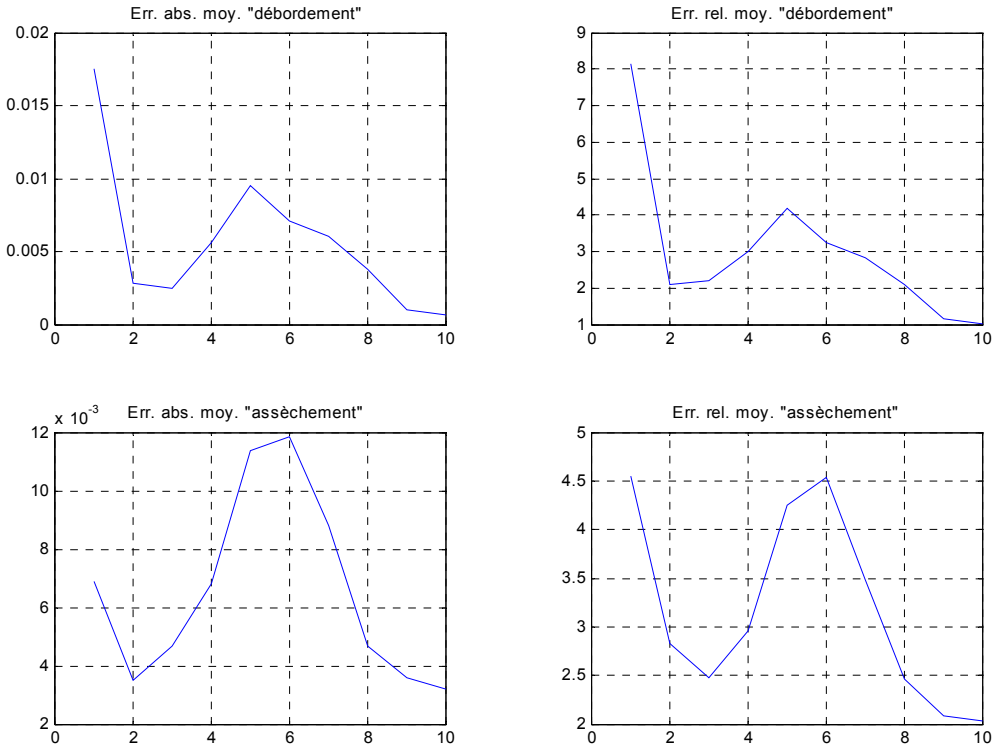


Figure 110 : Evolution des erreurs en fonction de N (x1000)

Simulation N°6				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Triggé	3000	10000	λ_1=2,2831.10^{-3} λ_2=2,8571.10^{-3} λ_3=1,5625.10^{-3}	Pas fixe Ode 1 Pas = 0,1

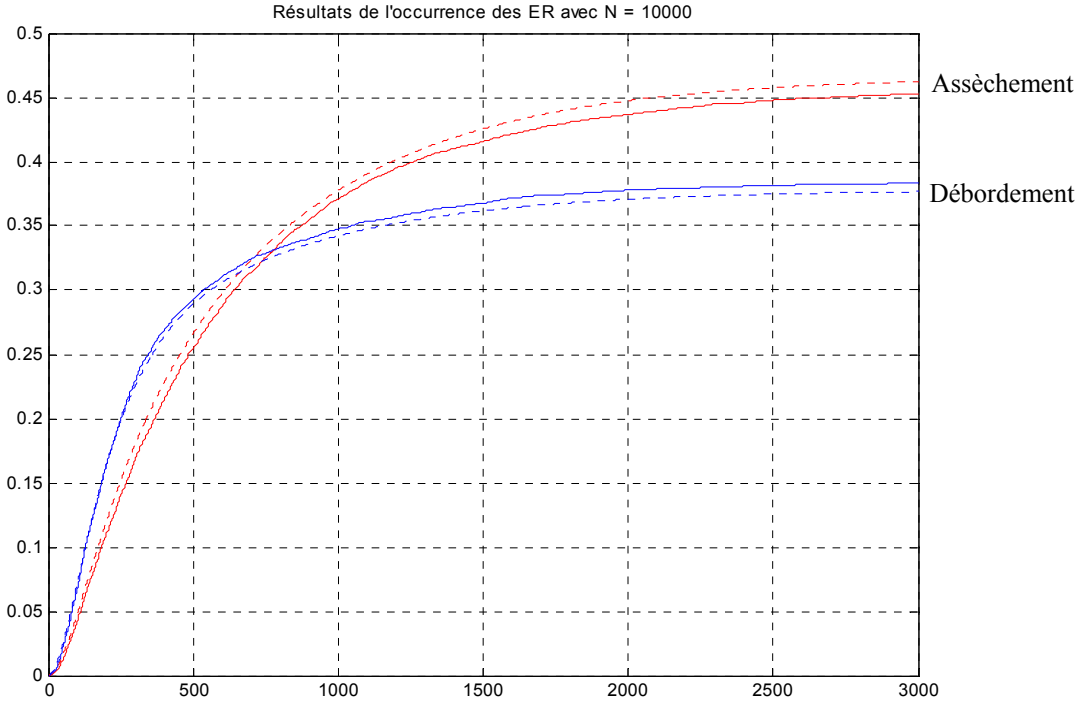


Figure 111 : Courbes obtenues après N=10000 histoires jouées

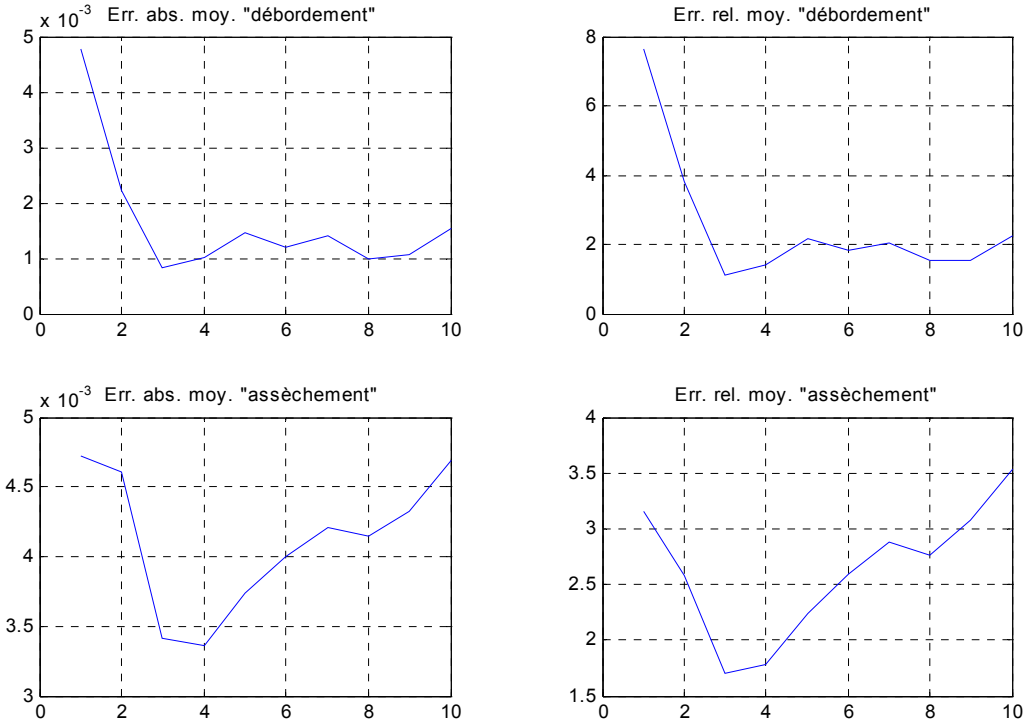


Figure 112 : Evolution des erreurs en fonction de N (x1000)

Simulation N°7				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Triggé	3000	10000	λ_1=2,2831.10^{-1} λ_2=2,8571.10^{-1} λ_3=1,5625.10^{-1}	Pas fixe Ode 1 Pas = 0,1

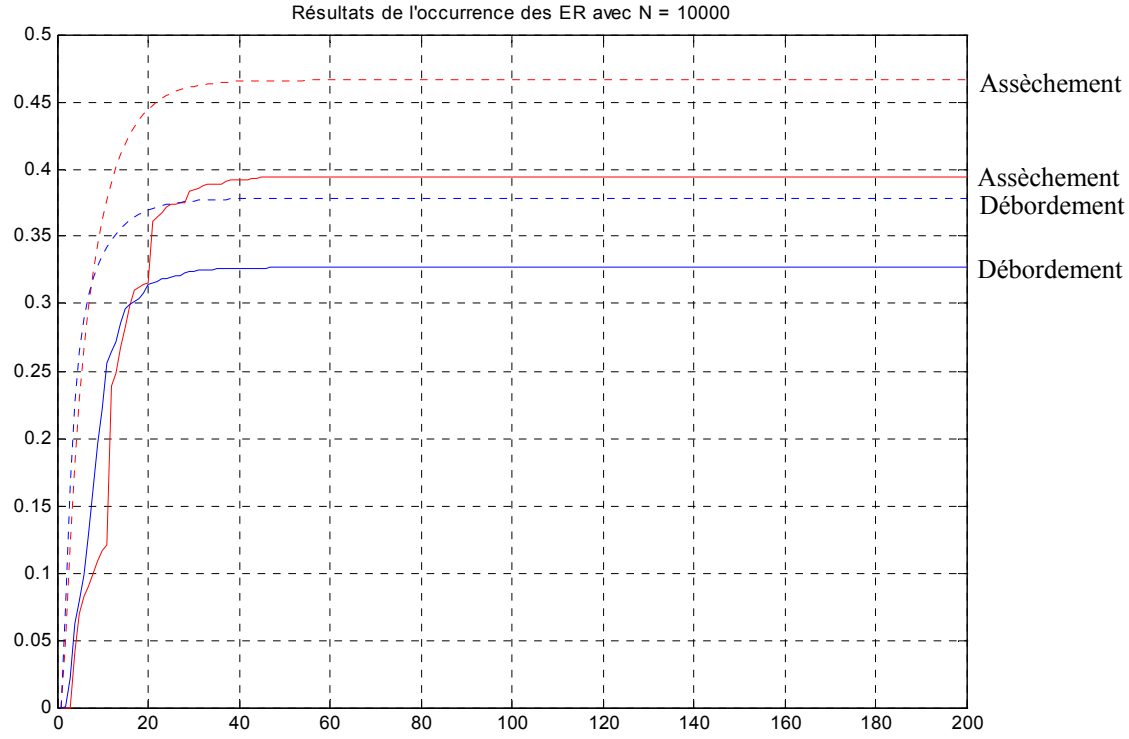


Figure 113 : Courbes obtenues après N=10000 histoires jouées

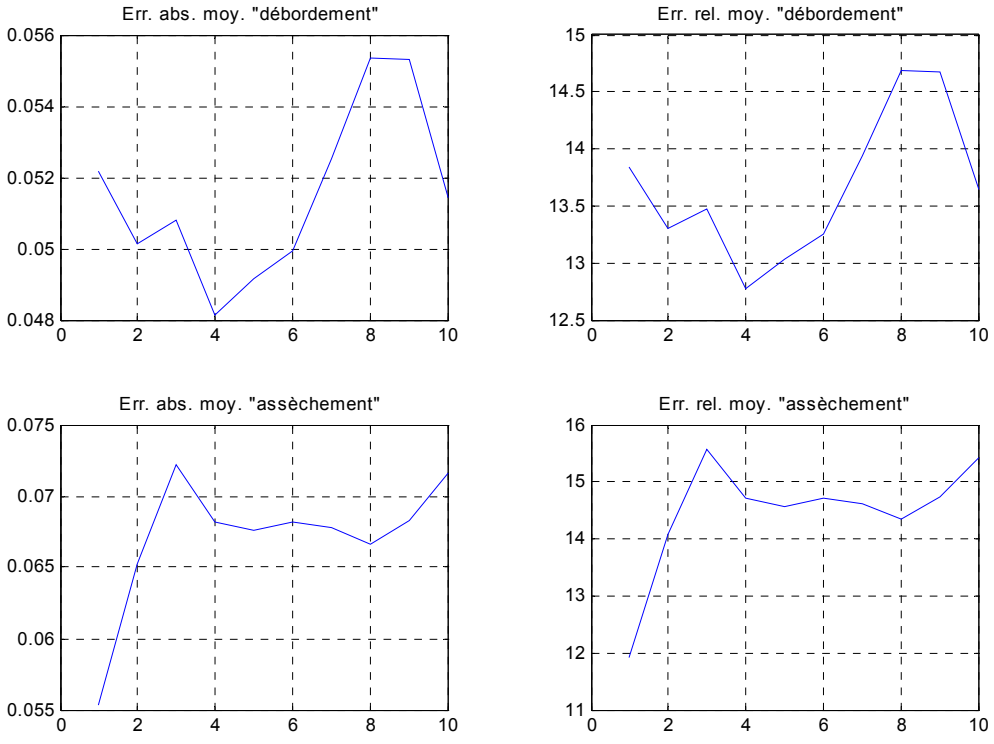


Figure 114 : Evolution des erreurs en fonction de N (x1000)

Simulation N°8				
Modèle	**T**	**N_{max}**	**Taux de défaillance**	**Solveur**
Triggé	150 000	10000	λ_1=2,2831.10^{-5} λ_2=2,8571.10^{-5} λ_3=1,5625.10^{-5}	Pas variable Ode 45

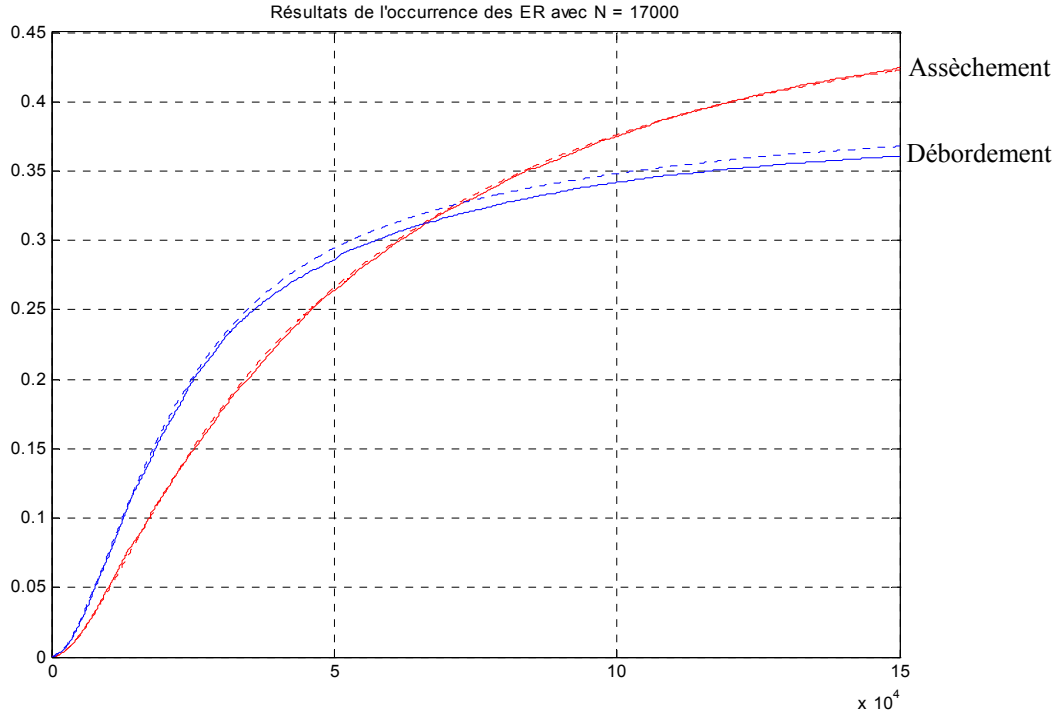


Figure 115 : Courbes obtenues après N=10000 histoires jouées

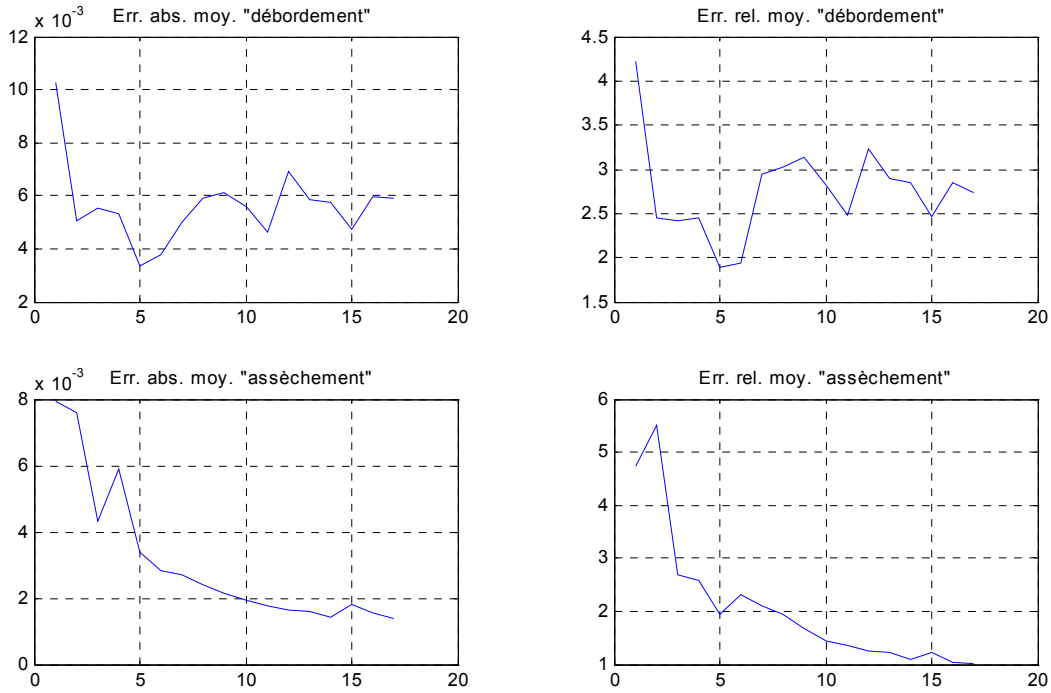


Figure 116 : Evolution des erreurs en fonction de N (x1000)

Références Bibliographiques

[AJM95] AJMONE MARSAN, M. & al. *Modelling with generalized stochastic Petri nets*. Wiley series in parallel computing. 1995.

[AMO99] AMODEO, L. *Contribution à la simplification et à la commande des réseaux de Petri stochastiques. Application aux systèmes de production*. Th. Automatique et informatique. 1999. U.F.R. des Sciences et Techniques de l'Université de Franche-Comté.

[AND96] ANDREU, D. *Commande et supervision des procédés discontinus : approche hybride*. Th. Informatique industrielle. 1996. Laboratoire d'Analyse et d'Architecture des Systèmes.

[ARL95] ARLAT, J. *Informatique sûre de fonctionnement : défis et solutions*. Sûreté des procédés industriels : journées CNRS/CRIN du 11 octobre 1995.

[ATA94] ATAMNA, Y. *Réseaux de Petri temporisés stochastiques classiques et bien formés : définition, analyse et application aux systèmes distribués temps réel*. Th. Informatique industrielle. 1994. Laboratoire d'Analyse et d'Architecture des Systèmes.

[AUB87] AUBRY, JF. *Conception des systèmes de commande numériques des convertisseurs électromécaniques : vers une méthodologie intégrant la sûreté de fonctionnement*. Thèse d'Etat. 1987. Institut National Polytechnique de Lorraine.

[AUB91] AUBRY, J-F & ZANNE, C. *Intégration de la sûreté de fonctionnement dans la conception des systèmes de commande numérique des processus électromécaniques*. APII, AFCET - Sûreté de fonctionnement. 1991, vol. 25, p.297-324.

[BEN82] BENNIS, O. *Analyse et commande des systèmes à deux échelles de temps version continue et discrète*. Th. Automatique. 1982. Université Paul Sabatier de Toulouse.

[BEN97] BENZINA, A. *Analyse et conception des systèmes temps réel : translation d'une approche fonctionnelle à une approche orientée objet*. Th. Informatique industrielle. 1997. Université Paul Sabatier.

[BER98] BERCHTOLD, A. *Chaînes de Markov et modèles de transition : applications aux sciences sociales*. Paris : Hermès, 1998.

[BET02] BETOUS-ALMEIDA, C. *Construction et affinement de modèles de sûreté de fonctionnement – Application aux systèmes de contrôle-commande*. Th. Informatique et télécommunications. 2002. Laboratoire d'Analyse et d'Architecture des Systèmes.

[BOB86] BOBBIO, A. & TRIVEDI, K. *An aggregation technique for the transient analysis of stiff Markov chains*. IEEE Transactions on computers. 1986, vol. C-35, no. 9, p. 803-814.

[BOU90] BOUAYAD, A. *Etude comparative de méthodes d'analyse de systèmes à échelles de temps multiples – Contribution à l'élaboration d'un processeur d'aide à la simplification de modèles*. Th. Automatique. 1990. Université des sciences et techniques de Lille Flandres Artois.

[BOW93] BOWEN, J. & STAVRIDOU, V. *Systèmes à sûreté de fonctionnement critique – Méthodes formelles et normes*. Génie logiciel & systèmes experts. N°30. Mars 1993.

[BRA83] BRAMS, G.W. *Réseaux de Petri : théorie et pratique*. Tome 2 : Modélisation et applications. Paris : Masson, 1983.

[BUC99] BUCHHOLZ, P. *An adaptative aggregation/disaggregation algorithm for hierarchical Markovian models*. European Journal of Operational Research. N°116. Elsevier. 1999. p. 545-564.

[CAB99] CABARBAYE, A. & NGOM, L. *Mise en œuvre de la méthode des états fictifs et génération automatique des matrices de Markov*. Qualita 99, 3ème congrès Qualité et Sûreté de fonctionnement, Paris, 1999. p. 615-625.

[CAL92] CALVEZ, J.P. *Spécification et conception des système – une méthodologie*. Paris : Masson, 1992.

[CAL97] CALMETTES, C. *Résolution numérique de systèmes markoviens : parallélisation de la méthode d'Arnoldi*. Th. Automatique. 1997. Laboratoire d'Analyse et d'Architecture des Systèmes.

[CAS02] CASSEZ, F. & al. *A timed extension for AltaRica*. IRCCyN. 2002.

[CEI90] CEI 80 191. Vocabulaire Electrotechnique International. Chapitre 191 : Sûreté de fonctionnement et qualité de service. 1990.

[CHA98a] Jean-Luc CHABOT. *Simulation hybride, méthode de modélisation intégrant phénomènes continus et discrets*. 11ème Colloque National de Fiabilité & Maintenabilité, Lambda Mu 11, p126-136, Arcachon, octobre 1998.

[CHA98b] Jean-Luc CHABOT. *Approche probabiliste relative à l'étude des scénarios d'incendie*. Thèse de doctorat de l'université de Poitiers, 16 octobre 1998.

[CHAM98a] CHAMPAGNAT, R. *Supervision des systèmes discontinus : définition d'un modèle hybride et pilotage en temps réel*. Th. 1998. Université Paul Sabatier de Toulouse.

[CHAM98b] CHAMPAGNAT, R. & al. *A gazstorage example as a benchmark for hybrid modelling*. ADPM'98. Reims. Mars 1998.

[CHAM98c] CHAMPAGNAT, R. & al. *Modélisation et simulation d'un système hybride à l'aide d'un modèle RdP Pr/Tr-EAD*. ADPM'98. Reims. Mars 1998.

[CHAM99] CHAMPAGNAT, R. & al. *Simulation orientée événements des modèles hybrides*. MOSIM'99. Annecy. Octobre 1999. p. 279-284.

[COS01] COSTE, P. *Conception des systèmes hétérogènes multilangages*. Th. Informatique. 2001. Université Joseph Fourier, Grenoble 1.

[DAU85] DAUPHIN-TANGUY, G., BORNE, P., FOSSARD, A. *Analyse et synthèse des systèmes à plusieurs échelles de temps*. APII, AFCET-Automatique. 1985, vol. 19, N°2, p. 169-196.

[DAV89] DAVID, R. & ALLA, H. *Du grafcet aux réseau de Petri*. Paris : Hermès, 1989.

[DUT96] DUTUIT, Y. & al. *Modélisation d'un système dynamique simple et évaluation de sa fiabilité par réseaux de Petri stochastiques*. Lambda-Mu 10, Saint-Malo, octobre 1996.

[DUT97] DUTUIT, Y. & al. *Dependability modelling and evaluation by using stochastic Petri nets : application to two test cases*. Reliability Engineering ans System Safety. Elsevier Science. N°55. 1997. p. 117-124.

[ELE01] Electronique International. *Dossier automobile : l'électronique accélère l'innovation*. N°448. 11 octobre 2001.

[ERE94] EREAU, JF. & al. *Réseaux de Petri pour l'évaluation des systèmes redondants*. Lambda Mu 9/ ESREL 1994, La Baule, 1994.

[ERE97] EREAU, J-F. *Réseaux de Petri pour l'étude de la disponibilité opérationnelle des systèmes spatiaux d'avant-projet*. Th. Automatique et informatique industrielle. 1997. Laboratoire d'Analyse et d'Architecture des Systèmes.

[FLO85] FLORIN, G. & NATKIN, S. *Les réseaux de Petri stochastiques*. Techniques et sciences informatiques, AFCET. 1985, vol. 4, n°1, p. 143-160.

[GAR96] GARNIER, R. & al. *Une méthode complète d'accélération de la simulation de Monte-Carlo d'un réseau de Petri stochastique généralisé pour les systèmes non markoviens*. ESREL, Crête, 1996.

[GAR98] GARNIER, R. *Une méthode efficace d'accélération de la simulation des réseaux de Petri stochastiques*. Th. Automatique, productique. 1998. Université Bordeaux I.

[GRIFF] GRIFFAULT, A. *Conception et validation d'un protocole avec le modèle AltaRica*. LaBRI, Université Bordeaux I.

[HAR87] HAREL, D. *Statecharts : a visual formalism for complex systems*. Science of Computer Programming 8. North-Holland : Elsevier Science Publishers B.V. Vol. 8, N°3. 1987, p. 231-274.

[HEN96] HENAULT, V. *Méthodologie de développement des systèmes électroniques embarqués automobiles, matériels et logiciels, sûrs de fonctionnement.* Th. Electronique. 1996. Université de Nantes.

[HEN97] HENAULT, V. *Sûreté de fonctionnement en embarqué*. Phoebus – La revue de la sûreté de fonctionnement. N°2. 1997.

[INC01] INCERA DIEGUEZ, J.A. *Contributions à la modélisation et à la simulation accélérée de réseaux de communication*. Th. Informatique. 2001. Université de Rennes 1.

[JAM01a] JAMPI, D. & al. *Spécification de systèmes de contrôle-commande embarqués et évaluation de sûreté de fonctionnement : fusion de deux activités*. Qualita 2003- Congrès Qualité et Sûreté de Fonctionnement, Annecy, mars 2001.

[JAM01b] JAMPI, D. & al. *Conception et sûreté de fonctionnement : deux activités indissociables*. 3ème Conférence Francophone de Modélisation et Simulation MOSIM 2001, Troyes, avril 2001.

[JAM01c] JAMPI, D. *Détermination d'une méthodologie d'aide à la conception d'un système de contrôle commande numérique sûr de fonctionnement*. Th. Automatique. 2001. Institut National Polytechnique de Lorraine.

[KOK99] KOKOTOVIC P., KHALIL H. K., O'REILLY J. *Singular Perturbation Methods in Control, Analysis and Design*. Philadelphia : SIAM, 1999.

[KOR92] KORDON, F. *Prototypage de systèmes parallèles à partir de réseaux de Petri colorés : application au langage ADA dans un environnement centralisé ou réparti*. Th. Informatique. 1992. Université Pierre & Marie Curie (Paris VI).

[LAB02a] LABEAU, P-E. & KERMISCH, C. *Approche dynamique de la fiabilité des systèmes*. Rapport MNFD 2002-10, Service de Métrologie Nucléaire, Université Libre de Bruxelles, novembre 2002.

[LAB02b] LABEAU, P-E. & KERMISCH, C. *Approche dynamique de la fiabilité des systèmes*. Rapport MNFD 2002-07, Service de Métrologie Nucléaire, Université Libre de Bruxelles, juillet 2002.

[LAF91] LAFIT, S. *Graphes d'événements déterministes et stochastiques : application aux systèmes de production*. Th. Mathématiques, automatique. 1991. Université Paris IX Dauphine.

[LAP95] LAPRIE, J-C, & al. *Guide de la sûreté de fonctionnement*. Toulouse : Cépaduès. 1995.

[LEN00] LENOIR, F-X. *S'engager sur la sécurité de fonctionnement*. Industries et Techniques. N°818. 2000.

[MAZ95] MAZIGH, B. & GRESSER, J. *Evaluation de la sûreté de fonctionnement des systèmes de production par réseaux de Petri stochastiques généralisés*. Diagnostic et sûreté de fonctionnement. 1995, vol. 5, n°3, p. 255-272.

[MES94] MESNARD, E. *Contribution à la vérification du comportement temporel d'applications Temps Réel*. Th. Automatique & Informatique Appliquée. 1994. Université de Nantes.

[MIC97] MICHALCZYK, P. *La sécurité dans le programme RAFALE. Phoebus* – La revue de la sûreté de fonctionnement. N°1. 1997.

[MOCA] IXI-GFI Consulting. *Manuel de l'utilisateur MOCA-RP V10.04*. Octobre 2000.

[MOI91] MOITESSIER, F. & al. *Une méthode de conception des systèmes de commande temps réel répartis pour processus physiques rapides et mixtes*. ECC91. Grenoble. 1991.

[MOL81] MOLLOY, M.K. *On the integration of delay and throughput measures in distributed processing models*. Th. 1981. Université de Californie, Los Angeles, EU.

[MOL82] MOLLOY, M.K. *Performance analysis using Stochastic Petri Nets*. IEEE Transactions on computers. 1982, vol. C31, n°9, p. 913-917.

[MON98] MONCELET, G. *Application des réseaux de Petri à l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques du monde automobile*. Th. Automatique et informatique industrielle. 1998. Laboratoire d'Analyse et d'Architecture des Systèmes.

[NAT80] NATKIN, S. *Les réseaux de Petri stochastiques et leur application à l'évaluation des systèmes informatiques*. Th. Informatique. 1980. Conservatoire National des Arts et Métiers. Paris.

[OCAS] GFI Consulting. *Manuel Utilisateur OCAS V2.5*. Novembre 2001.

[OFT97] Observatoire Français des Techniques Avancées. *Application des techniques formelles au logiciel*. Série ARAGO N°20. Paris, 1997.

[OLI94] OLIVERO, A. *Modélisation et Analyse de Systèmes Temporisés et Hybrides*. Th. Informatique.1994. Institut National Polytechnique de Grenoble.

[PAG80] PAGES, A., GONDRAN, M. *Fiabilité des systèmes*. Paris : Eyrolles, 1980.

[PEL91] PELLAUMAIL, J. *Systèmes markoviens discrets stationnaires et applications*. Publication interne n°579. 1991. Institut de recherche en informatique et systèmes aléatoires.

[POINT] POINT, G. & GRIFFAULT, A. *AltaRica : modélisation et vérification d'un régulateur de vitesse*. LaBRI.

[QUA93] QUAGLIARO, L. *Une nouvelle méthode pour l'analyse quantitative de la sûreté de fonctionnement des systèmes : la méthode des graphes fictifs*. Th. Contrôle des systèmes. 1993. Université de technologie de Compiègne.

[RAC97] RACOCEANU, D. *Contribution à la modélisation et à l'analyse des chaînes de Markov à échelles de temps et échelles de pondérations multiples. Application à la gestion d'un système hydro-énergétique*. Th. Automatique et informatique. 1997. U.F.R. des Sciences et Techniques de l'Université de Franche-Comté.

[SAH96] SAHRAOUI, A. *Vers une intégration de méthodes fonctionnelle et de défaillances*. 10ème Colloque National de Fiabilité & Maintenabilité, Lambda Mu 10, p. 737-750, Saint-Malo, octobre 1996.

[SCH02] SCHOENIG, R. & LEBLOND, A. *Méthodologie outillée d'aide à la conception des systèmes embarqués sûrs de fonctionnement*. Lambda Mu 13-ESREL 2002, Lyon, 2002, p. 1-9.

[SCH03a] SCHOENIG, R. & al. *An aggregation method of Markov graphs for the reliability analysis of hybrid systems*. Qualita 2003-5ème Congrès Qualité et Sûreté de Fonctionnement, Nancy France, mars 2003.

[SCH03b] SCHOENIG, R. & al. *An example of reliability assessment by an aggregation method of Markov graphs for hybrid systems.* Qualita 2003-5ème Congrès Qualité et Sûreté de Fonctionnement, Nancy France, mars 2003.

[SCH03c] SCHOENIG, R. & al. *Exemple de calcul de la fiabilité d'un système hybride*. 4ème Conférence internationale sur l'Automatisation Industrielle, Montréal Canada, juin 2003.

[SCH03d] SCHOENIG, R. & AUBRY, JF. *A design methodology for embedded control systems including safety assessment studies*. ESREL 2003, Maastricht The Netherlands, juin 2003.

[SCH04] SCHOENIG, R. & al. *Une méthodologie de conception des systèmes mécatroniques sûrs de fonctionnement*. Lambda Mu 14, Bourges, octobre 2004.

[SCH99] SCHNOEBELEN, P. *Vérification de logiciels – Techniques et outils du model-checking*. Paris : Vuibert, 1999.

[SIG96] SIGNORET, J-P. *Modélisation et simulation dans le domaine de la sûreté de fonctionnement.* Revue de l'électricité et de l'électronique. N°8. Septembre 1996.

[SIG97] SIGNORET, J-P. *Bâtir le futur sur les acquis du passé*. Phoebus – la revue de la sûreté de fonctionnement. N°3. 1997.

[SIMFI] SOFRETEN, *Manuel SIMFIA – Atelier de simulation pour l'ingénierie système.*

[TRO00] TROMP, L. *Surveillance et diagnostic de systèmes industriels complexes : une approche hybride numérique/symbolique*. Th. Traitement du signal et télécommunications. 2000. Université de Rennes 1.

[VIL88] VILLEMEUR, A. *Sûreté de fonctionnement des systèmes industriels : fiabilité, facteurs humains, informatisation*. Paris : Eyrolles, 1988.

[VIN03] VINCENT, A. *Conception et réalisation d'un vérificateur de modèles AltaRica*. Th. Informatique. 2003. Université Bordeaux I.

[WIL85] WILLIAMS, TWC. *Obtaining minimal Gershgorin discs by scaling the states*. International Journal of control.1985, vol. 42, n°5, p1155-1173.

[ZAN95] ZANNE, C. *Contribution à la conception des dispositifs de commande pour les systèmes dynamiques hybrides*. Rapport de synthèse présenté à l'INPL en vue d'obtenir l'habilitation à diriger les recherches. Automatique. 1995.

[ZIE96] ZIEGLER, C. *Sûreté de fonctionnement d'architectures informatiques embarquées sur automobile*. Th. Informatique. 1996. Institut National Polytechnique de Toulouse.

Résumé

La révolution technologique que connaît le secteur automobile avec la multiplication des systèmes électroniques et informatiques embarqués n'est pas près de s'essouffler. Il est manifeste que la complexité croissante et la spécificité des systèmes embarqués justifieront de plus en plus le besoin de structurer l'ensemble des activités de développement. Vérification, validation, conception et bien entendu sûreté de fonctionnement font partie intégrante d'un même processus. Nous cherchons tout d'abord à définir un formalisme de modélisation fonctionnelle et comportementale, support de la méthodologie, et en cohérence avec les spécificités des systèmes mécatroniques. En particulier, les aspects *hybrides* et *temps réel* doivent pouvoir être représentés. Ensuite, sur le plan de la vérification et de la validation, nous proposons d'exploiter les méthodes formelles, telles que le model-checking, en complément des tests et des simulations habituellement utilisées dans l'industrie. Enfin nous attachons une importance centrale à la sûreté de fonctionnement. Afin de pallier aux insuffisances et aux limites des méthodes habituellement utilisées, nous proposons une approche basée sur la construction d'un graphe de Markov agrégé. L'originalité tient dans sa capacité de répondre à un problème de représentation et d'évaluation de la fiabilité des systèmes dynamiques hybrides. Les principales étapes consistent à découpler la dynamique du système et la dynamique du processus de défaillance grâce à la théorie des perturbations singulières, puis d'identifier et estimer les grandeurs du système influençant la dynamique des défaillances. Ces grandeurs sont évaluées par de simples simulations dans le but de pouvoir traiter des systèmes complexes. Ceux-ci sont alors intégrés dans le graphe agrégé.

Mots-clés : Agrégation, graphe de Markov, perturbations singulières, simulation, système dynamique hybride, sûreté de fonctionnement.

Abstract

The introduction of embedded electronic in car industry induced a technological revolution that will last for years. Embedded electronic systems are more and more complex and their features involve the adaptation and structuring of the whole design activities. Verification, validation, design, and of course safety analysis should be integrated in a same process. First, we identify a class of formalism able to represent the functional and behavioral aspects of the system, and suited for hybrid and real time features. This model is the support of the methodology. Then, we propose an exploitation of formal methods, such as model-checking, to validate and check properties of the model and complement usual methods like tests and simulations. Finally, we particularly emphasize the dependability aspects. A method, based on the construction of an aggregated Markov graph, is developed. The advantage of this approach is to be able to model and assess the reliability of a dynamic hybrid system. The main steps consist in spliting up the dynamic of the system and the dynamic of the failure process by the mean of the singular perturbation method. Then, we identify and we evaluate variables of the system which interact with the failure dynamic. Simulation is used to estimate these variables, because of its ability to handle complex systems. The variables are then integrated in the aggregated Markov graph.

Keywords : Aggregation, Markov graph, singular perturbation, simulation, dynamic hybrid system, dependability.