Évaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride

Gabriel Antonio Perez Castaneda

▶ **To cite this version:**

Gabriel Antonio Perez Castaneda. Évaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride. Autre. Institut National Polytechnique de Lorraine, 2009. Français. NNT : 2009INPL016N . tel-01748753

HAL Id: tel-01748753
https://hal.univ-lorraine.fr/tel-01748753v1

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AVERTISSEMENT

Ce document est le fruit d'un long travail approuvé par le jury de soutenance et mis à disposition de l'ensemble de la communauté universitaire élargie.

Il est soumis à la propriété intellectuelle de l'auteur. Ceci implique une obligation de citation et de référencement lors de l'utilisation de ce document.

D'autre part, toute contrefaçon, plagiat, reproduction illicite encourt une poursuite pénale.

Contact : ddoc-theses-contact@univ-lorraine.fr

LIENS

Code de la Propriété Intellectuelle. articles L 122. 4
Code de la Propriété Intellectuelle. articles L 335.2- L 335.10
http://www.cfcopies.com/V2/leg/leg_droi.php
http://www.culture.gouv.fr/culture/infos-pratiques/droits/protection.htm

École doctorale IAEM Lorraine
Département de Formation Doctorale en Automatique

Institut National Polytechnique de Lorraine

Évaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride

THÈSE

présentée et soutenue publiquement le 30 mars 2009

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

Spécialité Automatique et Traitement du Signal

par

Gabriel Antonio PÉREZ CASTAÑEDA

Composition du jury

Président :	Yves Dutuit	Professeur Émérite à l'Université de Bordeaux
Rapporteurs :	Christophe Bérenguer	Professeur à l'Université Technologie de Troyes
	Dimitri Lefebvre	Professeur à l'Université du Havre
Examinateurs :	Ramine Nikoukhah	Directeur de Recherches à l'INRIA
	Jean-François Aubry	Professeur à l'INPL
	Nicolae Brinzei	Maître de Conférences à l'INPL

Centre de Recherche en Automatique de Nancy CNRS – UMR 7039
Nancy Université

Remerciements

Merci, tout d'abord, à toi Seigneur pour ta grâce et ta fidélité envers moi. Merci aussi pour tes promesses qui sont fidèles et vraies !

« Ne crains pas, car je suis avec toi ; ne sois pas inquiet, car moi je suis ton Dieu. Je te fortifierai ; oui, je t'aiderai ; oui, je te soutiendrai par la droite de ma justice »

La Bible, Ésaïe 41:10, traduction J. N. Darby

Les travaux de recherche présentés dans ce mémoire ont été effectués au sein de l'équipe SACSS (Systèmes Automatisés Contraints par la Sûreté de fonctionnement et la Sécurité) du groupe thématique SURFDIAG (Sûreté de Fonctionnement et Diagnostic des systèmes) du Centre de Recherche en Automatique de Nancy (CRAN) de l'Institut National Polytechnique de la Lorraine (INPL).

Je tiens à remercier la Direction Générale de l'Education Supérieure Technologique et l'Institut Technologique de Tehuacán du Mexique pour le financement et l'opportunité qui m'ont été donnés de réaliser cette thèse.

J'exprime mes profonds remerciements à mes deux directeurs de thèse de l'Institut National Polytechnique de Lorraine, Monsieur le Professeur Jean-François AUBRY et Monsieur le Docteur Nicolae BRINZEI, pour leur aide inestimable, leur patience, leur disponibilité et leurs encouragements tout au long de ce travail. Leurs compétences ont été un atout indéniable à la réussite de ces travaux et m'ont permis d'apprendre énormément au cours de ces années. J'ai été honoré de travailler avec eux !

Je souhaiterais aussi remercier les rapporteurs, Monsieur Christophe BERENGUER Professeur à l'Université Technologie de Troyes et Monsieur Dimitri LEFEBVRE Professeur à l'Université du Havre, qui ont accepté de se plonger dans ce sujet et pour l'intérêt et l'attention qu'ils ont accordés à cette étude. Ce fut un honneur pour moi qu'ils aient accepté d'évaluer mes travaux.

Je les prie de croire ici en ma haute considération aux membres du jury qui m'ont fait l'honneur de participer à l'examen de ces travaux : Monsieur Yves DUTUIT

Professeur Émérite à l'Université de Bordeaux, qu'il soit assuré de toute ma gratitude et mon profond respect pour l'attention qu'il a portée à mon travail en acceptant de présider ce jury de thèse ; Monsieur Ramine NIKOUKHAH Directeur de Recherches à l'INRIA, je lui suis profondément reconnaissant d'avoir accepté d'évaluer mon travail, témoignant ainsi de son intérêt, aussi que pour toutes les suggestions qu'il m'a faites par rapport à la boîte à outils Scicos de Scilab.

Je souhaite également exprimer ma reconnaissance et ma sympathie à tous ceux qui m'ont aidé au cours de ces années, notamment :

Mes collègues de travail de l'académie de Génie Electromécanique de l'Institut Technologique de Tehuacán pour leur appui. Un grand merci à l'Ingénieur Rodolfo MEDRANO pour son appui et la confiance qu'il m'a accordée.

Les membres de l'équipe SACSS qui m'ont aidé avec leurs remarques, leurs conseils et leurs suggestions pendant les présentations de mes travaux.

Tout le personnel professoral et administratif du Centre de Recherche en Automatique de Nancy pour son soutien et sa disponibilité.

Mon frère Oscar et sa famille pour leur aide à mon arrivée avec ma famille à Nancy en France, de même Manuel MEDINA et Rubén POSADA.

Masoud NAJAFI de l'INRIA pour son aide tellement précieuse par rapport au logiciel Scicos.

Mes amis Juan Manuel HERNANDEZ, Rosebet MIRANDA, Rebeca ROMO, Hugo VELEZ, Gilberto DIAZ et Ricardo SALIDO pour leurs encouragements et les agréables moments vécus en leur compagnie.

Mes collègues Emilie PERY et Yahir HERNANDEZ pour les échanges scientifiques que nous avons eus en plusieurs occasions et qui ont contribué à mes recherches.

Mes frères et sœurs en Jésus-Christ qui sont en France et au Mexique pour leur soutien par la prière.

A Philippe et Margarita VIARD ainsi que leurs enfants, pour leur amitié et leur accueil depuis notre arrivée en France, à Nancy.

Mes parents, mes frères et leurs familles pour leurs encouragements et leur soutien par la prière pendant ces années de travaux.

Que ceux que j'aurais oubliés ne m'en tiennent pas rigueur. Qu'ils trouvent là, la marque de ma profonde gratitude.

Enfin, mes remerciements les plus chaleureux vont à ma famille : à mon épouse et à mes filles pour m'avoir soutenu pendant toute la durée de mes recherches.

Je dédie cette thèse à ma chère épouse, Vicky,
et à mes quatre adorables filles,
Nuri, Katia, Stéfanny et Cinthia.
Merci de votre amour et de votre soutien !

A Dieu soit la gloire !

Table des matières

Production scientifique

Simulation de Monte Carlo par Automate Stochastique Hybride, application à un cas test pour la fiabilité dynamique - PEREZ CASTANEDA Gabriel Antonio, AUBRY Jean-François, BRINZEI Nicolae - 8ème Congrès International Pluridisciplinaire Qualité et Sûreté de Fonctionnement Qualita 2009, 18 – 20 mars 2009 - Besançon, France.

Simulation de Monte Carlo pour la fiabilité dynamique prenant en compte le vieillissement des composants - PEREZ CASTANEDA Gabriel Antonio, MARZAT Julien, AUBRY Jean-François, BRINZEI Nicolae - Congrès Lambda Mu 16 Maîtrise des Risques et Sûreté de Fonctionnement, 7 – 9 octobre 2008 – Avignon, France.

Automate stochastique appliqué à la fiabilité dynamique - PEREZ CASTANEDA Gabriel Antonio, AUBRY Jean-François, BRINZEI Nicolae - MOSIM 2008, 31 mars – 2 avril 2008 - Paris, France - sélectionné pour la revue JESA.

Modélisation et simulation d'un système dynamique hybride pour calculer sa fiabilité en utilisant la toolbox scicos de scilab - PEREZ CASTANEDA Gabriel Antonio, AUBRY Jean-François, BRINZEI Nicolae - 7è Congrès International Pluridisciplinaire Qualité et Sûreté de Fonctionnement Qualita 2007, 20 – 22 mars 07 - Tanger, Maroc.

Etat de l'art en fiabilité dynamique - PEREZ CASTANEDA Gabriel Antonio, AUBRY Jean-François, BRINZEI Nicolae - 2èmes Journées Doctorales du GDR MACS JDMACS, 9 – 12 juillet 2007 – Reims, France.

Table de figures

Liste des Tableaux

Liste des acronymes

ADS « Analyse of Dynamic Systems » (anciennement « Accident Dynamic Simulator »)

AH Automate hybride

ASH Automate stochastique hybride

DBN « Dynamic Bayesian Networks »

DETAM « Dynamic Event Tree Analysis Method »

DYLAM « Dynamic Logical Analytical Methodology »

MDT « Mean down time **»**

MC Monte Carlo

MTBF « Mean operating time between failures **»**

MTTF « Mean time to failure **»**

MTTR « Mean time to repair or restoration **»**

MUT « Mean up time »

PDMP **«** Piecewise deterministic Markov processes »

SdF Sûreté de fonctionnement

SME Système multi – état

RdP Réseaux de Petri

SDH Systèmes dynamiques hybrides

SED Systèmes à événements discrets

Introduction

Il n'est plus nécessaire aujourd'hui de montrer l'importance de l'évaluation prévisionnelle quantitative de la fiabilité dans la conception des systèmes. La demande sociétale croissante en matière notamment de sécurité se traduit de plus en plus dans les normes et réglementations par l'introduction d'exigences de calculs probabilistes. La complexité des systèmes conçus aujourd'hui par l'homme fait que trop peu de professionnels sont capables de mener ces calculs avec rigueur. Rappelons que cette complexité fait aujourd'hui que le problème de l'évaluation de la fiabilité d'un système est sorti du contexte combinatoire des premières approches pour entrer dans celui du contexte dynamique où la dépendance entre fiabilité du système et celle de ses composants est devenue très complexe.

La recherche de solutions analytiques pour l'évaluation de la fiabilité en contexte dynamique n'est pas résolue dans le cas général. Des approches partielles relatives à des hypothèses particulières existent. Les algorithmes d'optimisation numérique ont déjà prouvé leur capacité à améliorer l'efficacité d'approches analytiques, pour des cas simples, de l'évaluation de la sûreté de fonctionnement dans un contexte de fiabilité dynamique. Malheureusement, ces approches analytiques deviennent inapplicables lorsqu'il s'agit de systèmes complexes. Il est en effet dans ce domaine impossible de donner une forme analytique à la fonction de structure qui décrit la défaillance d'un système en fonction de celles de ses composants. L'évolution de ces systèmes est en général décrite par un automate représentatif d'un ensemble d'états discrets possibles pour le système, chacun d'entre eux étant décrit par un ensemble d'équations différentielles représentatives de l'évolution de variables continues. Il existe dans ces systèmes de très fortes dépendances entre les modèles fiabilistes des durées de vie des composants et certaines variables d'état continues du système, de même que ces modèles évoluent avec l'état discret en prenant des formes multiples.

La simulation de Monte Carlo (MC) serait le seul moyen d'évaluer la sûreté de fonctionnement des systèmes hybrides complexes si des logiciels adéquats existaient. Il n'existe pas, à notre connaissance, d'outils performants permettant la simulation simultanée de l'évolution discrète du système et de son évolution continue prenant en compte les aspects probabilistes. Dans ce contexte, nous avons introduit le concept

d'automate stochastique hybride (ASH) et développé une approche de simulation de MC dans l'environnement Scicos Scilab. Notre travail est présenté dans un document comprenant 4 chapitres.

Le premier chapitre est un état de l'art en matière de fiabilité dynamique, concept qui recouvre les activités d'évaluation probabiliste de la sûreté de fonctionnement des systèmes dans un contexte dynamique hybride. Il correspond à la première année de notre travail qui a exigé de notre part une importante reconversion.

Dans le second chapitre, nous présentons une approche de simulation d'un système de régulation thermique reconfigurable capable d'illustrer les principaux problèmes posés en fiabilité dynamique. Cette simulation a été réalisée dans l'environnement Scicos de Scilab dans lequel il n'existait pas de bibliothèque bien adaptée au problème. Bien qu'il soit arrivé à un résultat, celui-ci manquait de généralité et de réutilisabilité. Fort opportunément, l'INRIA travaillait à l'introduction dans cet environnement d'un automate hybride qui a permis d'avancer substantiellement dans le travail. Nous avons alors défini formellement le concept d'automate stochastique hybride et proposé une implantation dans l'environnement Scicos Scilab et proposé une résolution de l'évaluation de la sûreté de fonctionnement de l'exemple de régulation thermique.

Le chapitre 3 est consacré à l'application de la démarche en prenant en compte dans l'exemple du four des modes de vieillissement multiples des composants dépendant, pour certains, des variables continues, en l'occurrence la température. Nous avons également développé la simulation du cas test ou « benchmark » utilisé dans la communauté de la fiabilité dynamique : un réservoir contrôlé en niveau par une vanne et deux pompes et en température par une source de chaleur. Des résultats cohérents avec ceux développés dans d'autres approches comme par exemple les processus déterministes par morceaux, ont été obtenus. Bien que la méthode soit théoriquement capable de prendre en compte des aspects complexes que d'autres n'abordent pas, elle reste limitée par les performances des solveurs intégrés dans Scilab et par les interactions complexes entre Scicos et Scilab qui sont potentiellement améliorables.

Le chapitre 4 a pour objet d'introduire le rôle du diagnostic dans l'évolution du système hybride pour voir en quoi il peut contribuer à l'amélioration de la sûreté de fonctionnement. Les défaillances des composants dans le système sont en effet des événements non observables et partiellement contrôlables. Nous avons donc cherché à montrer en quoi la théorie de la supervision peut être un moyen d'aide à la conception de systèmes sûrs en postulant qu'une modification du système (introduction de barrières…) agirait comme un superviseur et qu'il suffirait de simuler le comportement du système ainsi « contrôlé » sans avoir besoin de reconstruire un modèle de simulation complet du système modifié.

Chapitre 1

Fiabilité dynamique

Nous pouvons mentionner au moins trois raisons différentes justifiant l'importance de la fiabilité :

1. elle a un impact direct sur la sécurité : la vie humaine est en jeu,
2. elle est un facteur d'aide à la décision vis-à-vis des enjeux économiques,
3. la complexité des systèmes accroît la probabilité de leurs défaillances et rend impératives les études de fiabilité.

1.1 Définitions

Avant de décrire l'approche que nous avons développée, il est évidemment essentiel de bien cerner les principaux concepts que nous allons utiliser.

1.1.1 Sûreté de fonctionnement

Les préoccupations dites de sécurité sont très présentes dans le monde des machines outils, des transports ou dans la pétrochimie. Dans les installations de production manufacturière ou batch, les préoccupations sont plutôt liées à la disponibilité. Dès lors que la sécurité ou la disponibilité d'un système est mise en défaut, on incrimine sa fiabilité. Enfin, en cas de dysfonctionnement, il convient de remettre le système en conditions de fonctionnement initial : c'est là qu'intervient la maintenabilité. Ces quatre caractéristiques constituent la Sûreté de Fonctionnement (SdF) d'un dispositif.

Un des grands mérites du concept de SdF est l'intégration des méthodes et techniques

destinées à garantir l'aptitude d'un système à délivrer un service dans lequel on puisse avoir confiance et à s'assurer que cette confiance est justifiée.

La SdF est définie par [Villemeur, 1988] comme la science des défaillances. Au sens plus strict, la SdF est l'aptitude d'une entité à assumer une ou plusieurs fonctions requises dans des conditions données [CEI 50 (191), 1990].

Selon [Laprie *et al.*, 1995] la SdF d'un système est la propriété qui permet à ses utilisateurs de placer une confiance justifiée dans le service qu'il leur délivre. Cette notion de confiance est fondamentale, étant donné que tout système matériel/logiciel contient des erreurs, la grande majorité d'entre elles étant des erreurs introduites lors des phases de conception. Son objectif est alors de spécifier, concevoir réaliser et exploiter des systèmes où la faute est naturelle, prévue et tolérable. Une définition alternative donné par [Laprie, 2004] de la SdF est l'aptitude à éviter des défaillances du service délivré plus fréquentes ou plus graves qu'acceptable.

Pour pratiquer la SdF, il est important, d'abord, d'identifier les sources de défaillances et de manière aussi exhaustive que possible. Ensuite, pour chacune d'elles, il conviendra d'en évaluer l'importance par rapport aux autres ou avec une échelle de mesure absolue (en calculant une probabilité d'apparition). En plus, prévoir les défaillances est aussi un objectif primordial. Pour cela, on doit observer et utiliser des modèles d'évolution. D'ailleurs, à toute observation d'une défaillance, on associera des mesures (statistiques – rendement) afin d'enrichir les modèles utilisés pour l'évaluation et la prévision. Finalement, l'ultime objectif est de maîtriser les défaillances par la réduction de leur occurrence, la prévention contre les conséquences ou par leur tolérance.

Par ailleurs, [Laprie *et al.*, 1995] définissent l'arbre de la SdF (figure 1.1). Cet arbre, montre les entraves, les moyens et les attributs associés à la SdF. Les entraves sont liées aux circonstances indésirables mais non inattendues. Les moyens correspondent aux méthodes et techniques permettant de garantir l'aptitude du système à délivrer un service conforme à l'accomplissement de sa fonction et de donner confiance dans cette aptitude. Finalement, les attributs permettent d'exprimer les propriétés qui sont attendues du système et d'apprécier la qualité du service délivré.

Au sens de la norme CEI 50 191 [CEI 50 (191], la sûreté de fonctionnement recouvre les concepts de fiabilité, maintenabilité et disponibilité (ou FMD). L'équivalent Anglo-Saxons est le terme dependability, (reliability, maintainability, availability) souvent désigné par l'acronyme RAM. La sécurité est souvent traitée à part. Cependant, l'acronyme RAMS (FMDS en français) est utilisé pour désigner l'ensemble des activités liées à ces quatre concepts.

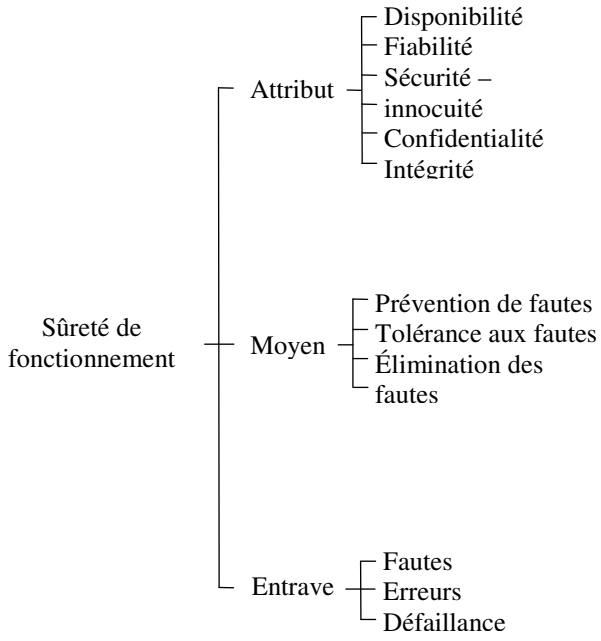


Figure 1.1 – L'arbre de la Sûreté de Fonctionnement [Laprie *et al.*, 1995].

1.1.2 Fiabilité

Dès que les hommes ont inventé les premiers instruments, ils sont devenus dépendants de leur bon fonctionnement. Dans ce sens, le concept de fiabilité était né. Avec l'entrée de l'électronique la fiabilité est entrée dans une nouvelle ère. Cependant, la fiabilité comme un sujet d'étude systématique a débuté dans les années soixante.

La fiabilité est un concept populaire qui a été utilisé pendant des années comme un attribut louable d'une personne ou un d'un objet façonné. L'Oxford English Dictionary définit la fiabilité comme la qualité d'une entité à être fiable ; sur laquelle on peut compter à un instant donné; dans laquelle la confiance peut être mise. En anglais « reliability » vient de « to rely on » signifiant « compter sur, avoir confiance en… » alors que fiabilité en français vient effectivement du mot « fiable », c'est-à-dire en qui on peut se fier.

En 1962, l'Académie de Sciences, l'a définie de la façon suivante : Grandeur caractérisant la sécurité du fonctionnement, ou mesure de la probabilité de fonctionnement d'un appareillage selon les normes prescrites. Plus tard, dans les années soixante dix, le Comité Électrotechnique International propose la définition suivante : Caractéristique d'un dispositif, exprimée par la fiabilité, qu'il accomplisse une fonction requise, dans des conditions données, pendant une durée donnée [Pagès et Gondran, 1980].

[Laprie *et al.*, 1995] définissent la fiabilité comme la mesure de la continuité de la délivrance d'un service correct ou de façon équivalente, mesure aussi du temps jusqu'à défaillance.

[CEI 50 (191), 1990] et [Villemeur, 1988] expriment que la fiabilité est l'aptitude d'une entité à accomplir une fonction requise dans des conditions données, pendant une durée donnée. Cette aptitude se mesure [Smith, 2001] par la probabilité qu'une entité réalise une fonction requise dans des conditions données pendant une période de temps donnée. La fiabilité peut être paraphrasée comme la probabilité de la non défaillance de l'entité dans un période de temps donnée.

A l'instant t, la fiabilité se mesure alors par la probabilité que l'entité E accomplisse une fonction requise dans les conditions données pendant l'intervalle de temps [*0; t*] [CEI 50 (191), 1990]. Ainsi,

$$R(t) = P[\text{E soit non défaillante sur } [0;t]] \tag{1.1}$$

ou

$$R(t) = P[\text{E soit non défaillante sur } [t_1,t_2]] \tag{1.2}$$

L'aptitude contraire est la probabilité de défaillance de l'entité, quelquefois appelée défiabilité. On écrit :

$$\overline{R} = 1 - R(t) \tag{1.3}$$

L'évaluation de cette probabilité peut être faite différemment selon la nature des entités considérées ou selon les moyen dont on dispose pour le faire.

- Fiabilité opérationnelle : (observée) résulte de l'observation et de l'analyse du comportement d'entités identiques dans des conditions opérationnelles.

- Fiabilité extrapolée : qui résulte d'une extension (par extrapolation définie ou par interpolation) de la fiabilité opérationnelle à des durées ou des conditions différentes.

- Fiabilité prévisionnelle : (prédite) qui estime une fiabilité future à partir de considérations sur la conception des systèmes et la fiabilité de leurs composants.

- Fiabilité intrinsèque : mesurée au cours d'essais spécifiques effectués dans le cadre d'un programme d'essais entièrement défini.

1.1.3 Disponibilité

C'est l'aptitude d'une entité à être en état d'accomplir une fonction requise dans des conditions données et à un instant donné [Villemeur, 1988].

La disponibilité est généralement mesurée par la probabilité qu'une entité E soit en état d'accomplir une fonction requise dans des conditions données à l'instant *t*.

$$A(t) = P[\text{E non défaillante à l'instant } t] \tag{1.4}$$

Cette caractéristique est appelée disponibilité instantanée. L'aptitude contraire sera dénommée indisponibilité ; sa mesure est notée $\bar{A}(t)$:

$$\bar{A}(t) = 1 - A(t) \tag{1.5}$$

La disponibilité ainsi définie ne fait pas appel à l'histoire de l'entité, qu'elle ait été ou non réparée une ou plusieurs fois avant l'instant t (c'est en quelque sorte une probabilité non conditionnelle). Il est donc évident que pour un système non réparable, la disponibilité est égale à la fiabilité et que d'une façon générale $A(t) \geq R(t)$. La disponibilité peut se décliner en termes de fiabilité et maintenabilité [Smith, 2001].

1.1.4 Maintenabilité

C'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 [Villemeur, 1988].

La maintenabilité est généralement mesurée par la probabilité que la maintenance d'une entité E accomplie dans des conditions données, avec des procédures et des moyens prescrits, soit achevée au temps t, sachant que l'entité est défaillante au temps $t = 0$. L'évaluation de cette probabilité est bien sûr liée à la manière dont est effectuée la remise en état de fonctionnement de l'entité.

1.1.5 Sécurité

Bien que la norme [CEI 50 (191), 1990] n'intègre pas la sécurité comme composant de la SdF, nous considérons qu'il est important de la prendre en compte car l'occurrence d'un événement catastrophique met en péril la vie humaine.

En fait, le concept de sécurité est probablement le plus difficile à définir et à évaluer, car il englobe des aspects très divers. Cependant, la norme [EN 292 – 1, 1991] sur la sécurité des machines donne cette définition :

Aptitude d'une machine à accomplir sa fonction, à être transportée, installée, mise au point, entretenue, démontée et mise au rebut dans les conditions d'utilisation normales spécifiées dans la notice d'instructions, sans causer de lésions ou d'atteinte à la santé.

La sécurité peut également s'exprimer sous forme d'une probabilité : probabilité que le système évite de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques [Villemeur, 1988]. Si on considère que les défaillances d'un système se partagent en deux catégories, celles qui sont dangereuses et celles qui ne le sont pas, la sécurité peut être considérée comme la part de la fiabilité relative aux défaillances

dangereuses. Ce concept peut devenir prépondérant dans une analyse de SdF, dans la mesure où une défaillance du système peut présenter un risque de dommage corporel à l'encontre des usagers.

1.1.6 Défaillance

C'est un événement. Un événement est présent ou non, il peut se combiner avec d'autres événements pour produire des événements composés. La défaillance d'une entité est la conséquence de l'imperfection de celle-ci (imperfection intrinsèque ou due à celle de ses composants). Selon [CEI 50 (191), 1990] et [Villemeur, 1988] une défaillance est : la cessation de l'aptitude d'une entité à accomplir une fonction requise. La figure 1.2 montre la relation entre la défaillance et l'état d'un composant [Chevalier *et al.*, 2004].

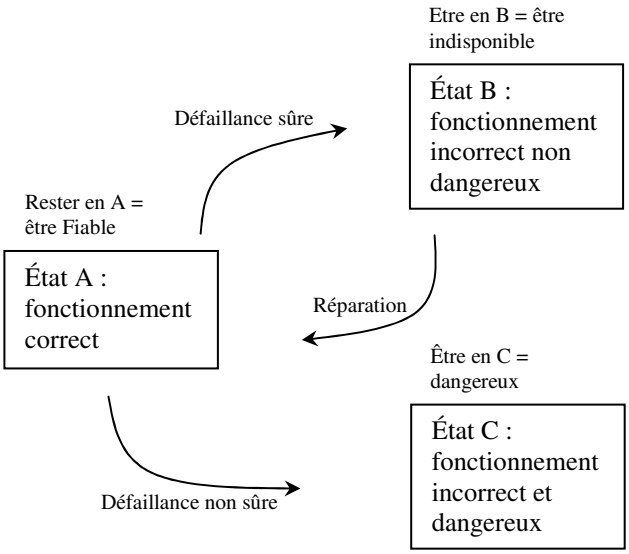


Figure 1.2 – Relation entre la défaillance et l'état d'un composant [Chevalier *et al.*, 2004].

Particulièrement, les composants électroniques sont soumis à un stress important de l'environnement qui constitue la principale cause de défaillance : humidité, vibration, température, champ électromagnétique, … La connaissance de la fiabilité ou bien encore du taux de défaillance des composants est primordiale pour pouvoir évaluer la SdF d'une architecture matérielle à l'aide de méthodes d'analyse quantitative. Le taux de défaillance peut être rapproché de probabilité pour que le composant soit défaillant à l'instant t+dt sachant qu'il n'est pas défaillant à l'instant t. On le note λ et il s'exprime à partir de la fiabilité par : $\lambda(t) = -\frac{d}{dt}(LogR(t))$

Cependant, les taux de défaillance 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 influants dans les études. Il existe des modélisations de taux de défaillance [Villemeur, 1988] 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 peut se décomposer de la façon suivante [Schoenig, 2004] :

$$\lambda = \lambda_b \pi_E \pi_A \pi_Q \pi_n \tag{1.6}$$

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).

Le document [MIL-HDBK-217F, 1991] est une des sources de taux de défaillance des composants électroniques. Le modèle du taux de défaillance (λ_p) d'un composant semi conducteur y est donné par :

$$\lambda_p = \pi_Q \left(C_1 \pi_t \pi_V + C_2 \pi_E\right) \pi_L \text{ défaillances/}10^6 \text{ heures} \tag{1.7}$$

- π_Q est le coefficient de qualité,

- π_t est le coefficient de température de jonction,

- π_V est le coefficient de la tension appliquée,

- π_E est un coefficient amplificateur pour l'environnement,

- π_L est un coefficient tenant compte de la maturité du processus de fabrication,

- C_1 est basé sur la quantité de transistors dans le composant,

- C_2 est associé au type de boitier du composant.

1.1.7 Les temps caractéristiques pour la Sûreté de Fonctionnement

Les différents temps caractérisant la SdF se définissent en fonction de leur état de fonctionnement : avant défaillance, entre défaillance, entre défaillance et réparation, etc. Ces temps dépendent des probabilités d'occurrences des divers événements comme les défaillances et les réparations des composants. Ce sont des variables aléatoires que l'on chercher à caractériser par leurs espérances mathématiques.

[Villemeur, 1988] et [Lannoy, 1996] présentent quelques temps caractérisant de la SdF. De même, ils présentent aussi, au moyen d'un graphique, la relation existant entre eux (figure 1.3). Cependant, la norme [CEI 50 (191), 1990] présentent les définitions d'une façon plus fines par rapport aux autres auteurs :

- **MTTF** (mean time to failure) : durée moyenne de fonctionnement avant défaillance, espérance mathématique de la durée de fonctionnement avant défaillance. La définition du MTTF est :

$$MTTF = \int_0^\infty R(t)dt \tag{1.8}$$

- **MTBF** (mean time between failures) durée moyenne entre deux défaillances consécutives d'une entité réparée.

- **MRT** (mean repair time) durée moyenne de réparation, espérance mathématique de la durée du temps de réparation.

- **MTTR** (mean time to repair or restoration) durée moyenne de panne ou moyenne des temps pour la remise en état de fonctionnement, espérance mathématique de la durée de panne. MTTRep est associé à la réparation du composant et MTTRes à sa restauration. La différence entre les deux est liée au fait que l'on considère ou non le temps mis pour remettre en service l'équipement, le MTTRes l'incluant.

- **MUT** (mean up time) ou TMD temps moyen de disponibilité, espérance mathématique de la durée de disponibilité.

- **MDT** (mean down time) ou TMI temps moyen d'indisponibilité, espérance mathématique de la durée d'indisponibilité.

Le MDT est décomposé en plusieurs phases lesquelles sont montrées par la figure 1.3.

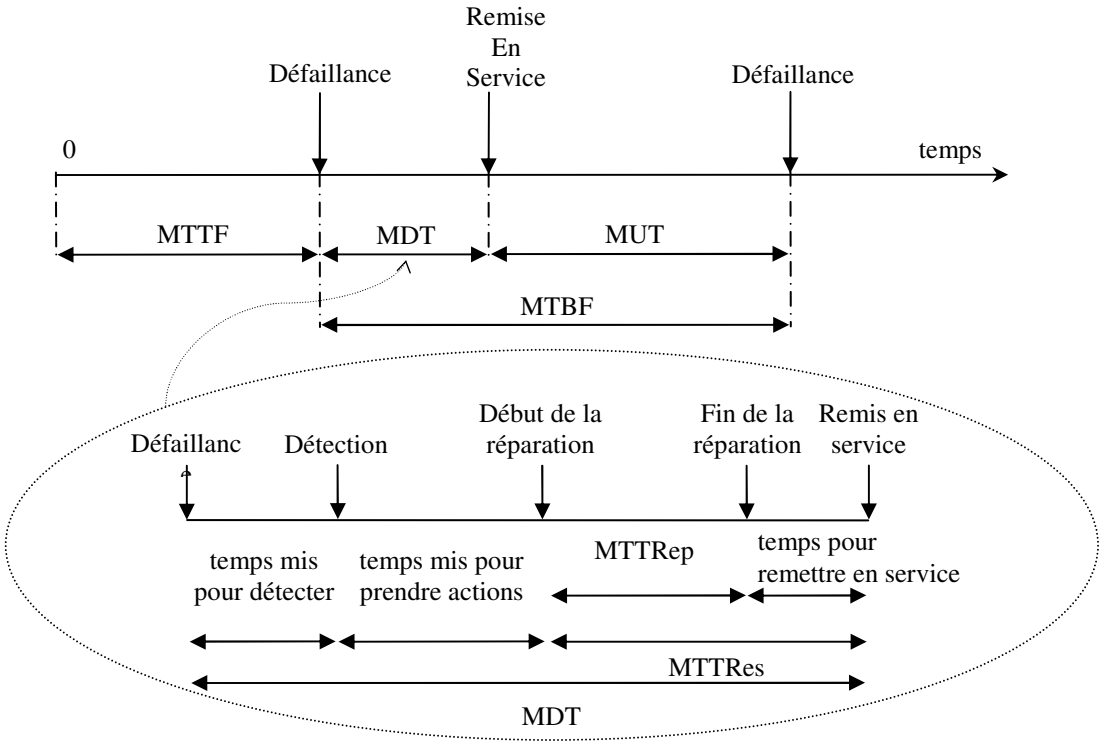


Figure 1.3 – Représentation du MTTF, du MUT, du MDT et du MTBF.

1.2 Les méthodes pour l'évaluation de la fiabilité prévisionnelle

Un premier type de méthodes utilisées en théorie de la fiabilité prévisionnelle regroupe les approches combinatoires. Elles sont utilisées pour identifier et évaluer les combinaisons de défaillances de composants pour lesquelles le système est défaillant. Dans ce groupe se classent les arbres de défaillances, les arbres d'événements, les diagrammes de fiabilité et la fonction de structure au sens de [Kaufman *et al.*, 1975].

Un deuxième type de méthode repose sur une représentation d'état du système, dont les transitions correspondent à une défaillance ou à une réparation d'un composant. Sous certaines conditions, ces modèles sont Markoviens (ou semi Markoviens) et permettent d'accéder à la probabilité de séjour dans chacun des états..La probabilité d'être dans un quelconque des états de fonctionnement est la disponibilité du système. Pour évaluer la fiabilité, il convient de modifier le modèle en rendant les états de panne absorbants (pas de réparations).

1.2.1 L'arbre d'événements et l'arbre des causes

L'analyse par arbre de causes ou de défaillances est une méthode de type déductif. Il s'agit de diagrammes logiques utilisant une structure arborescente pour représenter les causes de défaillances et leurs combinaisons conduisant à un événement redouté (racine de

l'arbre). La réduction des arbres de faute à partir du calcul des coupes minimales, permet d'identifier les chemins critiques. On en déduit les éléments matériel et logiciel du système dont la défaillance contribue le plus à la réalisation de l'événement redouté.

L'arbre d'événements est une technique d'analyse inductif qui est basée sur une logique binaire. Cette technique permet de décrire, de façon graphique, comment un événement initial se propage à travers un système, en évaluant systématiquement tous les chemins fortuits et de succès qui peuvent résulter. L'analyse débute avec un événement initial et il continue avec l'étude de ses conséquences possibles qui sont déterminées par l'action de succès ou de défaillance des dispositifs prévus pour empêcher la propagation. L'événement initial est une défaillance qui affecte une ou plusieurs des fonctions essentielles de l'installation et qui requiert l'intervention des systèmes de sécurité ou barrières pour la contrôler. L'événement initial peut être interne, par exemple la défaillance d'un composant du système ; ou externe, comme l'action humaine. Ces dernières peuvent être naturelles ou anthropogéniques, comme la chute d'un avion sur une installation [Villemeur, 1988].

Les arbres des causes peuvent mener à des évaluations quantitatives de la probabilité de l'événement sommet (ou racine) qui représente la défiabilité du système lorsque cet événement est la défaillance du système non réparable. Dans certains cas, il est aussi possible de calculer l'indisponibilité. La méthode de l'arbre de causes consiste à rechercher toutes les combinaisons possibles d'événements entraînant la réalisation de l'événement non désiré (ou dit aussi redouté) et représenter graphiquement ces combinaisons au moyen d'une structure arborescente dont l'événement non désiré est le sommet (ou racine).

1.2.2 Le diagramme de fiabilité

C'est la méthode la plus anciennement connue pour le calcul de la fiabilité des systèmes non réparables. Bien qu'elle puisse aussi s'appliquer aux systèmes réparables, son usage y reste limité. Le diagramme de fiabilité représente les conditions de réalisation de la fonction d'un système composé de sous systèmes caractérisés par leur fiabilité. On appelle parfois chemin de succès, un chemin permettant d'aller de l'extrémité gauche à l'extrémité droite du diagramme, symbolisant ainsi que la fonction du système est réalisée. C'est par analogie avec les circuits électriques que cette méthode a été proposée. La perte d'un composant dans un circuit électrique en série empêche le passage du courant. Par analogie, la défaillance d'un composant interrompt le chemin de succès [Villemeur, 1988].

1.2.3 La fonction de structure d'un système

L'étude d'un système par la fonction de structure permet de décrire de manière analytique comment l'état d'un système dépend de l'état de ses éléments. Les hypothèses sont les suivantes :

1. le système est à état binaire : il fonctionne ou ne fonctionne pas (en panne),

2. les composants du système : ils sont aussi à état binaire et l'état du système ne dépend que de l'état des composants.

Kaufmann, Gronchko et Cruon [Kaufmman et *al.,* 1975] ont défini une fonction à valeurs dans le sous ensemble des entiers {0,1} muni des opérations d'addition soustraction et multiplication sur les entiers et montré qu'elle peut se mettre sous une forme réduite à partir de laquelle on obtient la fiabilité du système.

On peut conclure que les méthodes combinatoires ne permettent que d'identifier et d'évaluer les combinaisons d'événements qui conduisent à l'occurrence d'un événement, redouté susceptible d'apparaître dans le système étudié. Ces combinaisons ou coupes, à la différence des séquences accidentelles, ne prennent pas en compte l'ordre d'apparition des événements qui la constituent. En fait, ces méthodes combinatoires ne s'intéressent pas aux instants d'occurrence des événements mais simplement au fait qu'elles ont eu lieu. Pour pallier ces carences, comme nous le verrons, différentes approches ont été proposées dans le passé et ont donné, lors de différentes applications, des résultats intéressants [Siu, 1994], [Cojazzi et Cacciabue, 1993], [Smidts et Devooght, 1992b], [Smidts, 1994], [Marseguerra et Zio, 1995], [Marseguerra et Zio, 1996] et [Tombuyses et Aldemir, 1996].

1.2.4 Les modèles Markoviens

Les modèles Markoviens représentent une classe de processus stochastiques. Un processus stochastique décrit l'évolution d'un système par les probabilités qu'il se trouve à un instant donné dans un quelconque de ses états (ou sous ensemble d'états) possibles. Un processus markovien est un processus stochastique dont l'état futur ne dépend pas de la trajectoire passée. Il est homogène lorsque les taux de transition entre états ne dépendent pas du temps. Lorsque le processus est défini de manière continue dans le temps on le représente par un graphe d'état dit de Markov. Lorsque le processus n'est décrit qu'à certains instants discrets, on parle de chaîne de Markov. Par abus de langage, ce dernier terme est parfois utilisé pour les modèles à temps continu. Ce sont ces derniers qui sont utilisés pour évaluer de façon quantitative la SdF des systèmes, notamment lorsque les taux de transition sont constants, c'est-à-dire que les instants de défaillance et de réparation des composants sont distribués selon des lois exponentielles [Cocozza-Thivent, 1997].

On suppose que le passage d'un état du système à l'autre survient aléatoirement par la défaillance d'un composant ou par la réparation d'un autre élément. Connaissant l'état initial du système, on peut en déduire soit la probabilité d'être dans un état donné après une durée déterminée, soit la probabilité moyenne d'être dans un état donné tout au long de sa durée de vie utile.

1.2.5 Les réseaux de Petri

Les réseaux de Petri sont également des modèles état-transition mais plus riches que les automates d'états car ils peuvent modéliser une classe de langage plus grande [Cassandras et Lafortune, 1999]. Leur intérêt est qu'ils permettent d'exprimer de manière

aisée les mécanismes de parallélisme, de synchronisation, de partage ou d'assemblage de ressources, grâce au concept de marquage L'intérêt est de pouvoir modéliser le comportement du système sans connaître a priori l'ensemble de ses états.

Ils peuvent être utilisés à la fois [David et Alla, 1989] pour des analyses de type qualitatif (vérification de propriétés) ou pour des analyses de type quantitatif, comme l'évaluation de performances fonctionnelles ou de SdF.

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 de marques forme le marquage. Le marquage définit à un instant donné l'état du système. L'ensemble de transitions représente quant à lui l'ensemble des événements dont les occurrences provoquent des modifications sur l'état du système. Chaque marquage peut être représenté par un vecteur et le réseau peut être modélisé comme un ensemble d'équations algébriques L'état du système est alors décrit par son graphe de marquage

Il existe plusieurs classes de réseaux de Petri. Les réseaux de Petri stochastiques associent à chaque transition un taux de franchissement aléatoire. Si tous les taux sont constants (distribution exponentielle de la durée de franchissement) alors le graphe de marquage est homogène à un graphe de Markov.

Très souvent cette hypothèse ne peut être satisfaite en raison de la complexité des phénomènes à modéliser. On utilise alors les modèles de type état transition (graphe d'état ou RdP) comme supports de simulation par des méthodes de Monte Carlo.

1.2.6 Simulation de Monte Carlo

La simulation de MC est un outil numérique basé sur le tirage au sort de nombres aléatoires. La quantité que l'on désire estimer correspond à l'espérance mathématique d'une variable aléatoire, évoluant selon un processus stochastique. L'estimation est obtenue en moyennant les résultats collectés lors d'un grand nombre d'histoires simulées du système. Afin de voir apparaître un événement redouté (rare en général) un nombre suffisant de fois, on doit faire un grand nombre de simulations, ce qui implique des temps de calcul important [Veach, 1997], [Dubi, 2000] et [Niel et Craye, 2002]. 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, en favorisant l'apparition des événements rares. Toutefois, ces méthodes ne sont pas toujours faciles à mettre en œuvre, car elles impliquent des hypothèses assez fortes et/ou ne fournissent pas forcement des estimateurs de qualité [Dubi, 2000].

La simulation de MC se prête particulièrement bien aux études de fiabilité dynamique étant donné le caractère stochastique naturel du problème étudié. Selon [Labeau *et al.*, 2000], [Smidts et Devooght, 1992], [Marseguerra *et al.*, 1998] et [Cabarbaye et Laulheret, 2005], cet outil de résolution s'avère peu sensible aux dimensions du problème, ce qui est un avantage précieux.

On peut effectuer une simulation de MC à partir d'un grand nombre de modèles comportementaux (automates d'états, réseaux de Petri, arbres de défaillance…) pour la résolution d'un problème de fiabilité prévisionnelle.

1.3 Fiabilité dynamique

[Kermisch et Labeau, 2000a] ont défini la fiabilité dynamique comme étant la partie des analyses probabilistes de sûreté qui étudie de manière intégrée le comportement des systèmes homme-machine-logiciel affectés par une évolution dynamique sous-jacente. Ils apportent aussi une généralisation de la définition : la partie de la SdF qui étudie de manière intégrée le comportement des systèmes industriels complexes affectés par une évolution dynamique continue sous-jacente.

Plus tard, [Cabarbaye et Laulheret, 2005] ont exprimé plus spécifiquement que la fiabilité dynamique recouvre l'évolution de systèmes dont le modèle de fiabilité évolue dans le temps en réponse à des événements aléatoires (une chaîne en redondance froide qui passe à l'état ON lors la défaillance de la chaîne nominale, par exemple) ou à des changements liés au franchissement de seuils par certaines variables continues inhérentes au systèmes (un mécanisme de contrôle activé dès l'entrée d'un paramètre en zone d'alerte, par exemple) ou à l'action humaine (par exemple, actions de contrôle, de pilotage ou de reconfiguration).

Lorsqu'on veut décrire l'interaction entre le comportement du matériel considéré et son environnement, on décrit cet environnement par un ensemble de variables continues, appelées variables physiques. Les taux de transition instantanés entre les différents états du matériel dépendent alors de ces variables physiques et inversement, l'évolution au cours du temps des variables physiques est décrite par un système d'équations différentielles dont les coefficients dépendent de l'état du matériel. C'est ce que [Cocozza-Thivent et Eymard, 2006a) appellent un modèle de fiabilité dynamique.

A partir de ces définitions, on peut remarquer que l'expression fiabilité dynamique de systèmes hybrides met en évidence la présence de deux constituants de nature différente : un état continu du système dépendant des variables physiques continues (fonctionnelles ou dysfonctionnelles) décrivant la dynamique et d'un état discret dépendant des états discrets de ses constituants. La fiabilité dynamique accentue aussi le fait que tels systèmes évoluent dynamiquement et que les défaillances (et réparations) peuvent influencer les dynamiques et réciproquement, que les dynamiques (et ces variables d'état associées) peuvent affecter les taux de défaillance et de réparation.

Ces définitions sont données d'un point de vue de la nature des systèmes à traiter. En considérant que la fiabilité dynamique est une discipline qui appartient à la SdF et qu'elle a eu pour objectif dès ses premiers balbutiements de proposer une approche intégrée tant des études probabilistes de sûreté que des études de fiabilité et disponibilité des installations industrielles, nous proposons la définition suivante :

La fiabilité dynamique est l'évaluation prévisionnelle de la fiabilité d'un système dont la structure fiabiliste, ce qui exprime comment la défaillance du système dépend des défaillances de ses composants, évolue dynamiquement dans le temps. On peut donc dire en général que la fiabilité dynamique est le problème de l'évaluation probabiliste de la défaillance d'un système dynamique hybride (système décrit par un ensemble de variables continues et un ensemble d'états discrets interagissant).

Conséquence sur la fiabilité ou tout paramètre de SdF :

$$Q_S(t) = P\left[S\left(X(T), \mathcal{X}(T), V(T)\right) \subset S_Q \right]_{0 \le T \le t} \tag{1.9}$$

Cette expression exprime que tout paramètre Q de la SdF à un instant t donné, est égal à la probabilité que l'état S du système dépendant des variables continues $X(T)$, discrètes $\mathcal{X}(T)$ et aléatoires $V(T)$, appartienne au sous ensemble des états caractérisant le paramètre Q.

Dans la plupart des cas, en dehors de l'hypothèse de lois exponentielle pour les défaillances des composants, les fiabilistes ont recours à la simulation (Monte Carlo) pour déterminer la fiabilité du système. Pour les systèmes « dynamiques » cette technique doit être « mélangée » avec la résolution des équations régissant l'évolution des variables continues afin de détecter les franchissements éventuels de seuils entraînant des changements d'état discret. Cette problématique est très proche de celle posée par la simulation des systèmes dynamiques hybrides.

En pratique, il est assez répandu d'évoquer le concept de fiabilité dynamique pour décrire l'ensemble des activités d'évaluation prévisionnelle de la sûreté de fonctionnement d'un système dans un contexte dynamique hybride. C'est le sens que nous lui donnerons en général dans ce document.

1.4 Les problèmes posés par la fiabilité dynamique

Comme nous l'avons mentionné, le concept de fiabilité dynamique a pour objet de prendre en compte les interactions entre le comportement fonctionnel dynamique et déterministe d'un système et le comportement dysfonctionnel stochastique de ses composants. Nous allons citer les principaux problèmes posés par l'évaluation de la fiabilité dynamique :

- prise en compte des interactions dynamiques entre les paramètres physiques et le comportement nominal ou dysfonctionnel des composants,
- prise en compte du temps (lois non exponentielles, usure, fatigue, lois discrètes…),
- prise en compte de l'ordre d'occurrence des événements (raisonner en termes de séquences possibles d'événements et non plus en termes de coupes),

- prise en compte des modes de vieillissement multiples des composants selon l'état discret du système,

- prise en compte du caractère déterministe ou stochastique des événements et de variables physiques (changements de mode non dus à des défaillances,…),

- prise en compte du temps de détection des défaillances et de son efficacité (latence, probabilité, fausse détection…).

Cette problématique implique que la fiabilité dynamique doit prendre en compte :

- les composants du système. Les dépendances fonctionnelles pouvant exister entre les différents composants d'un système doivent pouvoir être directement incluses dans l'approche méthodologique des problèmes,

- l'évolution dynamique de l'ensemble des processus physiques associés au comportement du système, dans ses modes opératoires comme dans ses états dégradés ou dysfonctionnels, joue un rôle primordial. Cette évolution dynamique est le vecteur par lequel s'expriment certaines interactions fonctionnelles entre les parties du système. Ainsi, les valeurs prises au cours du temps par les variables physiques (débit, température, pression, par exemple) décrivant la dynamique du système vont influencer les changements d'état que l'installation est susceptible de connaître,

- les paramètres de la fiabilité sont modifiés par la variation des grandeurs physiques. En retour, les changements d'états subis par le système affectent directement les lois d'évolution dynamique de l'installation.

Pour traiter ces problèmes, les outils traditionnels de la fiabilité prévisionnelle ne peuvent pas être appliqués efficacement car ils supposent, en général, une structure invariante dans le temps pour le système. D'un autre côté, l'ensemble de données décrivant le processus physique est potentiellement très grand (variables discrètes et continues). Une formulation mathématique rigoureuse de la fiabilité dynamique implique 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. La transcription mathématique aboutit à un système d'équations différentielles très complexe. Résoudre alors le problème de l'évaluation de la fiabilité dynamique implique l'usage d'une technique numérique. Plusieurs approches différentes ont été proposées.

1.5 Les approches

1.5.1 Introduction

Les premières approches à avoir invoqué des processus dynamiques à transitions stochastiques ont été désignées sous les noms de « Dynamique Probabiliste » ou « Théorie

des Arbres d'Événements Continus ». La fiabilité dynamique elle-même a aussi été qualifiée de cette façon [Labeau *et al.*, 2000], [Siu, 1994] et [Becker *et al.*, 2002].

L'interrelation existant entre les variables physiques et les changements d'états du système, au cours d'un fonctionnement transitoire peut être représentée par le schéma de la figure 1.4.

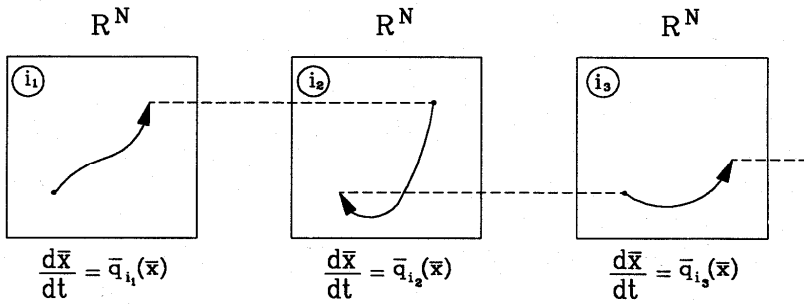


Figure 1.4 – Trajectoires des variables d'état continues d'un système soumis à des changements d'état discrets [Devooght, 1997].

Le schéma de gauche dans la figure 1.4 représente graphiquement l'évolution que suit le système après la réalisation de l'évènement lui faisant quitter son état stationnaire pour rentrer dans l'état i_1. Dans celui-ci, les variables physiques, représentées par le vecteur $\bar{X}$, obéissent aux lois déterministes de la dynamique correspondante, donné par $d\bar{X} / dt = \bar{q}_i(\bar{X})$. Au bout d'un certain intervalle de temps, le système subit une transition entre les états i_1 et i_2, soit parce qu'un seuil associé à une action de contrôle a été atteint, soit parce qu'une transition en opération s'est produite. Les variables continues décrivant le comportement de l'installation suivent alors de nouvelles lois dynamiques, correspondant au nouvel état i_2 (schéma central), et ce de manière déterministe jusqu'au prochain changement d'état, où la situation précédente se répète (schéma de droite).

Cette description permet de comprendre l'appellation de *dynamique probabiliste* donnée initialement à la fiabilité dynamique : le système évolue le long de sections déterministes de trajectoires dans l'espace $\mathbb{R}^N$ des variables physiques, mais ces différentes sections sont connectées par des transitions qui peuvent se produire dans certains cas à des instants aléatoires. L'expression *processus stochastique déterministe par morceaux (piecewise deterministic stochastic processes)* est un autre nom plus mathématique que l'on peut donner à ce type de phénomènes.

Ces processus évolutifs ne pouvant être représentés dans les modèles classiques de la fiabilité, des nouvelles méthodes ont été implémentées. Elles modélisent explicitement l'évolution des variables physiques et l'influence mutuelle entre le comportement stochastique (taux de transition) et le comportement déterministe du système. Elles diffèrent essentiellement par les approximations qu'elles supposent [Smidts, 1994].

1.5.2 Les méthodes

On trouve dans la littérature plusieurs travaux qui présentent de façon générale les différentes méthodes et outils qui ont été développés pour traiter les problèmes de la fiabilité dynamique et comment ces méthodologies ont évolué et ont été adaptées pour résoudre quelques difficultés présentées.

[Siu, 1994] a résumé les limites que présentent les méthodes classiques des arbres de fautes et des arbres d'événements pour l'évaluation des risques et il montre les méthodologies alternatives développées. Il a présenté aussi une classification des méthodes employées en dynamique probabiliste selon qu'elles discrétisent le temps ou bien les variables continues du processus. Selon le type de modèle, les algorithmes spécifiques sont développés afin de pouvoir accéder à une évaluation quantitative par simulation de type Monte Carlo.

Plus tard, [Devooght, 1997] a présenté, dans le livre édité par [Lewins et Becker, 1997], une formulation mathématique de la problématique de la fiabilité dynamique, mis en évidence les besoins nécessaires pour faire face aux problèmes de résolution numérique. Il a ainsi proposé un cadre « unificateur » des approches existantes.

[Belhadj et Aldemir, 1996] et [Marseguerra *et al.*, 1998] ont proposé une classification des méthodes existantes à l'époque :

- Les modèles à transition d'état dont les modèles Markoviens. Ils représentent l'évolution du système en termes des différents états entre lesquels une transition est possible. Les états sont définis et basés sur les combinaisons des modes de fonctionnements des composants, les valeurs des variables du processus et l'action humaine. Le modèle donne la probabilité de trouver le système dans un état donné à un moment donné.

- L'approche arbres d'événements dynamiques. En suivant un événement initiateur, cette approche simule toutes les évolutions possibles du système dans un intervalle de temps spécifié. La simulation s'arrête quand un nombre d'intervalles de temps spécifié ou un événement sommet sont atteints.

- Les techniques de simulation directe. Elles ne consistent pas à identifier tous les scénarios a priori possibles mais elles utilisent la simulation de MC pour construire une série de réponses du système à un événement initiateur par échantillonnage du nombre infini de ramifications possibles.

Par ailleurs, [Kermisch et Labeau, 2000a] et [Kermisch et Labeau, 2000b] présentent une revue des différentes méthodes et outils pour traiter la fiabilité dynamique ainsi que les critères pour choisir une méthode d'analyse.

[Labeau *et al.*, 2000] présente, plus qu'une classification en soi-même, l'architecture d'un environnement d'analyse de la fiabilité dynamique qui devrait permettre d'intégrer tous les éléments intervenant dans l'évaluation de la fiabilité d'un système dynamique. Une description modulaire est proposée, dans laquelle on trouve les principaux éléments suivants :

- Un modèle de représentation des données d'entrée (les diagrammes de transition d'état, les réseaux de Petri stochastiques, les diagrammes de séquences d'événements, GO-FLOW, le graphe de flux dynamique DFM et l'approche orientée objet).

- Le moteur de calcul (lequel inclut la méthode de résolution elle-même, la génération des séquences et la simulation de l'évolution du système). Ce moteur agit comme un gestionnaire des interactions entre les modèles de la physique, du comportement de l'homme, du logiciel et enfin du matériel.

- L'affichage des résultats de sortie (qui prend en compte les scénarios de l'accident, la définition et l'estimation de leur fréquence, les niveaux de risque et d'accidents…) pour donner une estimation complète du problème.

Nous voulons faire un rappel et compléter l'état de l'art en fiabilité dynamique fait par ces auteurs. A l'heure actuelle plusieurs méthodes et outils informatiques, permettant d'évaluer la fiabilité d'un système dynamique, existent. Quelques-uns ont été présentés à travers des cas simples afin de montrer la faisabilité de l'approche. D'autres l'ont été dans des cas plus réalistes pour montrer leur efficacité et leur puissance.

La grande variété de ces méthodes ne rend pas facile leur regroupement. Néanmoins, nous proposerons la classification suivante :

1.5.2.1 Les méthodes analytiques

Si on désire traiter les problèmes relatifs à la fiabilité dynamique à travers un formalisme mathématique, cela nécessite d'intégrer dans le modèle les interactions entre les phénomènes probabilistes et le processus physique. [Devooght et Smidts, 1992] prennent en compte cet aspect sur la base des équations de Chapman – Kolmogorov et introduisent la notion d'arbres d'événements continus. Il s'agit de la densité de probabilité $\pi(\boldsymbol{x},i,t)$ de trouver le système au temps *t* dans l'état discret *i* où le vecteur $\boldsymbol{u}$ des variables physiques prend la valeur $\boldsymbol{x}$. La forme intégrale (1.10) est équivalente à un arbre d'événements où le branchement arrive de façon continue.

$$\begin{aligned}\pi(\boldsymbol{x},i,t) = \int \pi(\boldsymbol{u},i,0)\delta[\boldsymbol{x}-\boldsymbol{g}_i(t,\boldsymbol{u})]\times\exp\left\{-\int_0^t \lambda_i[\boldsymbol{g}_i(s,\boldsymbol{u})]ds\right\}d\boldsymbol{u} + \sum_{j\neq i}\int p(j\rightarrow i\mid \boldsymbol{u})d\boldsymbol{u} \\ \times\int_0^t \lambda[\boldsymbol{x}-\boldsymbol{g}_i(t-\pi,\boldsymbol{u})]\times\exp\left\{-\int_0^{t-\tau}\lambda_i[\boldsymbol{g}_i(s,\boldsymbol{u})]ds\right\}\times\pi(\boldsymbol{u},j,\tau)d\tau\end{aligned} \quad (1.10)$$

$\boldsymbol{g}_i(t,\boldsymbol{u})$ représente la trajectoire suivie par les variables physiques dans l'état discret *i* jusqu'à l'instant *t*. δ est la fonction de Dirac qui permet de ne retenir que les trajectoires menant à $\boldsymbol{x}$ à l'instant *t*. $\lambda_i[\boldsymbol{g}_i(s,\boldsymbol{u})]$ est le taux global de sortie de l'état *i* qui dépend des variables physiques (et donc de la trajectoire). $p(j\rightarrow i\mid \boldsymbol{u})$ est la probabilité de transition de l'état *j* vers l'état *i* au point $\boldsymbol{u}$. Cette expression est la somme de deux contributions : la première correspond au cas où le système est resté dans l'état *i* pendant l'intervalle [*0, t*].

La deuxième correspond aux cas où le système est passé d'un autre état *j* à l'état *i* à l'instant τ. Si on prend $\pi(\boldsymbol{u}, i, t) = \delta_{i1}\delta(\boldsymbol{u} - x_0)$, alors chaque terme de la série correspond à un développement de l'arbre d'événements selon ses différentes bifurcations.

[Smidts et Devooght, 1992] montrent que la théorie des arbres d'événements continus peut traiter des problèmes réels. La théorie standard des arbres d'événements assume que les points de branchements sont déterministes et que les branches représentent l'évolution transitoire. Dans la pratique les taux de défaillances peuvent être des fonctions complexes des variables physiques et les branchements peuvent arriver à n'importe quel moment. C'est ce qui justifie la nécessité des arbres d'événements continus. De même, [Becker *et al.*, 2002] font une extension de cette approche pour évaluer la fiabilité d'une structure soumise à différentes charges aléatoires. La structure peut être modélisée comme un système avec un vecteur d'états et une entrée aléatoire. Le vecteur d'état correspond aux paramètres physiques du système, tels que les caractéristiques géométriques et les propriétés du matériel. L'entré aléatoire pour le système est le processus de chargement stochastique sur la structure.

[Kermish et Labeau, 2000a] montrent que la généralisation des équations de Chapman - Kolmogorov amène à un système mathématique de taille considérable puisque la densité de probabilité dépend, dans chaque état, d'un grand nombre de variables (les variables physiques et le temps). Par conséquent, la résolution de (1.10) n'est possible qu'au niveau de cas-test simples.

1.5.2.2 Les méthodes semi-analytiques

Ces méthodes ont comme but [Devooght et Smidts, 1995], [Labeau et Devooght, 1995] de résoudre le système d'équation de Chapman Kolmogorov à l'aide d'une technique numérique classique. La solution proposée est d'obtenir des équations pour les densités de probabilité marginales à partir d'une interpolation de la densité multivariée. Cette forme interpolée est introduite dans les équations de Chapman Kolmogorov. Le résultat est intégré sur toutes les variables physiques excepté une, afin d'obtenir les équations des marginales. Cependant, des dépendances tout à fait générales des lois d'évolution dynamique et des taux de transition en les variables physiques ne permettent pas d'aller plus loin dans les déductions des équations des marginales. Si les techniques numériques classiques sont appliquées, on se trouvera confronté à un important problème de dimension.

Par ailleurs, l'approche processus markovien déterministes par morceaux (Piecewise Deterministic Markov Processes PDMP) a été présentée par [Dufour et Dutuit, 2002] afin de traiter la fiabilité dynamique : le système suit une trajectoire déterministe, décrite par exemple à travers d'une équation différentielle ordinaire, jusqu'à un premier temps de saut arrivant soit spontanément de manière aléatoire, soit quand la trajectoire atteint un seuil. A partir de cet instant, un nouveau point est sélectionné à travers un opérateur aléatoire et le processus repart de ce nouveau point. Entre deux sauts le système suit une trajectoire déterministe. Alors, il existe deux types de saut : déterministes, par exemple dus à un changement de mode de fonctionnement par le franchissement d'un seuil et stochastiques

modélisant les défaillances de composants ou les entrées qui modifient le mode de fonctionnement du système. Le point clé de cette approche est qu'elle peut naturellement prendre en compte les événements liés à l'évolution déterministe des paramètres physiques du processus et les événements stochastiques correspondants à la demande aléatoire ou des défaillances des composants du système. Les limites de cette approche sont l'utilisation de la loi exponentielle, la complexité mathématique pour décrire le changement d'état (les sauts).

[Zhang *et al.*, 2006] ont pour objectif d'illustrer la mise en œuvre de cette méthode. Ils montrent sur un cas test la puissance de modélisation des processus déterministe par morceaux et l'efficacité calculatoire de la simulation de Monte Carlo pour traiter certains problèmes relevant du champ de fiabilité dynamique. Le cas test est issu de l'industrie gazière. Le système comprend deux unités de production d'oxygène fonctionnant en parallèle dont les taux maximaux de production sont définis.

Dans le cadre d'un processus de Markov et pour un aspect très particulier de la fiabilité dynamique, [Desgrouas et Mercier, 2005] présentent une méthode permettant de conclure l'existence et l'unicité d'une loi stationnaire ainsi que la convergence du processus. C'est-à-dire, s'il y a une convergence vers une loi invariante que l'on peut calculer, les quantités asymptotiques (comme par exemple la disponibilité ou le MUT) pourront être déterminées à partir de cette loi invariante. Dans [Cocozza-Thivent *et al.*, 2006b] les auteurs présentent un algorithme numérique pour calculer les quantités asymptotiques d'un système complexe. Cet algorithme est basé sur l'approximation d'un processus modélisé à l'aide de la fiabilité dynamique, par un processus markovien dans un espace fini. L'algorithme consiste d'une part à construire la matrice génératrice de ce processus, d'autre part à résoudre un système linéaire qui lui est associé. Selon les auteurs, par rapport aux Méthodes de Monte Carlo, cet algorithme est une alternative déterministe, mais, en grandes dimensions, les logiciels de calcul tels que Scilab ou Matlab ne pourront plus résoudre directement le système d'équations.

1.5.2.3 Les méthodes de discrétisation

Ces méthodes discrétisent soit la variable temps, soit les variables du processus, soit toutes les deux. Ce type de méthode est le plus utilisé.

[Aldemir, 1987] a présenté une approche dynamique pour la modélisation de la défaillance des systèmes de contrôle de processus. Les systèmes de contrôle de processus sont des systèmes avec des boucles fermées régis dynamiquement par des variables d'état continues, comme la température, la pression, le niveau d'un liquide, etc. La méthodologie est basée sur la discrétisation de l'espace d'état et de l'espace du temps, les unités de contrôle (composants informatiques du système) ayant des états discrets. On définit un pas de temps Δt. Les unités de contrôle ne changent pas pendant l'intervalle $[t,t+\Delta t]$. Elles peuvent changer instantanément en $t+\Delta t$. Le comportement probabiliste du système est simulé par une chaîne de Markov. Les défaillances des unités de contrôle et leurs réparations sont mutuellement et statistiquement indépendants. La défaillance du système est décrite comme l'accès aux états absorbants. Les taux de défaillances et de réparations

des unités de contrôle sont constants. L'inconvénient principal de cette approche réside dans la matrice de transition, car elle produit une quantité considérable de zéros et cela entraîne un problème de stockage en mémoire de l'ordinateur.

[Marseguerra et Zio, 1995] présentent une approche de discrétisation pour une intégration approchée des équations qui décrivent l'évolution du processus dans une étude d'évaluation probabiliste de sûreté. L'approche consiste à remplacer l'espace continu des variables du processus par une grille de nœuds discrets, qui constitue la base d'une procédure d'interpolation appropriée à la description du point de fonctionnement de l'état continu. Cela permettrait d'améliorer la vitesse de l'intégration numérique. Dans une phase de pré-simulation les quantités intéressantes pour la simulation sont alors calculées pour tous les nœuds. La figure 1.5 montre un essai (histoire) de simulation de MC : alors que le système fonctionne sur le mode physique noté PM1, sur l'occurrence d'une transition à l'instant t1 le système prend une trajectoire différente. Cette transition en particulier peut se produire de façon aléatoire ou bien être due à un franchissement de seuil d'une des variables physiques. Comme conséquence au changement de configuration du système, celui-ci évolue maintenant d'après le modèle physique noté PM2. Une nouvelle transition stochastique déterminée par tirage devrait intervenir à t2. Cependant, l'évolution physique du système est telle qu'une action de contrôle (sécurité) est demandé à l'instant t*<t2. A ce point-là, l'essai peut prendre deux évolutions différentes: si le système de contrôle est défaillant sur cette demande, le système poursuit son évolution selon PM2 et le temps t2 de la prochaine transition est conservé. Dans ce cas, les variables du processus sont hors contrôle et le système est en danger. Dans le cas contraire (souhaité !) le système de contrôle intervient pour modifier la dynamique du système afin qu'il soit contrôlé et donc hors de danger. Dans ce dernier cas, le système suit la dynamique définie par le modèle PM3. Le temps t2 est oublié et le nouveau temps pour la prochaine transition est obtenu par un nouveau tirage aléatoire. L'essai continue de la même façon jusqu'à que le système devienne défaillant ou que le temps de mission soit atteint.

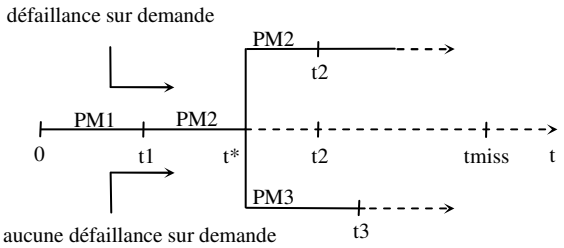


Figure 1.5 – Exemple de deux séquences possibles dans un essai de Monte Carlo [Marseguerra et Zio, 1995].

Dans le même ordre d'idées, [Tombuyses et Aldemir, 1997] présentent une approche Markovienne qui peut être utilisée pour l'analyse de fiabilité dynamique et l'analyse de systèmes de contrôle de processus. C'est une méthode de discrétisation nommée Continuous Cell-to-Cell Mapping Technique (CCCMT). Le principe de base qui sous-tend

cette méthode est le suivant : le CCCMT répartit les dynamiques du système en termes de taux de transition en un ensemble spécifique d'intervalles ou cellules sur les variables physiques, partitionnant ainsi l'espace d'état du système continu considéré.

Étant donné le système

$$\frac{dx_l}{dt} = f_{1,n}(x_1,...,x_L) \quad (l = 1,...,L) \text{ et } n = (1,...,N) \tag{1.11}$$

où $x_1(l = 1,...,L)$ sont les variables dynamiques, t est le temps et $f_{l,n}(x_1,...,x_l)$ sont les fonctions qui dépendent de la combinaison de l'état des composants $n = (n_1, n_2,..., n_C)$ où n_c désigne l'état du composant $c \in \{1,...,C\}$.

Le niveau et la température d'un liquide peuvent être les variables dynamiques du système. Les composants peuvent être une pompe, une vanne, etc. Chaque composant peut défaillir à un instant quelconque pendant son fonctionnement avec un taux de défaillance λ_c (bloqué dans l'état actif ou bloqué dans l'état de repos) ou β_c (ouverture intempestive, fermeture intempestive) aussi que la probabilité de la demande d_c.

La technique CCCMT partitionne les domaines des variables dynamiques en intervalles ou cellules :

$$V_j = V_{j_1,...,j_L} \equiv \left\{x_l : v_{l,j_l-1} \le x_l \le v_{l,j_l}; j_l = 1...J_l, l = 1,...,L\right\} \tag{1.12}$$

où $v_{l,0} \le x_1 \le v_{l,J_l}$ représente le domaine permis de la variable dynamique x_l, les limites correspondants aux événements redoutés.

La figure 1.6 montre une partition possible pour un système, incluant le rang d'opération normal, le rang d'opération autorisé et les événements sommets. Une possible partition de l'espace d'état pour les variables de processus (la température et le niveau d'un liquide, par exemple) est la suivante :

$$V_j \equiv V_{j_1,j_2}\,(j_1 = 1,...,9;\ j_2 = 1,...,16;\ j = 1,...,J_1J_2 = 9\times16 = 144 \text{ cellules.} \tag{1.13}$$

Les cellules V_{145} - V_{152} représentant les événements sommets (des états absorbants, par exemple le débordement, l'assèchement, la température maximum, la température minimum).

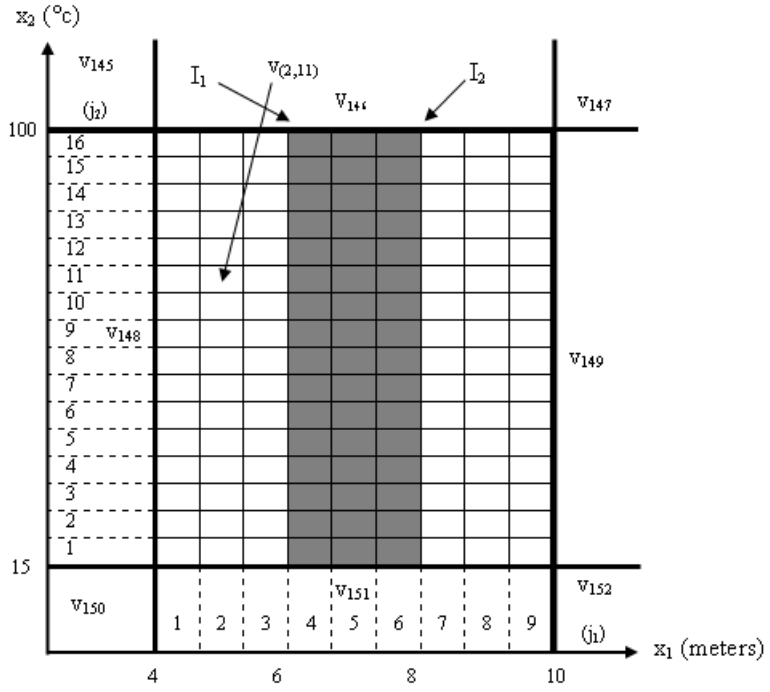


Figure 1.6 – Une partition possible de l'espace d'état à deux variables du processus [Tombuyses, 1997].

Si les probabilités ou taux de transition de l'état des composants (c'est à dire λ_c, β_c et d_c, par exemple) ne dépendent pas de l'histoire de l'évolution du système alors pour V_j dans le domaine de fonctionnement considéré, la probabilité $p_{n,j}(t)$ que $x=[x_1,...,x_L]\in V_j$ et que la combinaison de l'état des composants soit $n(n=1,...,N)$ au temps t peut être trouvée par la solution de

$$\frac{dp_{nj}(t)}{dt}=\sum_{e=1}^{E}\sum_{i=1}^{J}q_{mi}^{nj}p_{mi}(t) \tag{1.14}$$

où

$$q_{mi}^{nj}=\overline{q}_{n,m}^{H}(j)\delta_{ij}+\overline{h}(n\,|\,m,i\rightarrow j,t)q_{j,i}^{\phi}(m) \tag{1.15}$$

δ_{ij} est le symbole de Kronecker, $\overline{q}_{n,m}^{H}(j)$ est la valeur moyenne sur la cellule j du taux de transition de l'état des composants m à l'état des composants n, $q_{j,i}^{\phi}(m)$ est le taux de

transition des variables dynamiques de la cellule *i* à la cellule *j* pour l'état des composant *n* et $\overline{h}(n \mid m, i \to j, t)$ est la probabilité conditionnelle que la combinaison de l'état des composants soit *n* au temps *t* + *dt*, étant donné que la combinaison de l'état des composants est *m* au temps *t* et que le système évolue de la cellule V_i à la cellule V_j pendant l'intervalle [*t*,*t*+*dt*].

Pour un système à trois composants dont chaque composant a quatre états, quatre états absorbants, deux variables du processus et la partition considérée ci-dessus, on a : 9 x 16 (nombre de cellules) x 64 (nombre d'états des composants) + 4 (états absorbants) = 9220 équations à résoudre pour chaque cas. On peut utiliser un pas de calcul variable. Un algorithme Runge-Kutta d'ordre 4 avec pas de calcul constant a été utilisé.

Le principal avantage de cette méthode est la réduction substantielle de mémoire demandée.

[Cocozza-Thivent et Eymard, 2006a] ont développé une méthode issue d'un schéma implicite de volumes finis pour la résolution de systèmes d'équations aux dérivées partielles. Elle revient à effectuer une discrétisation de l'espace qui ramène à un processus markovien de sauts, suivie d'une discrétisation en temps pour résoudre le système d'équations différentielles linéaires qui lui est associé. Pour cette résolution, les auteurs proposent une méthode d'Euler implicite dont l'interprétation probabiliste permet de comprendre la nature : il s'agit d'un lissage dont on a explicitement les paramètres. Les taux de transition entre les états du matériel dépendent des variables physiques, dont les valeurs à l'instant *t* sont regroupées dans un vecteur. Tant que le matériel ne change pas d'état, ces variables physiques évoluent de manière déterministe, en suivant un système d'équations différentielles dont les paramètres dépendent de l'état du matériel. Au moment où il y a une transition de l'état *i* vers l'état *j*, si la variable physique était *x*, son nouvel état est choisi selon une loi appropriée.

Les auteurs remarquent que les temps de calcul demandés par cette méthode permettent de traiter rapidement certains cas à caractère industriel, alors que la simulation classique par méthode de MC nécessite des temps sensiblement plus longs.

La discrétisation du temps dans la méthode des arbres d'évènements continus pour obtenir un arbre d'événements dynamique discret a été l'outil le plus développé pour faire face aux problèmes de fiabilité dynamique [Labeau *et al.*, 2000]. Il s'agit d'une méthode de simulation qui modélise explicitement l'évolution des variables physiques tout en tenant compte des éventuelles modifications de l'état des composants du système et du comportement humain si nécessaire. Dans l'évolution du système, tous les branchements possibles qui suivent un événement initiateur sont pris en compte. La restriction de base des arbres d'événements dynamiques discrets par rapport aux arbres d'évènements continus est la suivante : les branchements peuvent seulement avoir lieu à des intervalles de temps discrets. Cependant, à cet instant, tous les changements possibles de comportement sont pris en compte pour générer de nouvelles branches. Tout d'abord cet intervalle de temps était directement déterminé par l'utilisateur. Postérieurement un critère probabiliste a été introduit : dès que la probabilité conditionnelle de rester sur la branche

courante depuis le dernier point de branchement descend sous un seuil donné, un nouveau point de branchement est défini.

Il existe plusieurs outils qui peuvent être regroupés sous le nom des méthodologies des arbres d'événements dynamique discret : [Cojazzi et Cacciabue, 1996] DYLAM (Dynamic Logical Analytical Methodology); [Kermisch et Labeau, 2000b] et [Acosta et Siu, 1993] DETAM (Dynamic Event Tree Analysis Method); [Kermisch et Labeau, 2000b] et [Huseh et Mosleh, 1996] ADS (Analyse of Dynamic Systems, anciennement Accident Dynamic Simulator). Ces méthodes se différencient par la manière dont elles sont mises en œuvre, les techniques de branchements, la mémorisation des arbres et la modélisation des interactions homme - machine.

1.5.2.4 Les Réseaux de Petri

Il existe des méthodes qui permettent d'aborder la fiabilité dynamique en utilisant les Réseaux de Petri (RdP). De même, on peut étendre la structure des RdP stochastiques à l'approche des « RdP stochastiques fluides » [Trivedi et Kulkarni, 1993] en introduisant des places contenant des jetons continus et des arcs à flux. Ces réseaux de Petri permettent de modéliser des flux, mais on est toujours limité à la seule loi exponentielle et on ne peut pas associer des équations différentielles pour modéliser l'évolution des variables physiques.

[Dutuit *et al.*, 1997] ont utilisé les RdP stochastiques pour modéliser le comportement hybride d'un système. Mais la partie continue ne peut pas être modélisée. Ils utilisent dans leur modèle un certain type de dépendance, telle que celle qui pourrait s'établir entre des données de fiabilité (par exemple : λ, μ) relatives aux composants d'un système et un paramètre (par exemple la température). Cette approche paraît être une méthode adéquate pour l'évaluation des systèmes de contrôle de processus. Cependant [Chabot *et al.*, 1998] présentent une approche qui permet de modéliser tant la partie continue que la partie discrète de l'évolution d'un système hybride. Cette méthode entre dans le cadre des modélisations dites « hybrides » qui consistent à piloter le modèle « continu » en fonction des événements qui se produisent dans le modèle « discret ». La méthode associe dans la même simulation les deux modélisations distinctes en interaction : les RdP stochastiques temporisés pour ce qui concerne les phénomènes discrets et un système d'équations différentielles ou intégro-différentielles avec une solution numérique pour ce qui concerne les phénomènes continus (figure 1.7). Les différents problèmes de la fiabilité dynamique (vieillissement, lois de probabilités en fonction du temps) font intervenir des lois de probabilités quelconques et, en conséquence, il faut recourir à des extensions non markoviennes des réseaux de Petri stochastiques [German, 2000] afin d'obtenir des résultats réalistes.

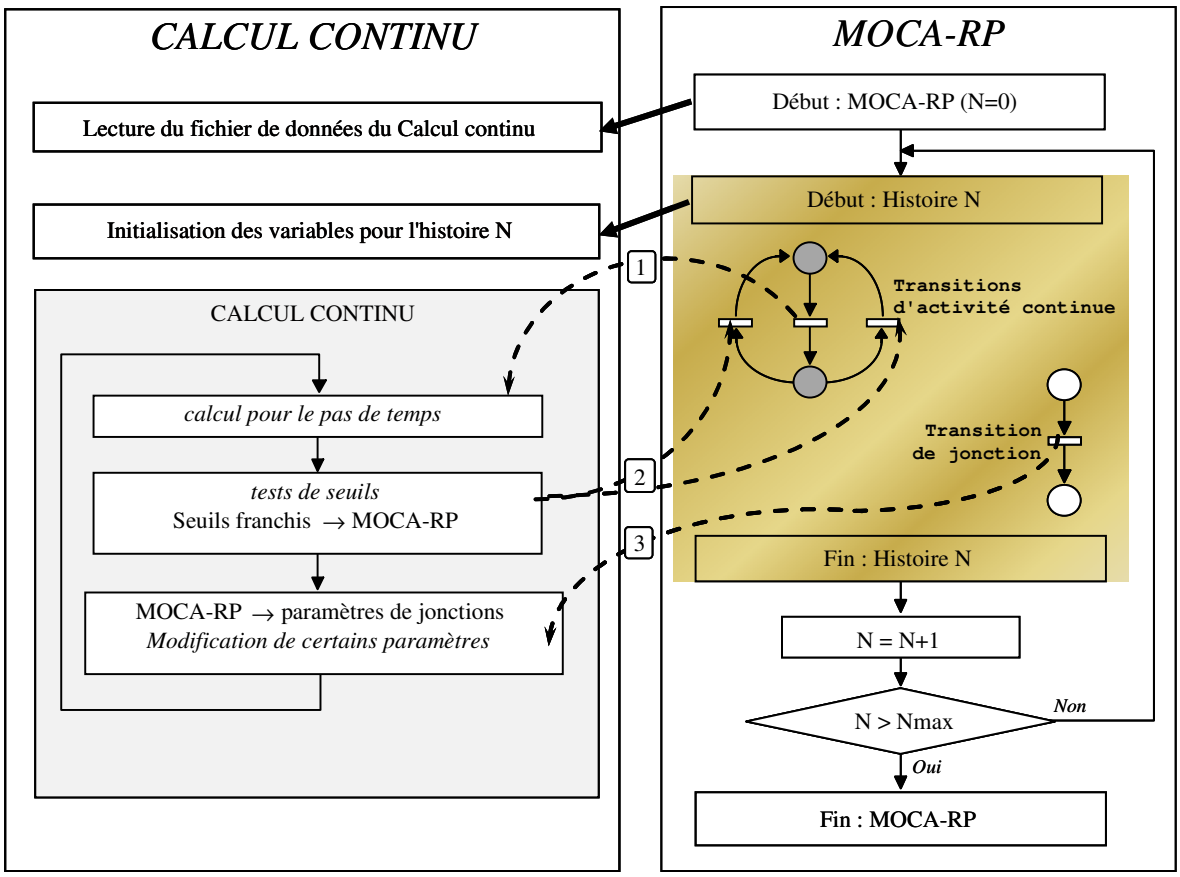


Figure 1.7 – Structure du couplage [Chabot, 1998].

[Schoenig, 2004] et [Schoenig *et al.*, 2006] proposent une méthode d'analyse quantitative basée sur la construction d'un graphe de Markov agrégé, lequel permet une limitation de l'expansion combinatoire. Ce graphe est directement déduit du modèle RdP du système. Il est conformé par un ensemble de modes fonctionnels et un ensemble de transitions pour lesquelles l'information statistique considérant les dynamiques du système ont été ajoutées. Le principe général de l'approche consiste à coupler les deux méthodes d'analyse classiques : la simulation et les graphes de Markov, afin de contourner les limites intrinsèques à ces méthodes (temps de simulation, explosion combinatoire). L'approche consiste à réaliser une agrégation des états élémentaires d'un graphe de Markov. L'influence du système sur le processus de défaillance est intégrée dans le modèle par le biais de termes de pondération des taux de défaillance. Ceux-ci permettent de prendre explicitement en compte la dynamique interne du système. Ces coefficients sont évalués par une simple simulation, permettant ainsi de traiter des systèmes complexes. Ce type d'approche, qualifié de « dynamique », permet de traiter des systèmes fortement dépendant du temps (par exemple, en cas d'existence de reconfiguration matérielles ou logicielles).

[Dejean *et al.*, 2005] présentent un modèle qui utilise les RdP stochastiques interprétés hybrides permettant de prendre en compte les aspects discrets et continus des systèmes de production pétroliers « offshore ». L'approche ne permet pas de prendre en compte des

systèmes différentiels. Le logiciel MOCA-RP® a été utilisé pour simuler ce genre de modèle.

[Labeau et Izquierdo, 2005] proposent une évolution des réseaux de Petri dérivée de la théorie de la dynamique probabiliste basée sur les stimuli. Les réseaux de Petri stochastiques généralisés permettent la modélisation de problèmes de SdF dépendant du temps. Les transitions discrètes entre états du système peuvent être conditionnées par des messages booléens prenant en compte les interactions entre les différentes parties du système modélisé. De plus, une extension récente permet de conditionner les changements des lois d'évolution dynamique du processus continu à l'activation de stimuli qui enrichissent la notion d'état du système.

D'autres approches que l'on peut classer également sous le concept de fiabilité dynamique se focalisent sur la recherche des séquences d'événements les plus critiques dans le but d'en réduire l'impact, voire de les interdire.

Dans ce sens, [Medjoudj, 2004] a présenté une version d'un algorithme qui permet la construction des scénarios critiques à partir d'un modèle RdP. Le principe de la méthode est d'enrichir progressivement le contexte dans lequel s'est produit l'événement conduisant à l'état redouté en étudiant les conflits de comportements ayant un lien de causalité avec l'occurrence de l'événement redouté. Partant d'une connaissance partielle des conditions d'occurrence de cet événement, on s'intéresse aux comportements qui permettent d'éviter le chemin critique et qui correspondent à des bifurcations représentées par des conflits de transition. L'étude des conditions de tir de ces transitions de bifurcation nous informe de manière plus complète sur les conditions d'occurrence de l'événement redouté.

Par ailleurs, [Sadou et al., 2006] présentent une méthode pour l'analyse qualitative des systèmes hybrides du point de vue de la fiabilité dynamique. La méthode est basée sur une modélisation orientée objet des systèmes hybrides par un réseau de Petri et un ensemble d'équations algébro-différentielles. Une analyse logique des causalités entre les changements d'états est alors possible. A partir d'une connaissance partielle de l'état redouté, il est possible de remonter une chaîne de causalité et caractériser tous les scénarios possibles menant vers cet état redouté. Chaque scénario est donné par un ordre partiel entre les événements nécessaires à l'occurrence de l'événement redouté, à la différence des arbres de défaillances qui donnent un ensemble de combinaisons des événements nécessaires pour obtenir une situation redoutée. Les scénarios sont représentés par un objet partiel défini par un graphe orienté : E correspond aux nœuds et A les arcs. Les nœuds E représentent un ensemble de franchissements de transitions et les arcs A sont un ensemble de paire (t_i, t_j) telles que t_i précède à t_j (t_i et t_j sont des franchissements de transition). Les ordres entre les événements sont représentés par des flèches discontinues. La figure 1.8 représente un scénario redouté. Elle montre que les défaillances des deux composants (deux pompes) précèdent l'événement redouté (par exemple l'assèchement d'un réservoir, transition t_5). Les événements E_1 et E_2 représentent l'enrichissement de marquage de la place PF, par contre I_1 représente l'événement initial (marquage initial) correspondant à la présence d'un jeton dans la place P_1 produit lors de l'étape du raisonnement arrière.

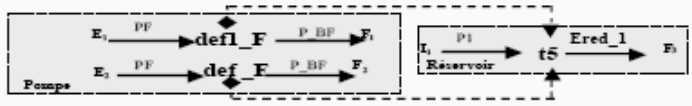


Figure 1.8 – Un scénario redouté [Sadou et al., 2006].

Dans un même ordre d'idée, S. Humbert [Humbert 2008] a proposé de modéliser en langage Altarica les propagations de pannes potentielles d'un système en intégrant des hypothèses de modes de défaillance sur ses parties logicielles et matérielles. L'analyse du modèle permet de déduire des exigences élémentaires sur les fonctions logicielles en particulier. Le raffinement de ces exigences les rend vérifiables sur un modèle métier de conception du logiciel.

1.5.2.5 Autres types de méthodes

[Cabarbaye et Laulheret, 2005] montrent l'apport de la modélisation récursive à l'évaluation de la SdF de systèmes dynamiques complexes, dont notamment des systèmes hybrides. Celle-ci se contente de décrire le comportement d'un système entre deux instants courants t et $t+\Delta t$. Ces instants peuvent aussi bien correspondre à une incrémentation temporelle (simulation - temps) qu'à l'occurrence d'événements particuliers (simulations - événement) tels que des changements aléatoires d'état du système : une défaillance, ou une remise en service ou un franchissement de seuils caractéristiques par des variables continues. Comme l'illustre la figure 1.9, la simulation comportementale du système au cours du temps consiste à réinjecter l'état final obtenu par le modèle récursif à $t_i + \Delta t_i$ en entrée du modèle à $t_j = t_i + \Delta t_i$, en partant d'un état initial défini à t_0. Il est à noter que cette simulation peut aussi bien rendre compte de caractéristique en régime asymptotique que transitoire durant une mission limitée dans le temps (la fiabilité opérationnelle par exemple) ; une réinitialisation périodique étant alors opérée par le modèle dès la fin de la durée de la mission. Son efficacité de traitement en termes de temps de calcul est remarquable car elle se limite au juste nécessaire à la modélisation dans l'état courant.

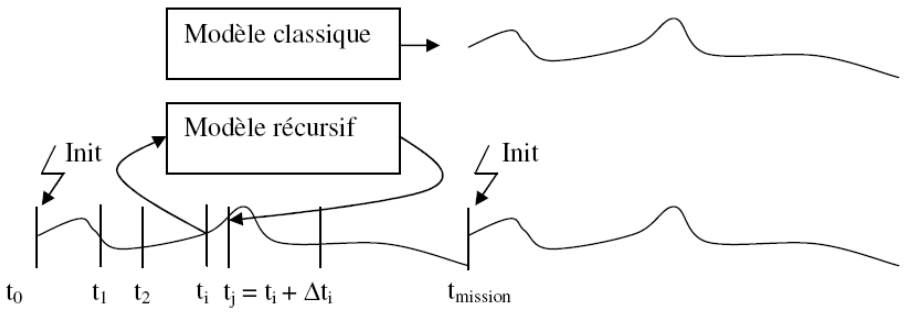


Figure 1.9 – Modélisation récursive [Cabarbaye et Laulheret, 2005].

D'un autre côté, [Tchangani et Noyes, 2005] considèrent que le problème de la modélisation et de l'analyse de la fiabilité dynamique se pose dès lors que l'état de fonctionnement du système et l'état des variables fonctionnelles du système lui même s'influencent mutuellement avec en plus la possibilité d'une éventuelle perturbation exogène. Les Réseaux Bayesiens Dynamiques (Dynamic Bayesian Networks, DBN) semblent constituer un outil mathématique intéressant pour modéliser ce problème en permettant une représentation graphique des processus stochastiques. Les DBN sont des modèles graphiques orientés des processus stochastiques représentant des états observés ou cachés en termes de variables d'état, lesquelles peuvent avoir des interdépendances complexes, afin d'obtenir une paramétrisation compacte du modèle. Les DBN sont complètement définis par deux composants : sa structure, laquelle est un graphe acyclique orienté (des nœuds représentant les variables et des arcs représentant la relation entre les variables) et les paramètres représentant les densités de probabilité. La figure 1.10 montre un exemple de la représentation graphique d'un DBN. Si l'intérêt de l'approche est évident, son application à des problèmes physiques complexes reste difficile.

Le contexte du problème de la fiabilité dynamique est défini par les variables suivantes :

- $s(t) \in S = \{1, 2, ..., s\}$ est l'état (discret) du système, composant ou matériel (les modes de fonctionnement) à l'instant t ; il appartient à un ensemble finit S.

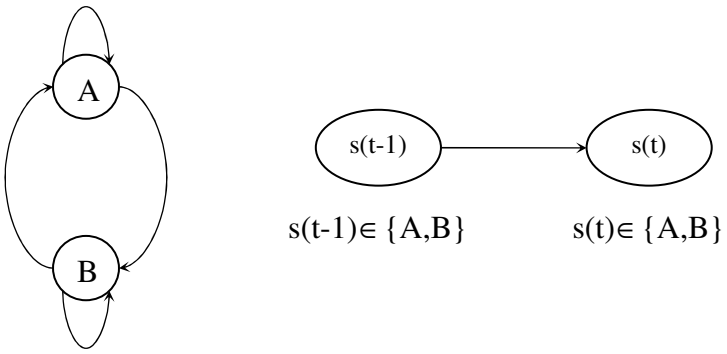
- $x(t) \in \mathbb{R}^n$ est l'état (continu) des variables du processus disponibles à l'instant t; $\mathbb{R}^n$ représente un espace vectoriel réel de dimension n.

- $y(t) \in \mathbb{R}^l$ est l'observation ou la mesure des variables du processus disponibles à l'instant t.

- $a(t) \in A = \{a_1, a_2, \ldots, a_n\}$ est l'action du marqueur de décision à l'instant t qu'influence l'état du système (matériel) ; il y a un nombre d'actions définies par l'ensemble A disponible pour le marqueur de décision à chaque instant t.

- $w(t) \in \mathbb{R}^p$ est la perturbation exogène qui influence le système et/ou le comportement des variables du processus.

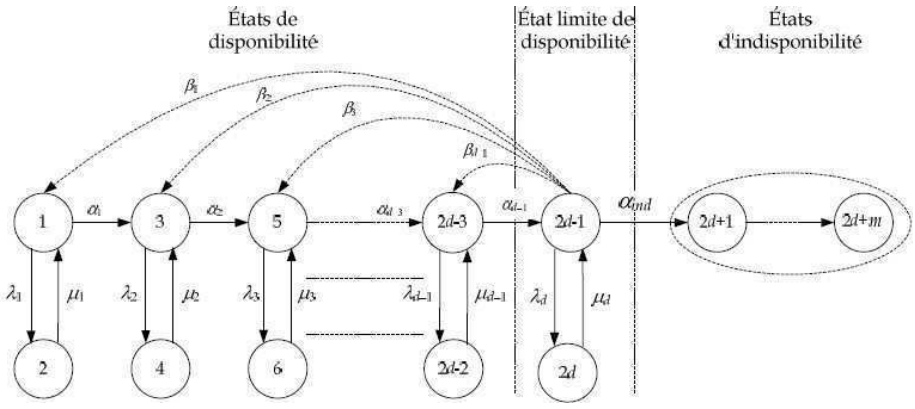
- $\pi(t) = [\pi_1(t) \ldots \pi_s(t)]$ où $\pi_i(t) = \Pr\{s(t) = i\}$ est la probabilité que le matériel est dans l'état i à l'instant de temps t.



a) Chaîne de Markov b) DBN

Figure 1.10 – Exemple d'une chaîne de Markov et de sa représentation DBN.

[Soro *et al.*, 2006] abordent un cas plus général de la fiabilité dynamique. Il s'agit d'un système multi - états - dégradable (SME), lequel peut avoir différents niveaux de performances, plusieurs modes de défaillances avec différents effets sur leurs performances, notamment la dégradation. Ils expliquent que ces performances peuvent être structurelles, fonctionnelles, etc. Ainsi, la productivité ou la capacité, la fiabilité ou la disponibilité, la vitesse de traitement, etc. peuvent représenter la mesure de performance d'un SME. Par conséquent, ils présentent un modèle qui se dégrade avec l'usage et évaluent les indices de performance qui le caractérisent à partir d'un profil de demande. Le diagramme de la figure 1.11 illustre les différents états possibles que peut occuper le système multi-états.



État (i) : État i du système
État (1) : État nominal
État ($2i$-1) : État dégradés (i=2,…,d)
État ($2i$) : État de défaillance (i=1,…,d)
État ($2d$+m) : État de défaillance complète
λ_i : Taux de défaillance (i=1,…,d)
μ_i : Taux de réparation minimale (i=1,…,d)
α_i : Taux de dégradation (i=1,…,d)
α_{ind} : Taux d'indisponibilité (α_d)
β_i : Taux de réparation (i=1,…,d-1)
$P_i(t)$: Pr{le système soit à l'état i au temps t}
G_j : Niveaux de performance des états j (j=2i-1)
W : Demande constant

Figure 1.11 – Diagramme de transition d'un SME.

1.6 Systèmes dynamiques hybrides et fiabilité dynamique

Les innovations technologiques récentes ont suscité un intérêt considérable pour l'étude des processus dynamiques complexes caractérisés par une interaction forte entre des dynamiques continues, régies par des équations différentielles ou aux différences, et des dynamiques discrètes, décrites par des machines à état fini, des règles logiques du type si-alors ou plus généralement par un système à événement discret. De tels systèmes appelés systèmes dynamiques hybrides (SDH) [Zaytoon, 2001], sont caractérisés par des commutations entre plusieurs modes de fonctionnement où chaque mode est régi par ses propres lois dynamiques continues. Les transitions entre les modes peuvent alors se produire selon un modèle événementiel :

- lorsque des variables continues atteignent certains seuils spécifiques (événements d'états),

- après une certaine durée ou période de temps (événements de temps),

- ou par des entrées externes (événements d'entrée).

C'est cette interaction forte entre les deux dynamiques qui pose les défis les plus importants dans l'analyse et le contrôle de ces systèmes. L'étude des SDH ne se justifie que pour des systèmes où la relation continu/discret entraîne l'existence de nombreuses situations et des risques de dysfonctionnement liés à cette relation. Les difficultés d'analyse et de synthèse des SDH font de leur simulation un outil d'étude privilégié. Les programmes de simulation sont très variés, ils dépendent souvent des modèles utilisés pour représenter le SDH, de leur objectif et de la manière dont ils ont été construits. Cependant, la simulation des SDH pose quelques problèmes :

- Coupler dans la simulation la partie discrète et la partie continue. Il ne sera possible d'effectuer l'intégration des équations différentielles par une procédure numérique, ou par le calcul dans le cas linéaire, que si le champ de vecteur est continu par rapport au temps.

- Traiter les événements : ne pas les rater, respecter l'ordre d'occurrence et les détecter avec précision.

- Réinitialiser : trouver le nouveau modèle et les conditions initiales.

- Essayer de garder des temps de calculs raisonnables en choisissant au mieux le pas de calcul.

Ces considérations amènent à proposer la structure d'algorithme de simulation des SDH suivante :

a. initialiser les parties événementielle et continue,

b. intégrer les équations différentielles sur un pas de calcul,

c. se recaler sur le premier événement intervenu :

- calculer l'instant,
- calculer l'état,
- calculer le nouveau modèle,

d. recommencer en b.

Le critère d'arrêt dépendra bien entendu de l'objet de la simulation.

Les performances d'une simulation de SDH vont dépendre :

- des précisions obtenues dans la résolution des équations de la partie continue,
- des précisions dans le calcul des instants de commutation,
- des précisions dans les calculs des recalages des états,
- de la manière de choisir le pas de calcul élémentaire.

Ces performances ne sont pas totalement indépendantes les unes des autres. En particulier, les précisions sont largement liées au pas de calcul, mais un pas de calcul trop petit peut entraîner des temps de simulation exagérés.

Le problème de la fiabilité dynamique s'apparente à celui des SDH, étant donné que la fiabilité dynamique a pour objet de prendre en compte les interactions entre le comportement fonctionnel dynamique et déterministe d'un système (partie continue) et le comportement dysfonctionnel stochastique (partie discrète) de ses composants. Les dépendances entre variables continues et stochastiques sont multiformes et n'interviennent pas uniquement lors du franchissement de seuils.

Or nous avons affaire à des systèmes à structure évoluant dans le temps selon des lois qui peuvent dépendre de l'évolution normale du système décrite par des modèles continus. Pour cela, il faut modéliser cette évolution du système avec un automate par exemple, dont les changements d'états sont définis par les franchissements de seuils de variables continues du système (par exemple une température, une pression, un flux…) ou par l'occurrence d'un événement aléatoire (une défaillance d'un composant ou d''un opérateur humain). Entre deux changements d'état, la structure du modèle dysfonctionnel du système reste inchangée.

Donc, le modèle à simuler doit prendre en compte les éléments suivants :

- le caractère hybride du système: combinaison d'équations d'état continu et d'automate d'états finis,

- la reconfiguration des équations d'état continu sur l'occurrence des événements (entraînant donc une modification de la structure fiabiliste),

- le caractère déterministe ou stochastique des variables et des événements,

- la possibilité d'injections des défaillances,

- le diagnostic et la réaction à ces défaillances en temps réel,

- la prise en compte des lois de probabilités quelconques (et notamment en fonction du temps) et l'interaction entre ces lois de probabilités et l'état continu du système (par exemple un taux de défaillance fonction de la température).

1.7 Conclusion

Nous avons présenté les différentes méthodologies qui ont été développées afin de traiter et surmonter les problèmes relatifs à la fiabilité dynamique. Ces méthodologies modélisent explicitement l'évolution des variables physiques et l'influence mutuelle entre le comportement stochastique (défaillances des composants) et déterministe du système. Ces méthodologies diffèrent en termes d'hypothèses et de modèles et de méthodes de calcul.

Cependant, étant donnée la complexité et les limites mathématiques pour évaluer la fiabilité d'un système dynamique hybride, nous pouvons conclure qu'elle n'est accessible que dans des cas simples. Nous nous proposons d'explorer la voie de la simulation du comportement complet du système. Pour cela, nous devons établir un modèle de type état transition simulable du système, capable de résoudre les équations différentielles associées aux états. Nous n'avons pas retenu les réseaux de Petri comme modèle car nous n'avons pas trouvé d'outil logiciel intégrant dans un même environnement l'exécution interactive d'un RdP et d'un puissant solveur numérique d'équations différentielles. Cela nous a emmené à définir et à implémenter dans un environnement adéquat un automate stochastique hybride. A l'aide de cet automate nous pourrons réaliser une simulation de Monte Carlo du comportement fonctionnel et dysfonctionnel du système et ainsi accéder à l'évaluation des indicateurs de la sûreté de fonctionnement.

Chapitre 2

Automate stochastique hybride

La complexité mathématique de l'évaluation analytique de la sûreté de fonctionnement d'un système dynamique nous amène à recourir à la simulation. Pour cela, il nous faut disposer d'un modèle adapté de la fonction de structure du système.

Nous postulons que la fonction de structure d'un système en contexte dynamique est un Automate Stochastique Hybride (ASH). C'est un automate parce qu'il est composé d'un ensemble d'états discrets. Il est hybride parce que chaque état discret est défini par un système d'équations continues et par un sous-ensemble de transitions de sortie définies par des seuils sur ces variables continues. Il est stochastique parce que chaque état discret est défini par un ensemble de variables aléatoires et un sous-ensemble de transitions de sortie définies par des seuils sur les variables aléatoires.

Dans ce chapitre, nous présentons l'ASH permettant de modéliser les interactions entre les défaillances et les réparations des composants et les dynamiques continues du système, dans le but d'accéder à l'évaluation de la fiabilité d'un système dynamique par la simulation.

2.1 Automates

2.1.1 Automate à états finis

La théorie des automates à états finis (à nombre d'états finis) a été principalement développée avec la théorie des langages. Ces modèles reviennent à spécifier des ensembles d'états et des transitions entre ces états [Arnold, 1992]. Les langages et les automates permettent de traiter mathématiquement les problèmes relatifs aux Systèmes à Événements Discrets (SED), essentiellement d'un point de vue logique (analyse qualitative).

Chaque SED a un ensemble d'événements qui lui est associé. Ces événements font évaluer le SED. Cet ensemble peut être vu comme un alphabet d'un langage et les séquences d'événements sont des mots (aussi appelés chaînes) de ce langage. Un automate est alors un dispositif qui engendre un langage en manipulant l'alphabet (les événements).

Un automate à états finis est un quintuplet (A,G,λ,I,F) dans lequel :

- G est un multi-graphe constitué d'un ensemble de sommets $\mathcal{X}$, d'un ensemble d'arcs $\mathcal{A}$ et de 2 applications de l'ensemble des arcs dans l'ensemble des sommets qui à chaque arc font correspondre un sommet origine et un sommet but.

- On distingue un sous ensemble d'états initiaux I et un sous ensemble F d'états terminaux.

- A est un alphabet, ensemble de lettres contenant un élément neutre.

- λ est une application qui à chaque arc associe un événement (élément de l'alphabet). λ est telle que si deux arcs ont même origine et même but, les événements associés à ces arcs doivent être différents.

En résumé Pour la modélisation des SED, les sommets du graphe sont appelés états, les lettres de l'alphabet sont les événements du SED. Un automate à états finis peut être défini par un quintuplet [Arnold, 1992] :

$$\mathsf{A} = (\mathcal{X}, \Sigma, f, x_0, \mathcal{X}_f) \tag{2.1}$$

dans lequel :

- $\mathcal{X}$ est l'ensemble (fini) des états ; c'est l'espace des états, il est discret,

- Σ est un alphabet fini, ou ensemble des événements.

- f est la fonction de transition d'état $f : \mathcal{X} \times \Sigma \to \mathcal{X}$,

- x_0 est l'unique état initial (automate déterministe),

- $\mathcal{X}_f$ est l'ensemble d'états finaux, $\mathcal{X}_f \subseteq \mathcal{X}$.

La fonction de transition *f* résume les trois applications précédentes. Elle est définie, pour chaque état, par les événements qui sont associés aux arcs qui ont cet état pour origine (ils sont définis par la fonction des événements actifs notée Γ). Elle fournit l'état but de chacun de ces arcs.

Considérons, par exemple, l'automate à états finis suivant [Cassandras et Lafortune, 1999] :

- $X = \{x, y, z\}$,

- $\Sigma = \{a, b, g\}$,

- $f : X \times \Sigma \to X$:

  · $f(x, a) = x$ · $f(x, g) = z$ · $f(z, b) = z$

  · $f(y, a) = x$ · $f(y, b) = y$ · $f(z, a) = f(z, g) = y$

où $f(x, g) = z$ signifie que si l'automate se trouve dans l'état x alors sur l'occurrence de l'événement g, l'automate fera une transition vers l'état z. De même pour les autres fonctions :

- x est l'état initial,

- $X_f = \{x, z\}$,

- $\Gamma(x) = \{a,g\}$, - $\Gamma(y) = \{a,b\}$, - $\Gamma(z) = \{a,b,g\}$

La figure (2.1) donne le dessin de cet automate, les états sont des cercles (doubles cercles pour les états finaux, une flèche sans origine indique l'état initial, les arcs sont des flèches et sur chaque arc est indiqué l'événement associé :

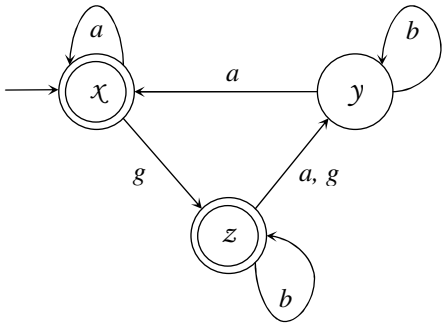


Figure 2.1 – Un automate simple [Cassandras et Lafortune, 1999].

Trois remarques peuvent être faites :

1. Le système peut ne pas changer d'état sur l'occurrence d'un événement :

   $f(x, a) = x$; $f(y,b) = y$; $f(z,b) = z$,

2. La même transition peut se produire sur l'occurrence de deux différents événements. Voir : $f(z, a) = f(z, g) = y$

3. Dans l'exemple, la fonction f est partiellement définie sur son domaine $\mathcal{X} \times \Sigma$: $f(x, b)$ et $f(y, g)$ ne sont pas définies. Lorsque f est complètement définie, on dit alors que l'automate est complet.

Le langage d'un automate est l'ensemble des suites d'événements associés à des chemins dans le graphe de l'automate. On distingue deux langages particuliers : le langage généré associé à tous les chemins possibles à partir de l'état initial et le langage reconnu associé aux chemins allant de l'état initial à un des états finaux. Dans les SED les langages représentent des séquences possibles d'événements : le langage généré représente toutes les séquences possibles, le langage reconnu (ou marqué) représente des séquences caractéristiques de certaines fonctions ou tâches particulières.

2.1.2 Automate hybride

[Alur *et al.,* 1995] ont modélisé les SDH comme un automate fini muni de variables qui évoluent continuellement avec le temps conformément aux lois dynamiques physiques. Un Automate Hybride (AH) est « un modèle mathématique pour les systèmes hybrides, lequel combine, dans un seul formalisme, les transitions de l'automate pour capturer les changements discrets et avec les équations différentielles pour capturer les changements continus » [Alur *et al.,* 1997]. Ce sont [Alur *et al.,* 1993] qui ont introduit la structure mathématique de l'automate hybride comme un modèle et un langage de spécification pour les systèmes hybrides.

Un AH est un 6-tuple [Johansson, 2000]

$$\mathsf{H} = (\mathcal{X}, X, A, \text{Init}, \text{Inv}, \text{Jump}) \tag{2.2}$$

dans lequel :

- $\mathcal{X} = \{x_1, x_2, \ldots, x_p\}$, $p>0$ est un ensemble fini de modes discrets ou états discrets,

- $X = \{x_1, \ldots, x_n\}$, $n \geq 0$, est un ensemble fini ordonné de variables réelles (vecteur) évoluant dans le temps appelées variables continues. Il représente les dynamiques continues de H,

- A est une fonction de $\mathbb{R}^n \times \mathcal{X}$ dans $\mathbb{R}^n$ appelée champ de vecteur. Elle définit la dynamique continue dans chaque état discret à travers une équation différentielle du temps de la forme $\dot{x} = f(\chi, x)$, Elle est définie pour chaque état : $\dot{X} = f(x_i, X)$,

- Init $\subset \mathcal{X} \times \mathbb{R}^n$ définit l'état initial de H à l'instant t=0. Un état initial de l'AH est une paire (x_0, X_0) consistant d'un état discret $x_0 \in \mathcal{X}$ et d'une valeur

$X_0 \in \mathbb{R}^n$ qui initialise X,

- Inv est un ensemble d'invariants spécifiques à chaque état (condition de séjour dans l'état),

- Jump est un ensemble de fonctions appelées conditions de saut. Celles-ci spécifient si un saut d'un état discret vers un autre est possible et quelle est la nouvelle valeur de l'état continu après le saut. (inclus donc la fonction de transition définie dans les automates à états finis).

Etant donné que nous avons développé l'automate stochastique hybride sur la base de l'automate hybride défini par l'INRIA, nous présentons leur définition [Najafi et Nikoukhah, 2007]. Cette définition est fondée sur la définition donné par Henzinger [Henzinger et *al.*, 1995].

H est un 7-tuple lequel est défini comme :

$$H = (V,E,X,F,Invariant,Initial,Jump) \tag{2.3}$$

- V est l'ensemble d'états discrets ou « modes de contrôle » $\{v_1, \ldots, v_M\}$ où M est le numéro de modes de contrôle,

- E est l'ensemble de « commutations de contrôle » lesquels identifient le mode initial et le mode destination durant une transition,

- X est l'ensemble de variables de valeur réelles $\{x_1,\ldots,x_N\}$ où N est la dimension de H. $\dot{X}$ est la première dérivée en fonction du temps de X,

- F est le prédicat sur $\{X, \dot{X}\}$ assigné à chaque mode de contrôle,

- *Invariant* définit l'intervalle admissible pour $\{X, \dot{X}\}$ dans chaque mode,

- *Inital* définit les valeurs initiales de $\{X, \dot{X}\}$ dans chaque mode,

- *Jump* est une fonction d'étiquetage des arcs qui assigne un prédicat sur $\{X, \dot{X}\}$ à chaque arc.

2.1.3 Automate stochastique hybride

Étant donné que de nombreuses applications de sécurité critique sont des systèmes hybrides, des analyses de fiabilité rigoureuses sont requises exigeant une modélisation formelle [Henzinger, 1996]. La réalité impose donc de prendre en compte les défaillances des composants ou les incertitudes sur la connaissance du système. Certains événements ou variables prennent alors un caractère stochastique. Pour cette raison, nous avons défini et implémenté un automate stochastique hybride (ASH) pour modéliser le comportement

dysfonctionnel d'un SDH pour évaluer par simulation les paramètres de la Sûreté de Fonctionnement. L'ASH prend en compte les différents modes continus de fonctionnement du système et le passage de l'un à l'autre sur l'occurrence des événements déterministes et stochastiques. Les premiers sont produits par franchissement de seuils des variables continues, les seconds sont produits par les défaillances des composants simulées par un générateur aléatoire en fonction de leurs lois de probabilités. L'automate nous permet d'accéder aux grandeurs de la SdF, lesquelles sont obtenues par statistique sur un grand nombre de simulations (méthode de Monte Carlo).

L'automate hybride décrit précédemment ne fait référence au temps qu'à travers la dynamique des variables continues dont l'évolution est décrite par des équations différentielles du temps. Les changements d'états ne dépendent que de cette évolution définie par certaines conditions (ce que certains appellent des événements endogènes). Ici, les instants de changement d'état peuvent aussi être dus à des événements aléatoires donc non prévisibles (on en connaît seulement qu'une distribution des probabilités d'occurrence). Le temps doit donc apparaître explicitement dans la définition de l'automate comme une variable partagée par les deux dynamiques continue et discrète. Il faut noter également que les lois de distribution de probabilités de défaillance des composants sont susceptibles de voir leurs paramètres dépendre de certaines variables continues (température, pression, etc.) mais aussi de la trajectoire suivie par le système (chemin suivi dans l'automate depuis l'état initial) pour décrire certains phénomènes de vieillissement (exemple nombre de commutations d'une vanne…). Pour répondre à ces exigences, nous proposons la définition suivante du concept d'automate stochastique hybride :

Un automate stochastique hybride est défini comme un 11-tuple

$$ASH = (\mathcal{X}, \mathcal{E}, \mathcal{A}, X, A, \mathcal{H}, \mathcal{F}, p, x_0, x_0, p_0) \tag{2.4}$$

dans lequel :

- $\mathcal{X}$ est un ensemble fini d'états discrets $\{x^1, x^2, \ldots, x^m\}$,
- $\mathcal{E}$ est un ensemble fini d'événements $\{e_1,\ldots,e_r\}$ déterministes ou stochastiques,
- X est un ensemble fini de variables réelles évoluant dans le temps $\{x_1,\ldots,x_n\}$,
- $\mathcal{A}$ est un ensemble fini d'arcs de la forme (x, e, G, R, x') où :

  x et x' sont les états origine et but de l'arc k, e_j est l'événement associé à l'arc, G_k est la condition de garde sur X dans l'état x et R_k est la fonction de réinitialisation de X dans l'état x',

- $A : \mathcal{X} \times X \to (\mathbb{R}^{n+} \to \mathbb{R})$ est une fonction des « activités », qui associe à un élément de $\mathcal{X} \times X$ une fonction définie sur $\mathbb{R}^{n+}$ et à valeur dans $\mathbb{R}$,
- $\mathcal{H}$ est un ensemble fini d'horloges définies sur $\mathbb{R}$,
- $\mathcal{F} : \mathcal{H} \to \left(\mathbb{R} \to [0,1]\right)$ est une application qui associe à chaque horloge une fonction de répartition de probabilité,

- p_t^f est une distribution de probabilités de transition d'état $p(x^i \mid x^l, e)$. Par exemple, si nous avons le même événement e_q définissant les transitions de l'état discret x^f vers les états discrets x^1 , x^2 , …, x^j (nous disons qu'il y a j transitions en conflit, l'automate à états finis sous jacent ne serait alors pas déterministe), nous pouvons définir la probabilité p_1^f de passer de l'état x^f à l'état x^1, la probabilité p_2^f de passer de l'état x^f à l'état x^2 et la probabilité p_j^f de passer de l'état x^f à l'état x^j, avec $p_1^f + p_2^f + \ldots + p_j^f = 1$
- x^0, X_0 et p_t^0 correspondent respectivement à l'état discret initial, à la valeur initiale du vecteur d'état continu dans l'état initial discret et à la distribution initiale de probabilités de transition dans l'état initial discret.

Les éléments $\mathcal{X}$, $\mathcal{E}$ et $\mathcal{A}$ de l'automate stochastique hybride correspondent à l'automate à états finis définissant sa partie événementielle (discrète). En revanche, X, A, R et G définissent sa partie continue. $\mathcal{H}$ correspond à son aspect temporisé et finalement $\mathcal{F}$ et p expriment son aspect stochastique.

Le fonctionnement de cet automate s'interprète de la manière suivante : si le système est dans l'état x^f , il est réceptif à un sous ensemble des événements de $\mathcal{E}$ associés aux différents arcs sortant de cet état. Sur occurrence d'un de ces événements e_q associé à l'arc k, si la condition de garde G_k associée à cet arc est vérifiée, le système passe à l'état x^i but de l'arc. La fonction R_k également associée à cet arc définit les valeurs initiales des variables continues X du système dans l'état x^i. Si e_q est associé à plusieurs arcs, l'état final résultera du tir de la distribution de probabilité p_t^f.

Les durées de bon fonctionnement et de réparations des composants sont matérialisées par les horloges $\mathcal{H}$. Ces durées sont élaborées par tirage aléatoire à partir des fonctions de répartitions de probabilité $\mathcal{F}$.

Nous avons implémenté cet automate dans l'environnement de simulation Scicos – Scilab. Nous en exposerons les modalités dans le paragraphe 2.3.3. Auparavant, nous donnerons quelques éléments introductifs à l'environnement Scicos - Scilab.

2.2 Scicos

L'outil informatique que nous avons utilisé pour implémenter l'ASH est la boîte à outils Scicos de Scilab [Scilab, 1989] [Scicos, 1989], environnement de simulation libre développé par l'INRIA et qui connaît un fort succès dans le monde universitaire.

2.2.1 Boîte à outil Scicos

Scicos (« Scilab Connected Object Simulator ») est une boîte à outil de Scilab pour modéliser et simuler des systèmes dynamiques. Scicos traite, en particulier, l'interaction

entre les dynamiques en temps continu et les systèmes à événements. Scicos inclut un éditeur graphique pour la construction des modèles à travers des blocs, un compilateur et un simulateur.

Un diagramme à blocs de Scicos est composé d'un ensemble de blocs et de liens de connexion entre eux. Un bloc correspond à une opération, les blocs sont interconnectés à travers des liens permettant de construire un modèle ou un algorithme. Nous pouvons aussi construire d'autres blocs à partir des blocs basiques ou bien définir de nouveaux blocs. Ceux-ci peuvent avoir plusieurs entrées et sorties, des états à temps continu, des états à temps discrets, des fonctions de croisement par zéro (en anglais « zero-crossing »),… (figure 2.2). Mais, il n'est pas nécessaire d'avoir tous ces éléments dans un bloc. Scicos permet aussi d'avoir une interaction avec les codes C, Fortran et même Scilab pour créer des nouveaux blocs [Campbell *et al.*, 2006]. En particulier, les entrées régulières reçoivent les données d'autres blocs à travers les liens de connexion. Les sorties régulières envoient les données vers d'autres blocs à travers les liens de connexion. Par contre, les entrées et les sorties d'activation correspondent aux événements reçus et/ou transmis par le bloc.

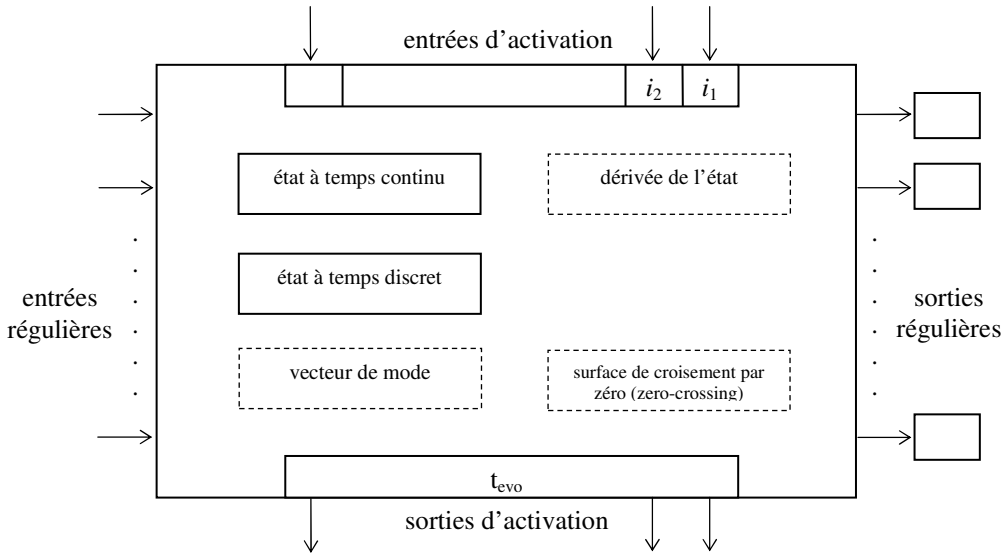


Figure 2.2 – Bloc Scicos : zoom sur l'intérieur d'un bloc [Campbell *et al.*, 2006].

2.2.2 Comportement des blocs

Vu de l'intérieur de Scicos, chaque bloc Scicos est défini par deux fonctions (figure 2.3). La première, appelée fonction d'interface, doit être faite en langage Scilab. La structure de données du bloc est une liste Scilab laquelle contient toute l'information nécessaire à compiler et simuler du diagramme Scicos. A travers cette fonction nous pouvons spécifier la géométrie du bloc, le nombre d'entrées et de sorties, le type du bloc, etc. Elle traite les interactions avec l'éditeur et parfois avec l'utilisateur. Pour actualiser les paramètres et les états initiaux du bloc, si nécessaire, une fenêtre d'interface peut être

développée. La seconde fonction, appelée fonction de calcul, est normalement écrite en langage C, mais elle peut être développée en langage Scilab. Cette fonction définit le comportement du bloc pendant la simulation. Cette fonction est appelée par le simulateur pour réaliser le calcul numérique tels que le calcul des dérivées de l'état continu, la mise à jour des sorties, l'évaluation des fonctions de croissement par zéro, etc. [Najafi, 2005].

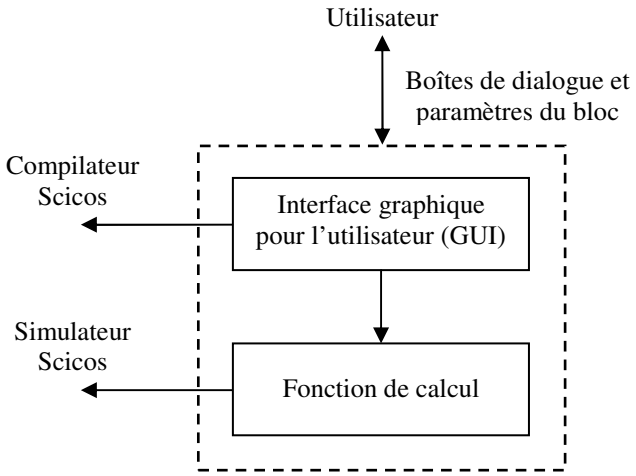


Figure 2.3 – Un bloc Scicos.

2.2.3 Résolveurs numériques

Pour simuler les diagrammes ou modèles le logiciel Scicos utilise deux résolveurs numériques : LSODAR et DASKR [Najafi, 2005]. LSODAR est un résolveur d'équations différentielles ordinaires (en anglais ODE). Cela signifie que la dynamique continue du modèle Scicos peut être représentée par l'équation suivante :

$$\dot{X} = F(t, X) \tag{2.5}$$

où t est le temps, X la variable physique et $\dot{X}$ la dérivée de X par rapport au temps. Ce système est résolu en appelant le résolveur LSODAR de façon répétitive. Dans chaque appel le résolveur calcule la solution de (2.5) sur un période de temps. La fonction F peut changer en fonction de la dynamique de temps discret (les événements).

Le modèle dynamique de certains systèmes hybrides ne peut pas être décrit par l'équation (2.5). Ces dynamiques sont décrites par des équations algèbro-différentielles (en anglais DEA). Les dynamiques du système peuvent être représentées de façon implicite par les équations :

$$0 = f_1\left(\dot{X}, X, t, U\right) \tag{2.6}$$

$$y = f_2\left(\dot{X}, X, t, U\right) \tag{2.7}$$

où U, X et y représentent respectivement les entrées du système, le vecteur d'état continu et les sorties. Pour résoudre de telles équations, Scicos fait appel de façon répétitive à DASKR. Du point de vue de l'utilisateur, la différence entre les ODE et les DAE réside dans le fait, que pour initialiser la simulation, les valeurs initiales pour X et $\dot{X}$ pour les DAE doivent être apportées tandis que pour les ODE il est suffit de fournir la valeur de X.

2.2.4 Passage par zéro

Pour détecter le passage par zéro (zero-crossing) d'une fonction les deux résolveurs numériques utilisent à peu près les mêmes routines. Le résolveur DASKR utilise deux routines pour détecter les passages par zéro : Drchek et Droots. Le rôle de la routine Drchek (« root-finder module ») est déterminé si une des fonctions de « zero-crossing » change de signe pendant l'intervalle d'intégration des fonctions. Lorsque le résolveur démarre l'intégration, au début de chaque intégration, le temps d'intégration global est partitionné en plusieurs petits intervalles de temps avec une taille constante ou variable, appelés pas d'intégration. Quand un passage par zéro arrive, le résolveur numérique arrête la simulation, identifie l'instant exact du croisement et retourne à la simulation.

La méthode d'intégration « backward differentiation formula (BDF) » est utilisée pour faire avancer le temps sur chaque pas d'intégration. Drchek lit toutes les fonctions de « zero-crossing », sauvegarde leurs valeurs en début de chaque pas d'intégration et les compare à la fin des pas d'intégration. S'il y a un changement de signe Drchek appelle la routine Droots pour déterminer l'instant exact du « zero-crossing » le plus proche. S'il y a un croissement par zéro, la routine Droots déterminera l'instant exact du croissement. Si plus d'une fonction a changé de signe pendant le dernier pas d'intégration, Droots déterminera le croissement le plus à gauche. Les méthodes BDF sont des méthodes multi-pas linéaires implicites qui dépendent des points de solution précédents pour générer un nouveau point de solution approchée. La méthode BDF d'ordre 1, est une méthode d'Euler implicite, c'est-à-dire, pour intégrer (2.5), la DAE est discrétisée par l'équation (2.8) et elle est résolue pour x_{n+1}.

$$\frac{x_{n+1} - x_n}{h_{n+1}} = f(x_{n+1}, t_{n+1}) \tag{2.8}$$

2.2.5 Mode batch

Etant donné que Scicos réside à l'intérieur de Scilab et qu'il a été développé en langage Scilab, les simulations peuvent non seulement s'effectuer directement sur Scicos, mais aussi être relancées à partir de Scilab, c'est-à-dire que les fonctionnalités de Scicos peuvent être pilotées à travers les commandes de Scilab. Bien entendu, le modèle à simuler doit être construit dans Scicos. Ce mode d'interaction entre Scilab – Scicos et le pilotage spécifié est connu sous le nom de « mode batch ». Cela nous a permis d'effectuer l'analyse statistique des simulations de Monte Carlo.

Le mode batch est possible grâce aux fonctions « scicosim » et « scicos_simulate ». La première fonction est une interface du simulateur Scicos. Elle a besoin des résultats de la compilation du modèle Scicos. Cette fonction est appelée de trois manières différentes: d'abord pour initialiser la simulation, ensuite pour l'exécuter et enfin pour la finir. Un des avantages d'utiliser cette fonction est qu'elle nous permet d'effectuer plusieurs simulations du modèle Scicos en changeant un paramètre d'un bloc ou une condition initiale chaque fois que chaque simulation est recommencée pour les traiter de manière statistique dans Scilab. La deuxième fonction, scicos_simulate, est une fonction qui a été construite sur la base de la fonction scicosim. Elle nous permet d'exécuter les simulations du modèle Scicos dans Scilab en utilisant la structure de donnée du diagramme ou modèle Scicos. Les principaux types de paramètres que nous pouvons modifier ou traiter en utilisant cette fonction sont les paramètres de valeur constant définis dans la partie nommée « context » dans Scicos.

2.3 Réalisation de l'automate stochastique hybride

Avant que le concept d'automate hybride n'ait été introduit par l'INRIA dans l'environnement Scicos, nous avions développé une première approche pour coupler le comportement des variables physiques d'un système dynamique avec le fonctionnement ou dysfonctionnement de ses composants en utilisant les blocs déjà existants dans Scicos [Pérez *et al.*, 2007a]. Nous montrons cette approche dans le paragraphe 2.3.2. Dans une deuxième approche, nous avons défini l'ASH [Pérez *et al.*, 2008a] sur la base de l'automate hybride (AH) proposé par [Najafi et Nikoukhah, 2007] et nous l'avons implémenté et adapté pour la plate-forme Windows.

Dans le chapitre suivant nous présenterons l'évaluation de la fiabilité de deux cas test effectuée par l'automate stochastique hybride à travers une simulation de Monte Carlo. Le premier cas test correspond au contrôle de la température d'un four par un contrôleur Proportionnel-Intégral (PI) ou par un contrôleur Tout ou Rien (TOR). Le deuxième cas test correspond au benchmark proposé par [Aldemir, 1987], [Dutuit *et. al.*, 1997], [Marseguerra et Zio, 1996], [Kermish et Labeau, 2000] et [Zhang *et. al.*, 2008]. Il s'agit du système de contrôle de niveau du liquide d'un réservoir.

2.3.1 Cas test 1 : un four et son système de contrôle de température

Le cas test que nous proposons a la prétention de permettre la prise en compte d'un grand nombre des problèmes relevant de la « fiabilité dynamique » de manière aussi réaliste que possible. Il s'agit d'un four dont la température évolue en fonction de l'énergie qui lui est transmise. Dans notre exemple nous supposons que l'énergie est électrique mais le modèle s'appliquerait à la combustion par exemple. Le système de contrôle contient deux boucles de régulation. La première inclut un contrôleur proportionnel et intégral (PI) dont le rôle est de contrôler la température du four en fonction de la température de référence en fournissant à chaque instant la puissance juste nécessaire. Elle correspond au fonctionnement normal du système, elle délivre l'énergie de manière continue. En cas de

défaillance de cette boucle de régulation, une boucle de secours de type tout ou rien (TOR) permet de maintenir la température du four aux alentours de la température de référence +/- ΔT en chauffant à pleine puissance ou en ne chauffant pas. Les deux boucles ne doivent évidemment pas fonctionner en même temps. Pour cela, un relais bascule ses deux contacts permettant ainsi d'activer soit le régulateur PI soit le régulateur TOR. L'ordre de basculement est donné par un système de détection dont le rôle est d'identifier les défaillances et les réparations et de réagir en commutant d'un régulateur à l'autre.

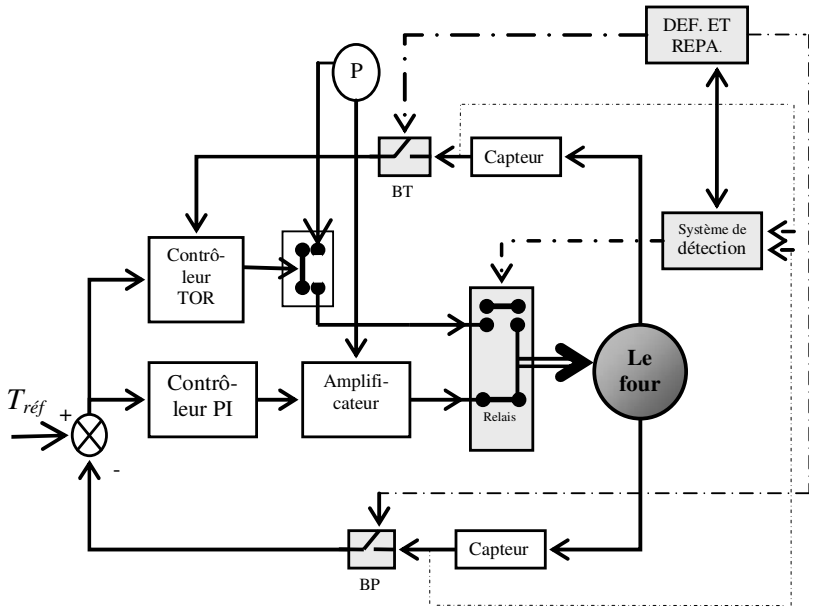


Figure 2.4 – Diagramme structurel du système de contrôle de la température d'un four.
$T_{réf}$ – Température de référence P – Alimentation de puissance BP – ouverture de la boucle du contrôleur PI
DEF. ET REPA. – défaillances et réparations BT – ouverture de la boucle du TOR

Le système fonctionne de la manière suivante : au démarrage la température x du four est contrôlée par le contrôleur PI. Au bout d'un certain temps aléatoire, le contrôleur tombe en panne (avec un taux λ_{PI}) et la température du four augmente rapidement. Le système de détection de franchissements de seuils détecte que la température du four a atteint une valeur dangereuse ($x \geq x_{max}$) déduisant ainsi que la température du four est hors contrôle. Il donne alors l'ordre au relais de basculer sur la boucle TOR. La boucle du contrôleur PI est maintenant ouverte et la boucle TOR fermée. La température du four est contrôlée maintenant par le TOR ($x_{infTOR} \leq x \leq x_{supTOR}$). Dès que le système de détection a repéré que la température était hors contrôle, il a donné l'ordre de basculer au relais vers la boucle du TOR et enclenché le processus de réparation du contrôleur PI (on considère une réparation de durée aléatoire μ_{PI}). Cependant, la possibilité de défaillance du régulateur TOR existe, après une durée également aléatoire (λ_{TOR}). Une fois que le contrôleur PI est réparé, le système de détection bascule le relais sur la boucle de celui-ci et ouvre la boucle

du TOR. La température du four est maintenant à nouveau régulée par le contrôleur PI. On inclut également le processus de réparation du TOR (μ_{TOR}).

Ce système est assurément un SDH, certains événements sont liés à l'évolution continue (seuils), d'autres sont aléatoires. L'hypothèse de défaillance d'une boucle caractérisée globalement par un taux peut être raffinée en sélectionnant les sources possibles. On peut ainsi utiliser des modèles de distribution très variés (défaillance des éléments chauffants, des capteurs, des régulateurs, etc.), dépendant éventuellement de variables continues ou discrètes (influence de la température ou du nombre de commutations sur la fiabilité des éléments chauffants, etc.). Dans un premier temps, l'organe de détection est supposé parfait mais il est possible de lui affecter une probabilité de succès par exemple. Notre travail à partir de ce cas test a abordé un certain nombre de ces problèmes.

2.3.2 Première approche

Dans notre première approche, nous avons construit un automate à états finis pour le système de contrôle de la température du four. Dans un souci de recherche de faisabilité, nous n'avons considéré seulement qu'un sous ensemble réduit des états théoriquement possibles. La figure 2.5 montre cet automate. Nous avons considéré que le four n'est pas défaillant. Les états discrets considérés du système sont :

- état 1 : les deux boucles fonctionnent, mais seule le PI est actif,
- état 2 : la boucle PI est défaillante mais toujours active car cela n'a pas été détecté,
- état 3 : la défaillance du PI est détectée, il est désactivé, le contrôleur TOR est activé et on lance la réparation du PI,
- état 4 : les deux boucles sont défaillantes mais la défaillance du TOR n'est pas détectée, il reste donc actif,
- état 5 : les deux boucles sont hors service (HS) et inactives, on lance les réparations des deux boucles.

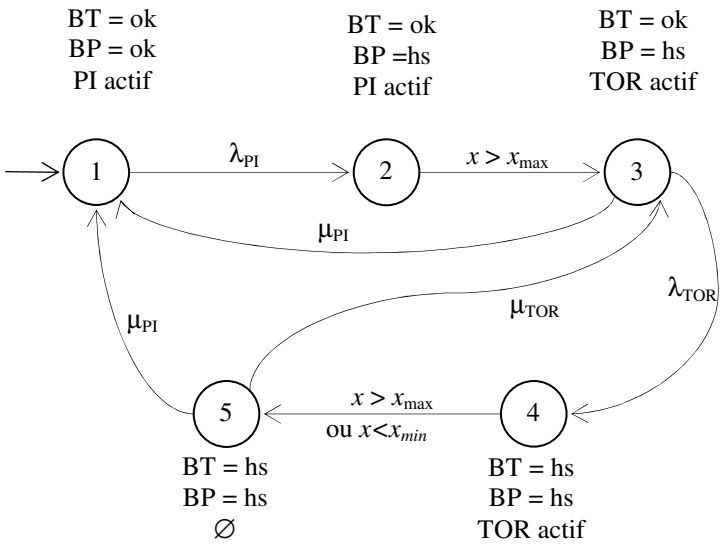


Figure 2.5 – Automate d'états du système de contrôle de la température d'un four.
BT – Boucle du TOR BP – Boucle du PI ok – fermée hs – ouverte

Les événements aléatoires λ_{PI}, λ_{TOR}, μ_{PI}, μ_{TOR} étiquetant les arcs du graphe (transitions) sont respectivement les défaillances (λ) et les réparations (μ) du contrôleur PI et du contrôleur TOR. Ils correspondent donc à des transitions stochastiques du système. Ils seront produits par un générateur aléatoire dans la simulation. x est la variable température du four. x_{max} et x_{min} sont respectivement les seuils de température maximale et minimale au-delà et en dessous desquelles le système de détection identifie la défaillance des boucles. Les transitions (2→3 ou 4→5) sont déterministes, elles correspondent aux franchissements de ces seuils par la température.

Nous avons modélisé dans Scicos, figure 2.6, l'automate à états finis présenté ci-dessus. Le modèle Scicos est principalement composé de quatre différentes parties :

1. la boucle du contrôleur PI (– – –),
2. la boucle du contrôleur TOR (·····),
3. le bloc Scicos « relay » (◯) et
4. le système de détection de franchissements de seuils/DEFA. et REPA. (◌).

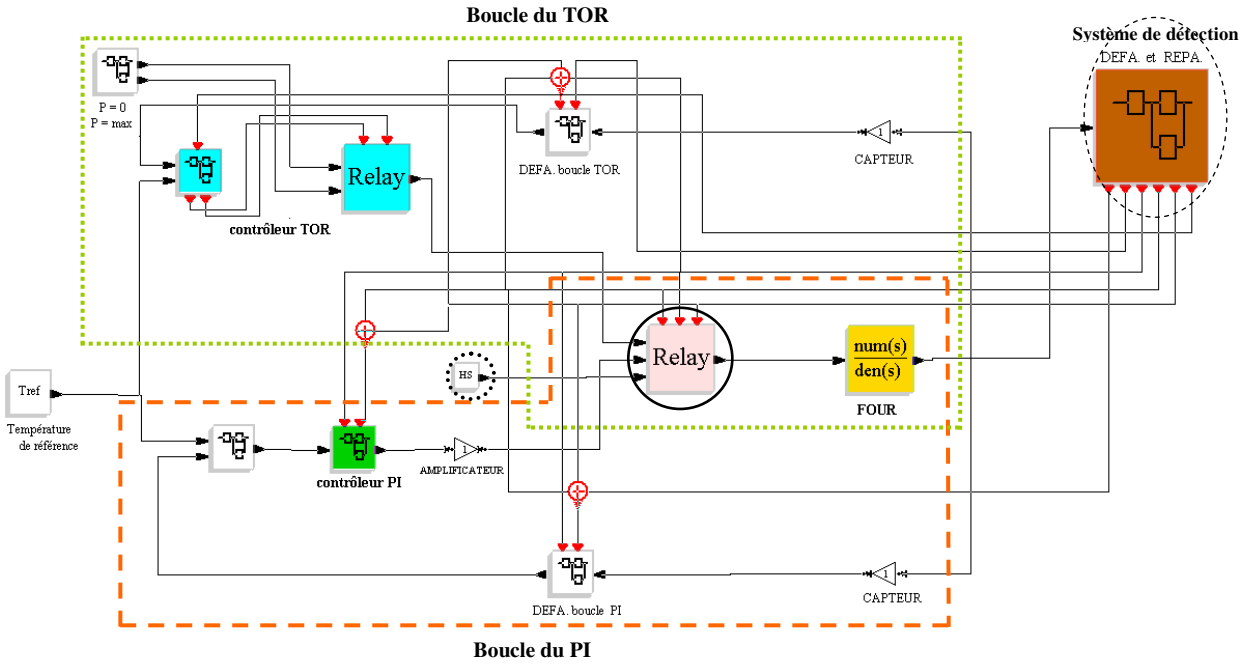


Figure 2.6 – Modèle Scicos du système de contrôle de la température du four [Pérez *et al.*, 2007a].

Le bloc HS (⋰), du modèle Scicos, correspond à l'état discret où le four n'est pas contrôlé ni par le PI ni par le TOR (état 5 de l'automate à états finis de la figure 2.5). Pour représenter le fonctionnement du contrôleur PI et du four dans le modèle, nous avons utilisé le bloc Scicos « continuous transfer fonction ». La fonction de transfert du contrôleur PI (F_{PI}) et la fonction de transfert du four (F_{four}) sont respectivement :

$$F_{PI}(s) = \frac{2.25s + 0.0015}{s} \tag{2.9}$$

$$F_{four}(s) = \frac{1}{1 + 1500s} \tag{2.10}$$

Ces choix sont bien sûr arbitraires et académiques. Toute autre fonction plus complexe est bien sûr possible, notamment pour le four (les modèles de fours industriels sont parfois relativement sophistiqués en raison des interactions avec la charge qui peut elle-même être dynamique comme dans les fours sidérurgiques). Le type de modèle aura finalement assez peu de répercussion sur les temps de la simulation.

Pour la génération aléatoire des défaillances et réparations, nous avons utilisé les blocs « Scifunc » et « Event Delay ». Le premier contient la fonction de répartition des temps de défaillance ou réparation aléatoires à estimer et le deuxième bloc génère l'événement correspondant à sa sortie, ce qui permettra le changement d'état

correspondant. La détection de la défaillance des composants ou le franchissement des seuils sont faits à travers le bloc « zero-crossing ». Ce bloc génère un événement à sa sortie lorsqu'il y a à son entrée un changement de signe entraîné par le franchissement de seuils des variables d'états continus ou des variables aléatoires. Ces événements produiront le basculement d'un contrôleur à l'autre.

La figure 2.7 montre un comportement simulé de la température du four sous contrôle alterné des deux boucles. Sur la même figure nous pouvons aussi apprécier le comportement de la température du four quand les contrôleurs sont défaillants (les états 2, 4 et 5). Les taux de défaillance ainsi que les taux de réparation sont constants (distribution exponentielle des durées de fonctionnement et de réparation). Les valeurs utilisées par la simulation sont :

- $x_{réf}$ = 190 °C ; x_{max} = 240 °C ; x_{min} = 150 °C ;

- $\lambda_{pi} = 20 \cdot 10^{-06}$ h^{-1} ; $\lambda_{Tor} = 8 \cdot 10^{-06}$ h^{-1} ; $\mu_{pi} = 14 \cdot 10^{-05}$ h^{-1} ; $\mu_{Tor} = 10 \cdot 10^{-05}$ h^{-1}.

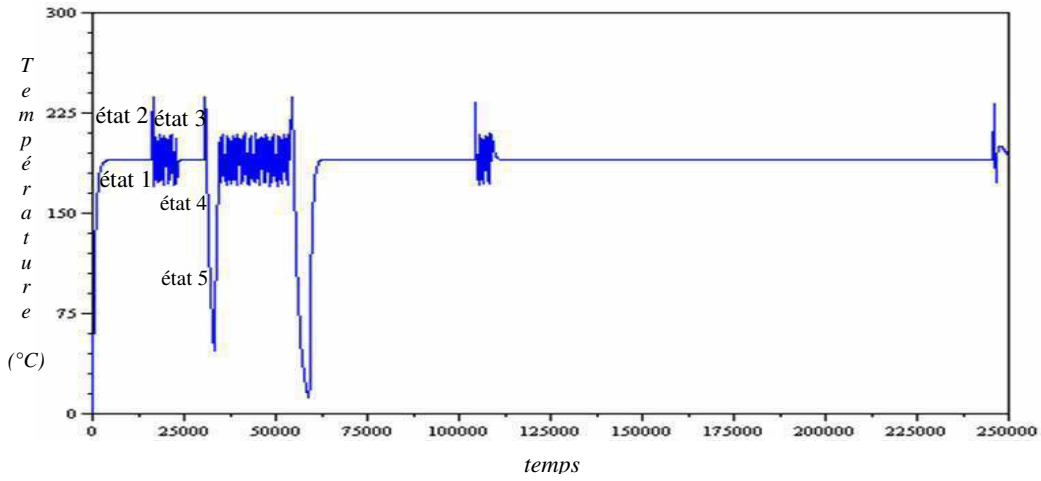


Figure 2.7 – Graphique du comportement de la variable physique température.

La modélisation et la simulation du système dynamique avec la boite à outils Scicos de Scilab a permis d'évaluer la fiabilité et la disponibilité de ce système. L'intérêt du formalisme est bien entendu de simplifier l'aspect mathématique en évitant le recours à des expressions analytiques complexes. Nous avons pu prendre en compte les interactions entre fonctionnement et dysfonctionnement pour une évaluation fine de ces paramètres de SdF. Cependant, le modèle Scicos obtenu est d'une complexité excessive par rapport à la simplicité de l'exemple choisi, ceci étant dû au fait que l'automate des états du système n'existe pas en tant qu'élément du modèle mais il est diffusé dans le modèle au moyen des boites de commutation. Par ailleurs, l'approche souffre du manque de caractère systématique et de réutilisation. Nous avons constaté qu'il n'est ni facile, ni pratique, ni efficace de modéliser ce petit système de cette manière. C'est pour tout cela que nous avons cherché un autre moyen d'implémenter l'ASH.

2.3.3 Implémentation de l'automate stochastique hybride

L'implémentation de l'ASH doit prendre en compte les différents éléments présentés à la fin du chapitre précédent (§ 1.7). Pour ce faire, nous avons commencé à traiter les problèmes relatifs à la fiabilité dynamique de façon progressive :

- Un automate à états finis nous permet de définir la partie événementielle. En plus, dans chaque état discret, nous spécifions le mode de fonctionnement continu du système. Cela nous amène à transformer l'automate à états finis en AH pour gérer la partie continue et la partie événementielle. L'AH est donc composé de différents sous-systèmes. A n'importe quel instant de temps, seulement un sous-système est actif et les autres restent inactifs. L'AH va piloter les événements qui se produisent dans la partie continue en fonction des événements qui se produisent dans la partie discrète.

- Compte tenu du caractère déterministe et stochastique des variables et des événements, nous avons introduit dans l'automate hybride l'aspect stochastique par l'utilisation d'un générateur aléatoire afin de modéliser les défaillances et les réparations du système.

- Une fonction de détection de passage par zéro ou « zero-crossing » nous permet de déterminer les franchissements des seuils soit des variables du processus, soit des variables stochastiques pour les défaillances et réparations des composants.

- L'ASH permet de prendre en compte différentes lois de probabilité ainsi que différents modes de défaillance pour les composants.

L'ASH est composé d'un automate et d'un générateur aléatoire liés aux différents modes de fonctionnement à travers un descripteur de modes. Il est présenté sur la figure 2.8.

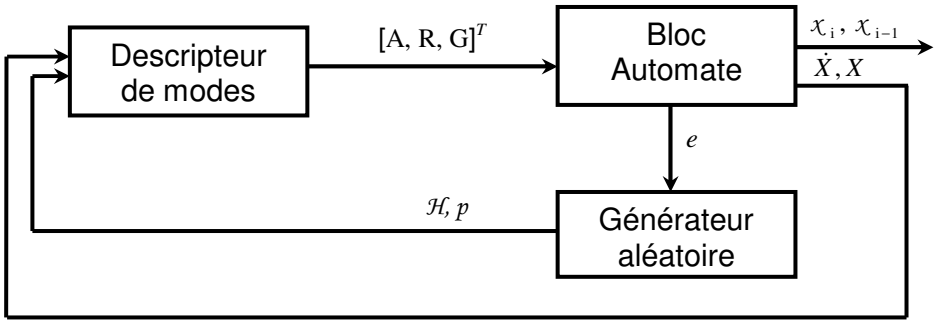


Figure 2.8 – Modèle de l'automate stochastique hybride.

2.3.3.1 Bloc automate

C'est un bloc Scicos (figure 2.9) [Najafi et Nikoukhah, 2007]. Il est constitué de *i* ports d'entrée (à gauche du bloc) et de deux ports de sortie (à droite du bloc). L'unique sortie *e* (en dessous du bloc) indique l'occurrence d'un quelconque des événements de l'automate (changement d'état).

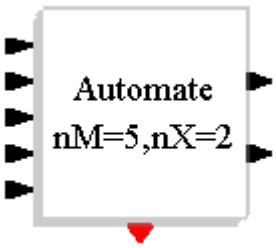


Figure 2.9 – Bloc : automate.

L'expression « nM » qui se trouve sur le bloc automate exprime le nombre d'états discrets définis (« number Modes ») et l'expression « nX » exprime le nombre de variables continues (« number of continuous-time states ») décrivant le système. En cliquant sur le bloc, nous pouvons accéder à la fenêtre d'interface (figure 2.10) afin de pouvoir définir les propriétés et le comportement du bloc. Nous pouvons spécifier le nombre d'états discrets, l'état initial, le nombre et le type (différentiel ou algébrique) des états continus affectant le système dans chaque état discret et les différentes transitions entre les différentes dynamiques. Les sauts entre les états discrets (« Jump from Mode ») sont spécifiés par le vecteur j_i. Par exemple pour $i = 3$, nous avons le vecteur $j_3 = [1;4]$. C'est-à-dire que le système peut basculer vers l'état 1 ou vers l'état 4 selon l'événement qui s'est produit le premier.

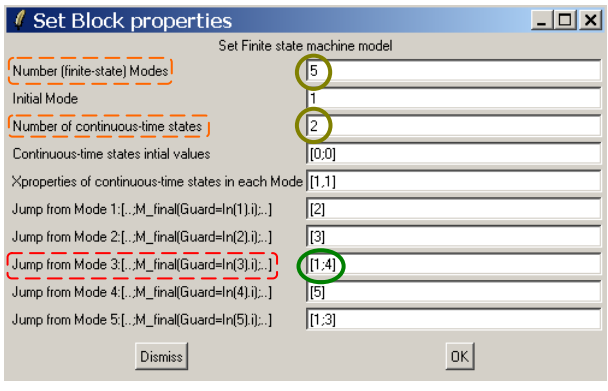


Figure 2.10 – Fenêtre d'interface du bloc automate.

L'avantage, que le bloc automate offre sur l'utilisation des blocs existant dans Scicos, est sa capacité à gérer les différents états discrets du système hybride. Chaque état discret est considéré comme un sous-système, l'ensemble des sous-systèmes étant défini dans le descripteur de modes. A n'importe quel instant, un sous-système seulement est actif et les autres sont inactifs (figure 2.11). Cette structure améliore la performance du résolveur numérique. Un deuxième avantage est la réduction des blocs « zero-crossing » (croisement par zéro) pour détecter les différentes transitions pour le système.

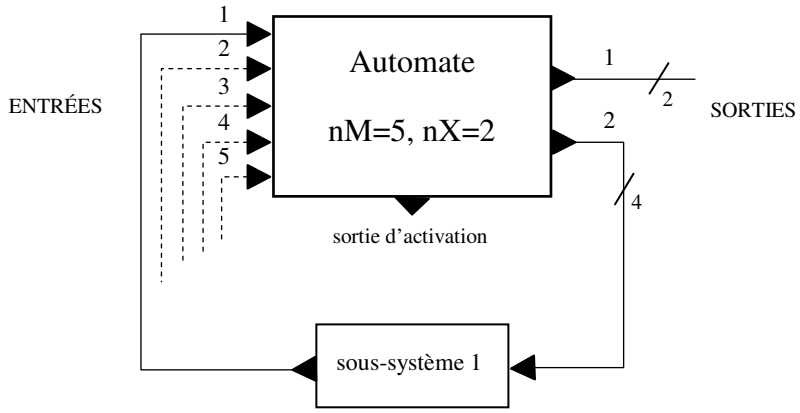


Figure 2.11 – Un bloc automate avec 5 états discrets et 2 variables continues [Najafi et Nikoukhah, 2007].

En ce qui concerne les entrées, il y a autant d'entrées que d'états discrets (ou modes de fonctionnement) permettant de décrire le système. Chaque entrée correspond à un vecteur donné par l'équation (2.11) qui sort de chaque sous-système du SDH.

$$V_i = \begin{bmatrix} A_i\left(\dot{X}, X, U, t\right) \\ \mathrm{R}_i\left(\dot{X}, X, U, t\right) \\ \mathrm{G}_i\left(\dot{X}, X, U, t\right) \end{bmatrix} \tag{2.11}$$

où le premier élément du vecteur correspond aux dynamiques A_i du système dans l'état courant, lesquelles sont exprimées par des équations différentielles de la forme :

$$0 = A_i\left(\dot{X}, X, U, t\right) \tag{2.12}$$

X et $\dot{X}$ étant respectivement les variables d'état continu et leurs dérivées, U le vecteur des variables d'entrée et t le temps. Le deuxième élément R_i, du vecteur V_i d'entrée, correspond aux valeurs qui sont utilisées pour réinitialiser les variables d'état continu lors

de l'entrée dans un nouvel état discret. Enfin, le troisième élément G_i, du vecteur V_i, correspond aux conditions de garde associées aux transitions de sortie de l'état x^i.

La condition de garde G_i est une fonction qui doit être exprimée de telle façon que, quand l'équation (2.13) est vérifiée, un événement s'est produit, figure 2.12. La condition de garde G_i doit s'exprimer de telle façon que le passage par zéro doit se faire avec une pente positive (G_i passe d'une valeur négative à une valeur positive).

$$G_i > 0 \tag{2.13}$$

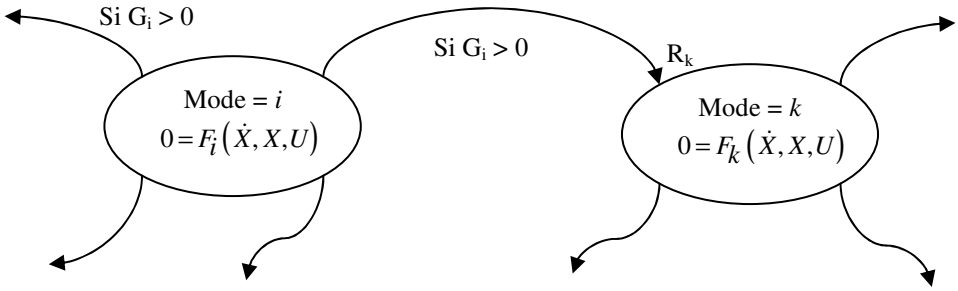


Figure 2.12 – Représentation graphique d'un automate hybride [Najafi et Nikoukhah, 2007].

Le croisement par zéro est le formalisme qu'utilise le bloc automate pour détecter ou déterminer les franchissements des seuils par les variables du processus ou par les variables stochastiques. La fonction « zero-crossing » du résolveur numérique est associée à la condition de garde G_i. Elle a la capacité de préciser l'instant de temps dans lequel la fonction G_i passe par zéro (figure 2.12).

En ce qui concerne les deux sorties de l'automate, la sortie 1 (figure 2.9) est un vecteur qui indique les numéros d'état discret courant x_i et précédent x_{i-1}. La sortie 2, qui sera liée au descripteur de modes spécifiquement à chaque sous-système, correspond au vecteur des variables d'état continu X et de leurs dérivées $\dot{X}$ (dans l'exemple, avec 2 variables continues, il y a 4 composantes).

La sortie d'activation du bloc automate (en bas) est activée lorsqu'une transition d'état discret se produit, c'est-à-dire que cette sortie génère un événement qui active le générateur aléatoire pour produire un tirage aléatoire dont la valeur sera transmise au descripteur de modes.

2.3.3.2 Générateur aléatoire

Le générateur aléatoire correspond à la structure temporisée stochastique $\mathcal{H}$ de la définition donnée (2.4). La figure 2.13 présente le bloc générateur aléatoire que nous avons développé. Les fonctions d'interface et de calcul du bloc sont présentées dans l'Annexe 1. La figure 2.14 montre la fenêtre d'interface de paramétrage du bloc générateur aléatoire que nous avons également développée. Elle permet de modifier rapidement les paramètres du modèle pour effectuer diverses simulations. Le générateur aléatoire réalise des tirages aléatoires correspondant aux transitions stochastiques vers les états concernés à travers ces sorties. Chaque fois qu'une transition d'état discret se produit la sortie d'activation du bloc automate génère un événement *e* activant le bloc générateur aléatoire à travers son entrée d'activation (au dessus du bloc figure 2.13). A ce moment le tirage des valeurs aléatoires se fait. Le bloc générateur aléatoire permet de choisir la loi de probabilité à utiliser : exponentielle, de Weibull ou normale. Si nous avons l'intérêt de changer de loi de probabilité, les fonctions de calcul et d'interface du bloc peuvent être modifiées.

Figure 2.13 – Générateur aléatoire.

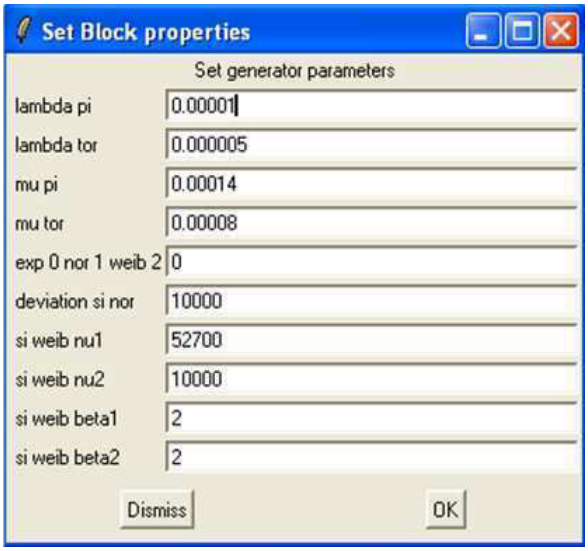


Figure 2.14 – Fenêtre d'interface du générateur aléatoire.

2.3.3.3 Descripteur de modes

Le descripteur de modes du modèle de l'ASH (figure 2.8) correspond aux différentes dynamiques continues du système ou aux sous-systèmes. Il y a autant de dynamiques continues que d'états discrets. Le descripteur a deux entrées : la première correspond aux variables physiques et leurs dérivées qui viennent du bloc automate et la deuxième entrée reçoit les valeurs aléatoires du tirage que produit le générateur aléatoire. La sortie du descripteur de modes contient l'ensemble des valeurs des variables continues associées à chaque état discret du système. Cet ensemble est transmis aux entrées du bloc automate. Chaque sortie de chaque état discret est un vecteur V_i donné par l'équation (2.4). Les blocs Scicos utilisés pour décrire les dynamiques continues sont les blocs « Mathematical Expression ». La dynamique continue est exprimée comme une expression mathématique Scilab (équation différentielle ordinaire) dans le bloc.

2.4 Composition parallèle d'automates

Afin de décrire de façon générale et complète le comportement d'un système dynamique, il est nécessaire et important de trouver un moyen qui nous permette d'obtenir de façon formelle et systématique un automate qui prenne en compte tous les aspects du système. Construire un tel automate à la main peut entraîner à la négligence d'un comportement particulier du système. La composition parallèle est le moyen qui va nous assurer que tous les états et toutes les transitions du système soient pris en compte.

A partir des automates simples ou élémentaires nous pouvons obtenir l'automate global du système. Cette opération permettra de synchroniser les différents automates en fonction des tous leurs événements. Dans une composition parallèle (ou synchronisation) l'occurrence d'un événement commun aux deux automates ne peut avoir lieu que si elle est possible simultanément dans les deux automates. Cela signifie que les deux automates sont synchronisés sur le même événement commun. La composition parallèle des automates permettra donc de déterminer la séquence d'exécution des événements par l'automate global. Ainsi, l'occurrence de tous les événements des automates simples sera synchronisée et formellement exprimée par l'automate global du système.

A la différence du produit des automates, la composition parallèle (synchronisation) prend en compte toutes les transitions des automates ($\mathcal{E}_1 \cup \mathcal{E}_2$). Le produit, en revanche, ne prend en compte que les transitions qui leurs sont communes ($\mathcal{E}_1 \cap \mathcal{E}_2$), figure 2.15.

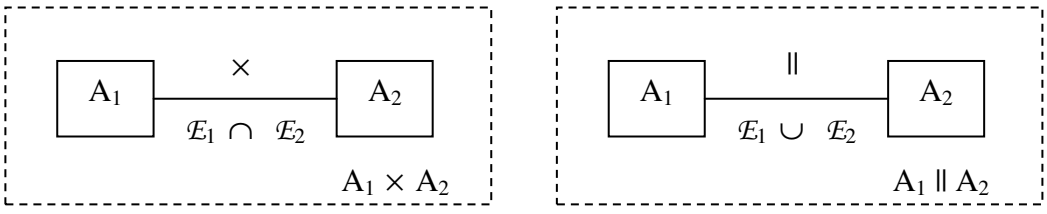


Figure 2.15 – Interconnexion de deux automates [Cassandras et Lafortune, 1999].

La composition parallèle de deux automates (ou encore synchronisation) notée $A_1 \| A_2$ est l'automate défini par [Cassandras et Lafortune, 1999] :

Soient deux automates $A_1 = (X_1, \mathcal{E}_1, f_1, \Gamma_1, x_{01}, X_{m1})$ et $A_2 = (X_2, \mathcal{E}_2, f_2, \Gamma_2, x_{02}, X_{m2})$ avec X_1, X_2 ensembles de leurs états, $\mathcal{E}_1, \mathcal{E}_2$ ensembles de leurs événements, f_1, f_2 leurs fonctions de transition, Γ_1, Γ_2 leurs fonctions des événements actifs, x_{01}, x_{02} leurs états initiaux et x_{m1}, x_{m2} leurs états finaux ou marqués.

$$A_1 \| A_2 := \mathrm{Ac}\left(X_1 \times X_2, \mathcal{E}_1 \cup \mathcal{E}_2, f, \Gamma_{1\|2}, (x_{01}, x_{02}), x_{m1} \times x_{m2}\right) \tag{2.14}$$

avec

$f((x_1, x_2), e)$: {Si $e \in \Gamma_1(x_1) \cap \Gamma_2(x_2)$ alors $f((x_1, x_2), e) := (f_1(x_1, e), f_2(x_2, e))$

Si $e \in \Gamma_1(x_1) \setminus \mathcal{E}_2$ alors $f((x_1, x_2, e) := (f_1(x_1, e), x_2)$

Si $e \in \Gamma_2(x_2) \setminus \mathcal{E}_1$ alors $f((x_1, x_2), e) := (x_1, f_2(x_2, e))$

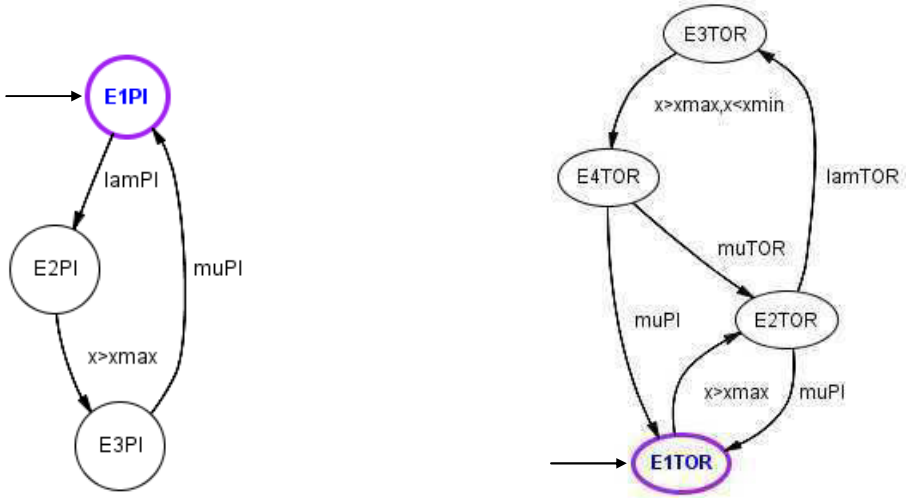
Sinon, $f((x_1, x_2), e)$ n'est pas définie}

La fonction des événements actifs de $A_1 \| A_2$ est définie par :

$$\Gamma_{1\|2}(x_1, x_2) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus \mathcal{E}_2] \cup [\Gamma_2(x_2) \setminus \mathcal{E}_1]$$

Dans cette opération, un événement commun (appartenant à $\mathcal{E}_1 \cap \mathcal{E}_2$) ne pourra être exécuté que si les deux automates l'exécutent simultanément. Les deux automates sont alors synchronisés sur les événements communs. Les autres événements ($e \in (\mathcal{E}_1 \setminus \mathcal{E}_2) \cup (\mathcal{E}_2 \setminus \mathcal{E}_1)$ peuvent être exécutés sans contrainte quand c'est possible. Si $\mathcal{E}_1 \cap \mathcal{E}_2 = \varnothing$ les deux automates s'exécutent indépendamment (concurrence).

Revenons à l'automate d'états du système de contrôle de la température du four, figure 2.5. Nous l'avons obtenu par composition parallèle à partir des automates élémentaires des contrôleurs PI et TOR, figure 2.16.



a) Automate du contrôleur PI b) Automate du contrôleur TOR

Figure 2.16 – Automates élémentaires du cas test.

Les états ainsi que les événements des contrôleurs PI et TOR sont présentés respectivement sur les tableaux 2.1 et 2.2.

Contrôleur PI	
États	**Événements**
E1PI : l'état initial ; le contrôleur est actif,	**lamPI :** défaillance du contrôleur
E**2PI :** le contrôleur est défaillant mais actif. Sa défaillance n'est pas encore détectée,	**x>xmax :** détection de la défaillance du contrôleur
E3PI : la défaillance du contrôleur est détectée. Il est inactif	**muPI :** réparation du contrôleur

Tableau 2.1 – Événements et états du contrôleur PI.

Contrôleur TOR	
États	**Événements**
E1TOR : l'état initial ; le contrôleur est inactif	**x>xmax :** détection de la défaillance du contrôleur PI
E2TOR : le contrôleur est actif	**muPI :** réparation du contrôleur PI,
E3TOR : le contrôleur devient défaillant, mais il est actif car sa défaillance n'est pas encore détectée	**lamTOR :** défaillance du contrôleur TOR
E4TOR : la défaillance du contrôleur est détectée, il est inactif	**muTOR :** réparation du contrôleur TOR
	x>xmax,x<xmin : détection de la défaillance du contrôleur TOR

Tableau 2.2 – Événements et états du contrôleur TOR.

Appliquant la définition donnée par l'équation (2.14) aux automates de la figure 2.15 nous avons :

- $\mathcal{E}_1$ = {lamPI,muPI,x>xmax}
- $\mathcal{E}_2$ = {muPI,lamTOR,muTOR,x>xmax, x>xmax,x<xmin}
- $\mathcal{E}_1 \cup \mathcal{E}_2 = \{\text{lamPI}, \text{muPI}, \text{lamTOR}, \text{muTOR}, x > \text{xmax}, x > \text{xmax}, x < \text{xmin}\}$
- $\mathcal{E}_1 \cap \mathcal{E}_2 = \{\text{muPI}, x > \text{xmax}\}$

État initial (E1PI,E1TOR) :

- Γ_1(E1PI) = {muPI) ; Γ_2(E1TOR) = {x>xmax}
- $\varnothing \in \Gamma_1(\text{E1PI}) \cap \Gamma_2(\text{E1TOR})$
- muPI $\in$ Γ_1(E1PI)\ $\mathcal{E}_2$
- x>xmax $\notin$ Γ_2(E1TOR)\ $\mathcal{E}_1$
- f((E1PI,E1TOR),muPI) = (f_1(E1PI,muPI),E1TOR) = (E2PI,E1TOR) - nouvel état

État (E2PI,E1TOR) :

- Γ_1(E2PI) = {x>xmax} ; Γ_2(E1TOR) = {x>xmax}
- x>xmax $\in$ Γ_1(E2PI) $\cap$ Γ_2(E1TOR)
- f((E2PI,E1TOR), x>xmax) = (f_1(E2PI,x>xmax),f_2((E1TOR,x>xmax)) =

  (E3PI,E2TOR) – nouvel état

État (E3PI,E2TOR) :

- Γ_1(E3PI) = {muPI} ; Γ_2(E2TOR) = {muPI,lamTOR}
- muPI $\in$ Γ_1(E3PI) $\cap$ Γ_2(E2TOR)
- lamTOR $\in$ Γ_2(E2TOR)\ $\mathcal{E}_1$
- f((E3PI,E2TOR,muPI) = (f_1(E3PI,muPI),f_2(E2TOR,muPI)) = (E1PI,E1TOR) – état initial
- f((E3PI,E2TOR,lamTOR) = (E3PI,(f_2(E2TOR,lamTOR)) = (E3PI,E3TOR) – nouvel état

État (E3PI,E3TOR) :

- Γ_1(E3PI) = {muPI} ; Γ_2(E3TOR) = {x>xmax,x<xmin}
- muPI $\notin$ Γ_1(E3PI)\ $\mathcal{E}_2$
- x>xmax,x<xmin $\in$ Γ_2(E3TOR)\ $\mathcal{E}_1$
- f((E3PI,E3TOR), x>xmax,x<xmin) = (E3PI,f_2(E3TOR, x>xmax,x<xmin)) = (E3PI,E4TOR) – nouvel état

État (E3PI,E4TOR) :

- Γ_1(E3PI) = {muPI} ; Γ_2(E4TOR) = {muPI,muTOR}
- muPI $\in$ Γ_1(E3PI) $\cap$ Γ_2(E4TOR)
- muTOR $\in$ Γ_2(E4TOR)\ $\mathcal{E}_1$
- f((E3PI,E4TOR),muPI) = (f_1(E3PI,muPI),f_2(E4TOR,muPI)) = (E1PI,E1TOR) – état initial

- f((E3PI,E4TOR),muTOR) = (E3PI,f_2(E4TOR,muTOR)) = (E3PI,E2TOR) – état déjà connu

Cette composition parallèle des automates nous donne l'automate global de la figure 2.17 lequel correspond bien à celui de la figure 2.5.

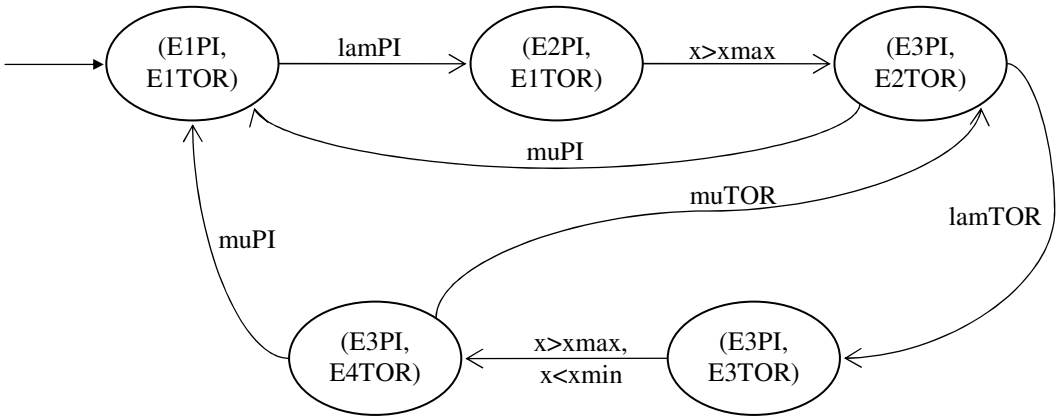


Figure 2.17 – Automate global du cas test.

2.5 Simulation de Monte Carlo et convergence

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 [Incera, 2001].

La simulation de Monte Carlo est une méthode numérique basée sur le tirage de nombres aléatoires. La quantité que nous désirons 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 au cours de ce que nous appelons un scénario ou histoire.

La quantification de la quantité recherchée, par exemple la fiabilité ou la probabilité d'apparition d'un événement redouté, 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, nous pouvons associer un estimateur de type binaire à cette probabilité et incrémenter un compteur d'une unité pour chaque histoire dans laquelle l'événement 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 [Labeau et Kermisch, 2001].

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 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 la simulation des événements rares. Un nombre important d'histoires est nécessaire pour voir apparaître un nombre suffisant d'occurrence de l'événement redouté, impliquant des temps de simulation importants, d'autant plus que l'on souhaite obtenir un intervalle de confiance acceptable.

Le principe de la simulation de Monte Carlo est donc le suivant [Niel et Craye, 2002] :

- établissement d'un modèle fonctionnel et dysfonctionnel du système à étudier ;
- animation du modèle par tirage de nombres au hasard pour réaliser des histoires possibles du système étudié ;
- récupération et exploitation des résultats statistiques intéressants.

Du principe ci-dessus, il résulte plusieurs choses très importantes :

- Le modèle doit être suffisamment détaillé pour représenter tous les modes de fonctionnement et de dysfonctionnement significatifs ;
- Le modèle doit être capable de représenter tous les changements d'état pour tenir compte des événements aléatoires mais aussi déterministes (liés à l'évolution continue) ;
- Un générateur de nombres aléatoires de bonne qualité est nécessaire pour animer le modèle ;
- Comme nous parlons de statistiques, cela sous-entend que de nombreuses histoires devront être réalisées afin de rassembler des échantillons statistiquement représentatifs et suffisants pour la précision recherchée.

Nous pensons avoir construit un outil de simulation de Monte Carlo pour l'évaluation des grandeurs de la sûreté de fonctionnement en contexte dynamique, répondant à ces critères.

De nombreux outils de simulation de Monte Carlo utilisés en SdF reposent sur le principe de la répétition d'un grand nombre d'histoires fixé à l'avance. Les résultats peuvent alors dépendre de ce nombre. Nous avons préféré utiliser un critère de convergence plutôt qu'un nombre a priori. L'arrêt de la simulation sera décidé lorsque deux conditions seront vérifiées :

- l'apport d'une $i^{ème}$ histoire simulée sur la valeur du résultat est inférieur à la précision cherchée. Ce critère est donné par l'équation suivante :

$$\left| \frac{v_{m(i)} - v_{m(i-1)}}{v_{m(i)}} \right| \leq \varepsilon \tag{2.15}$$

où $v_{m(i)}$ et $v_{m(i-1)}$ représentent respectivement les valeurs moyennes de la variable mesurée après i histoires et après $(i-1)$ histoires simulées. ε correspond à la précision relative de calcul désirée sur cette variable,

- la condition précédente est vérifiée pour un nombre k suffisant d'histoires parmi les i histoires simulées. L'équation 2.16 exprime ce critère. θ est la « probabilité » pour que le nombre total d'histoire assure un résultat avec la précision ε donnée :

$$\frac{k}{i} \geq \theta \tag{2.16}$$

Sur la figure 2.18 nous présentons l'organigramme général de la séquence de Simulation de Monte Carlo que suit notre programme pour évaluer les indices de performances de la sûreté de fonctionnement. La même figure montre aussi les conditions mentionnées ci-dessus arrêtant la simulation.

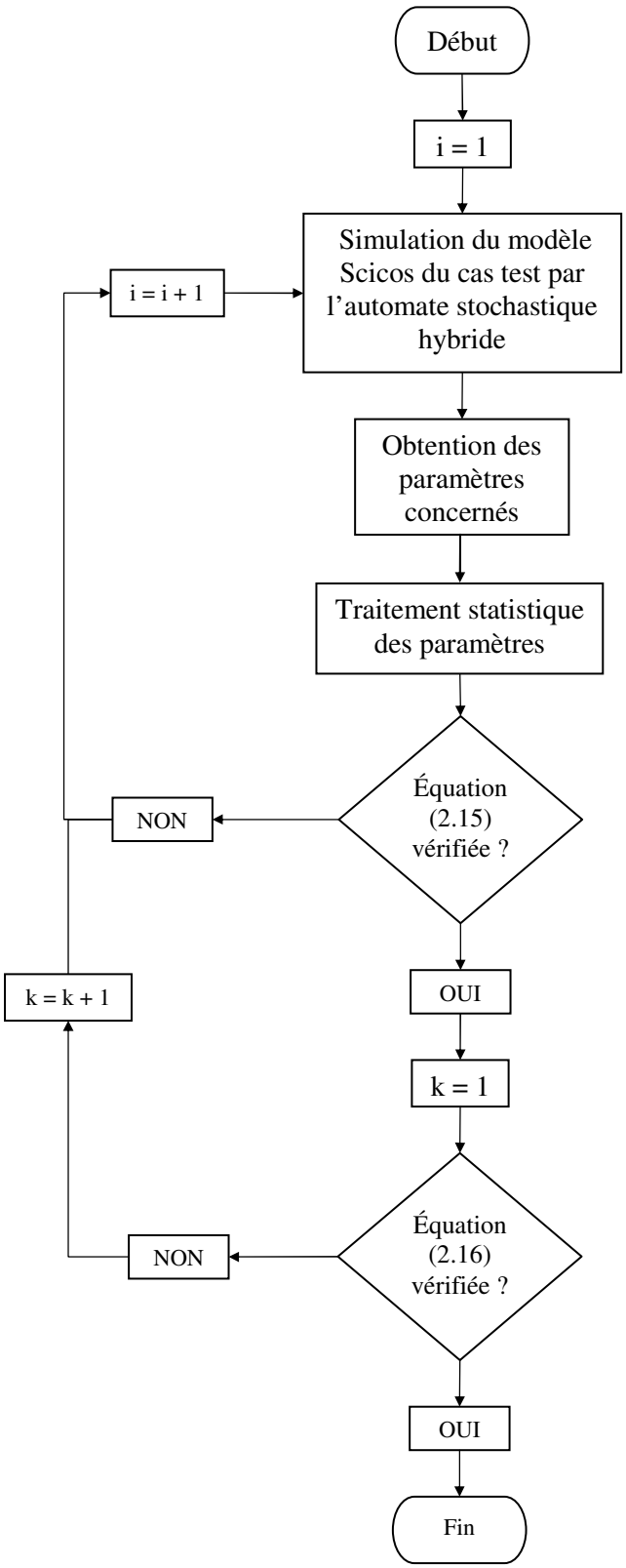


Figure 2.18 – Organigramme du Programme de Simulation de Monte Carlo.

2.6 Conclusion

La théorie des automates à états finis nous permet de définir de manière sûre et complète le comportement d'un système dynamique hybride. Pour évaluer sa fiabilité nous avons défini un automate stochastique hybride capable de surmonter les problèmes posés par la fiabilité dynamique et d'accéder à l'évaluation des grandeurs de la sûreté de fonctionnement par une simulation de Monte Carlo.

Le cas test du système de contrôle de la température d'un four à travers les contrôleurs PI et TOR nous permettra d'illustrer cette approche et d'en démontrer la faisabilité. Les avantages de l'approche dans le contexte de la sûreté de fonctionnement seront constatés. Le cas test du système de contrôle de niveau du liquide d'un réservoir nous permettra de confronter nos résultats avec les résultats obtenus par d'autres approches.

Chapitre 3

Évaluation des indicateurs de sûreté de fonctionnement des cas tests

3.1 Fiabilité dynamique et cas tests

Une caractéristique d'abord importante de nombreux systèmes industriels est que leur comportement, comme par exemple la réponse à une perturbation, change en fonction du temps en raison des interactions entre les composants de ce système ou avec l'environnement. Chaque comportement donné du système est défini par les lois de la physique qui lui sont propres. Le passage d'un comportement à un autre peut être dû à plusieurs causes : l'intervention humaine, l'action de l'organe de contrôle agissant sous l'influence des variables physiques qui décrivent l'état du système (détection d'une alarme…), une discontinuité propre au système (diode dans un circuit, couplage intermittent…) ou encore une défaillance de composant (auquel cas le système peut lui-même être dans un état de défaillance). En plus de l'hybridité, due à l'interaction entre la partie continue et la partie à événements discrets, il faut donc tenir compte du caractère stochastique du système imposé par les défaillances des composants ou par les incertitudes sur la connaissance du système. Pour cette raison, actuellement, il est très important de pouvoir évaluer la fiabilité de ces types de systèmes.

Dans ce chapitre nous allons présenter l'évaluation de la fiabilité de deux cas test à travers l'automate stochastique hybride. Le premier a été déjà présenté dans §2.3.1. Il s'agit d'un four contrôlé en température par un contrôleur PI ou par un contrôleur TOR. Le deuxième cas test correspond au benchmark présenté par [Aldemir, 1987], [Dutuit *et al.*, 1997], [Marseguerra et Zio, 1996], [Kermish et Labeau, 2000a] et [Zhang *et al*., 2008] : le réservoir de liquide contrôlé en niveau. De cette façon nous pourrons confronter notre approche et nos résultats au benchmark.

3.2 Cas test 1 : système de contrôle de la température d'un four

Le système qui contrôle la température d'un four est un système dynamique hybride simple, cependant, il nous permet de traiter la plupart des problèmes posés par la fiabilité dynamique. Nous avons traité deux cas généraux :

1. celui où les taux de défaillance des composants sont constants (les durées de fonctionnement suivent une loi exponentielle),

2. celui où les modes de vieillissement des composants sont multiples selon l'état du système avec des durées de fonctionnement des composants suivant différentes lois de probabilité.

3.2.1 Taux de défaillance constant

3.2.1.1 Loi exponentielle

La plupart des méthodes et techniques qui abordent le problème de la fiabilité dynamique utilisent pour exprimer le taux de défaillance des composants une distribution exponentielle. Ce modèle est le plus utilisé car il permet une simplification pour le calcul de fiabilité de systèmes composés de nombreuses entités différentes. En outre, le passé de tels systèmes n'affecte pas leur évolution future. C'est-à-dire, un composant n'a aucune mémoire de son passé. Le fait que ce composant ait vécu un nombre d'années ne change pas son taux de défaillance. Cette loi est appliquée, par exemple, aux processus de Poisson et aux processus Markoviens.

Ainsi, la loi exponentielle représente correctement la distribution des durées de vie lorsque :

- les défaillances sont indépendantes et imprévisibles, et dont la génération obéit à un processus de Poisson,

- le taux de défaillance est constant et noté λ.

C'est le cas des composants électroniques et de nombreux composants électriques en période utile. Cette loi est caractérisée par les relations suivantes :

- taux de défaillance :

$$\lambda(t) = \lambda = \text{constante} \tag{3.1}$$

- fonction de répartition :

$$F(t) = 1 - e^{-\lambda t} \tag{3.2}$$

- fiabilité :

$$R(t) = 1 - F(t) = e^{-\lambda t} \tag{3.3}$$

- densité de probabilité :

$$f(t) = -\frac{dR(t)}{dt} = \lambda e^{-\lambda t} \tag{3.4}$$

- MTTF :

$$MTTF = \int_0^\infty R(t)dt = \int_0^\infty e^{-\lambda t}dt = \left[-\frac{1}{\lambda}e^{-\lambda t}\right]_0^\infty = \frac{1}{\lambda} \tag{3.5}$$

La fonction de répartition *F(t)* est égale à la défiabilité d'un composant et la densité de probabilité de défaillance d'un composant est la dérivée de la fonction de répartition *F(t)* dans les domaines où elle est continue. Cela veut dire que *f(t)dt* est la probabilité inconditionnelle que la défaillance d'un composant intervienne entre *t* et *t* +*dt*. $\lambda.dt$ est par contre la même probabilité mais sachant que la défaillance ne s'est pas encore produite à l'instant *t*.

3.2.1.2 Description du comportement du système

Le système de contrôle de la température du four, du point de vue fiabiliste, a deux composants : le contrôleur PI et le TOR. Nous considérerons donc leurs taux de défaillance et de réparation sont constants.

Le comportement du système de contrôle de la température du four par le contrôleur Proportionnel-Intégral (PI) ou par le contrôleur Tout ou Rien (TOR) est résumé par la figure 3.1.

- état 1, le contrôleur PI est actif et il contrôle la température du four,

- état 2, le contrôleur PI est actif mais défaillant alors que le contrôleur TOR n'est pas sollicité ; le contrôleur TOR est donc inactif,

- états 3 et 4, le contrôleur TOR est actif et le contrôleur PI est en réparation,

- états 5 et 6, le TOR est défaillant mais toujours actif. Le temps de détection de la défaillance du TOR est déterministe et on le souhaite aussi bref que possible. L'occurrence de l'événement « fin de réparation du PI » pendant ce temps est considéré comme hautement improbable. En conséquence, nous négligeons l'occurrence de cet événement lors du séjour du système dans les états 5 et 6.

- état 7, la défaillance du TOR est détectée par le système de détection de

franchissements de seuils. Dans cet état les deux boucles sont défaillantes et inactives, mais le PI est en réparation.

- état 8, le PI est réparé et il est maintenant actif. Le TOR est maintenant en réparation.

- état 9, le PI est actif mais défaillant alors que le TOR est encore en réparation.

Les transitions : 2→3, 3→4, 4→3, 5→7, 6→7 et 9→7 sont déterministes, elles correspondent aux franchissements de seuils de la température. Les autres transitions sont stochastiques elles correspondent aux défaillances ou aux réparations des contrôleurs.

L'automate a donc 9 états discrets, l'état 1 étant l'état initial. Dans chaque état discret il y a une équation différentielle de la forme donnée par (2.12) laquelle représente la dynamique continue du système. Sur chaque transition sont indiqués les paramètres *e*, G et R.

L'automate stochastique hybride de la figure 3.1 décrit un comportement plus complet du système par rapport à celui montré à la figure 2.5. Nous pouvons distinguer les compléments suivants :

- les états 1 et 2 sont inchangés.

- l'ancien état n°3 dans lequel le contrôleur TOR était actif est remplacé par les états 3 et 4 afin de décrire le fonctionnement de celui-ci. Dans l'état 3 le four chauffe et dans l'état 4 il refroidit.

- les états 5 et 6 indiquant la défaillance du TOR pendant la chauffe à pleine puissance et pendant la chauffe à puissance minimale (les taux de défaillance peuvent effectivement être différents). Dans ces états, la défaillance n'est pas encore détectée, le contrôleur est toujours actif.

- l'état 7 correspond à l'ancien état 5 (les deux régulateurs sont inactifs, le système est défaillant).

- l'état 8 dans lequel le PI est actif après sa réparation.

- l'état 9 dans lequel le PI est défaillant mais toujours actif et le TOR n'est pas encore réparé.

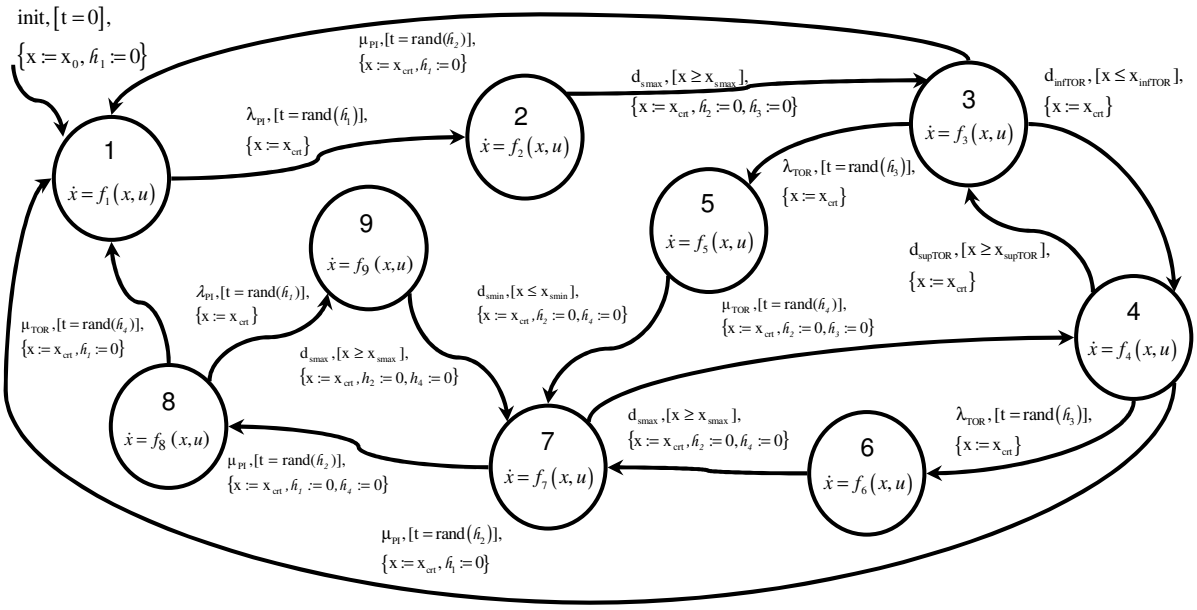


Figure 3.1 – Automate stochastique hybride du système de contrôle de la température d'un four. Pour chaque transition les conditions de garde sont indiquées entre crochets et les réinitialisations entre accolades.

Partant de la définition de l'automate stochastique hybride, équation (2.4), nous avons les expressions suivantes :

$\mathcal{X} = \{1,2,3,4,5,6,7,8,9\}$,

$\mathcal{E} = \{\lambda_{PI}, \lambda_{TOR}, \mu_{PI}, \mu_{TOR}, d_{smin}, d_{smax}, d_{infTOR}, d_{supTOR}\}$,

avec

- λ_{PI}, λ_{TOR}, μ_{PI}, *et* μ_{TOR} sont respectivement les taux de défaillance du PI et du TOR ainsi que leurs taux de réparation. Ces taux correspondent aux transitions stochastiques du système. Les événements correspondants sont produits par le générateur aléatoire pendant la simulation.

- d_{smin} *et* d_{smax} sont les événements de franchissement des seuils de température minimum et maximum au-delà desquelles le système de détection de franchissements de seuils identifie la défaillance des contrôleurs PI et TOR.

- d_{infTOR} *et* d_{supTOR} sont les événements de franchissement des seuils de commutation du régulateur TOR. Quand il détecte le passage de la température sous le seuil d_{infTOR} il applique toute la puissance de chauffe au four et quand il détecte le passage au dessus du seuil d_{supTOR} il coupe la puissance.

$G = \{t = rand(h_1);\ t = rand(h_2);\ t = rand(h_3);\ t = rand(h_4);\ x \le x_{smin};\ x \ge x_{smax};$
$x \le x_{infTOR};\ x \ge x_{supTOR}\}$,

$X = \{x\}$, représente la variable physique du système : la température,

$A : \mathcal{X} \times X \to \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\}$,

$R = \{x = x_{crt}\}$. La valeur de la température x, à l'entrée dans chaque état, est la même que celle atteinte quand le système a quitté l'état précédant (température courante x_{crt}). Par exemple, $R = \{h_2 := 0\}$ représente la réinitialisation de l'horloge h_2 qui modélise le temps de réparation du contrôleur PI,

$\mathcal{H} = \{h_1, h_2, h_3, h_4\}$ où h_1 et h_2 représentent respectivement le temps de bon fonctionnement et le temps de réparation du contrôleur PI. h_3 et h_4 représentent respectivement le temps de bon fonctionnement et le temps de réparation du contrôleur TOR,

$\mathcal{F}(h) = 1 - e^{-\lambda h}$. Nous avons utilisé la loi exponentielle pour h_i, $i = 1 \ldots 4$,

u est le vecteur des entrées, il définit :

- u_{ref} : la température de référence,
- u_{map} : la température asymptotique à puissance maximale,
- u_{mip} : la température asymptotique à puissance minimale,
- u_s : la température ambiante.

Pour le cas test il n'y a pas besoin d'utiliser p, parce il n'y a pas de transition en conflit.

Le tableau 3.1 définit la fonction de transition de l'automate : pour chaque état, l'état discret futur dépend de l'événement associé à l'arc et de la condition de garde, l'état continu étant réinitialisé par la fonction R associée à cet arc.

État origine	***e***	**Condition de garde [G]**	**Réinitialisation {R}**	**État but**
1	λ_{PI}	$[t = rand(h_1)]$	$\{x := x_{crt}\}$	2
2	d_{smax}	$[x \ge x_{smax}]$	$\{x := x_{crt}, h_2 := 0, h_3 := 0\}$	3

3	μ_{PI} d_{infTOR} λ_{TOR}	$[t = rand(h_2)]$ $[x \le x_{infTOR}]$ $[t = rand(h_3)]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$	1 4 5
4	μ_{PI} d_{supTOR} λ_{TOR}	$[t = rand(h_2)]$ $[x \ge x_{supTOR}]$ $[t = rand(h_3)]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$	1 3 6
5	d_{smin}	$[x \le x_{smin}]$	$\{x := x_{crt}, h_2 := 0, h_4 := 0\}$	7
6	d_{smax}	$[x \ge x_{smax}]$	$\{x := x_{crt}, h_2 := 0, h_4 := 0\}$	7
7	μ_{PI} μ_{TOR}	$[t = rand(h_2)]$ $[t = rand(h_4)]$	$\{x := x_{crt}, h_1 := 0, h_4 := 0\}$ $\{x := x_{crt}, h_2 := 0, h_3 := 0\}$	8 4
8	μ_{TOR} λ_{PI}	$[t = rand(h_4)]$ $[t = rand(h_1)]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$	1 9
9	d_{smax}	$[x \ge x_{smax}]$	$\{x := x_{crt}, h_2 := 0, h_4 := 0\}$	7

Tableau 3.1 – Définition de la fonction de transition de l'automate stochastique hybride

3.2.1.3 Paramètres pour la modélisation et la simulation

Les valeurs choisies pour les paramètres n'ont qu'une justification pédagogique. Notre objectif est seulement de montrer la faisabilité de l'étude. Ces paramètres sont:

- x_{smax} = 240 °C ; x_{smin} = 140 °C ; x_{infTOR} = 170 °C ; x_{supTOR} = 210 °C ;

- $\lambda_{PI} = 13 \cdot 10^{-05}\ h^{-1}$; $\lambda_{TOR} = 8 \cdot 10^{-05}\ h^{-1}$; $\mu_{PI} = 21 \cdot 10^{-03}\ h^{-1}$; $\mu_{TOR} = 14 \cdot 10^{-03}\ h^{-1}$

Les équations différentielles associées aux différents états discrets décrivant l'évolution de la température sont :

- états 1 et 8 :

$$\dot{x} + 0.0015x - 0.0015u_{ref} = 0 \tag{3.6}$$

Cette équation tient compte des paramètres physiques du four et des coefficients P et I du contrôleur.

- états 2, 4, 6 et 9 :

$$1500\dot{x} + x - u_{map} = 0 \tag{3.7}$$

Cette équation représente l'évolution libre de la température du four alimenté à pleine puissance.

- états 3 et 5 :

$$1500\dot{x} + x - u_{mip} = 0 \tag{3.8}$$

Cette équation représente l'évolution libre de la température du four alimenté à puissance minimale.

- état 7 :

$$1500\dot{x} + e^{1/1500}x - u_s = 0 \tag{3.9}$$

Cette équation représente l'évolution libre de la température du four non alimenté (aucun régulateur actif).

Où :

- u_{ref} = 190°C (température de référence),
- u_{map} = 300°C (température asymptotique à puissance maximale),
- u_{mip} = 25°C (température asymptotique à puissance minimale),
- u_s = 25°C (température ambiante).

3.2.1.4 Résultats

3.2.1.4.1 Le modèle et la simulation du système dynamique

La boîte à outils Scicos de Scilab offre une structure modulaire pour construire des systèmes dynamiques hybrides en utilisant un éditeur de blocs-diagrammes. La figure 3.2 présente donc le modèle Scicos du système dynamique hybride du contrôle de la température du four dont le comportement a été présenté par l'automate stochastique hybride de la figure 3.1. Ce modèle Scicos comporte la même structure que le modèle de l'automate stochastique hybride de la figure 2.8. Le descripteur de modes est constitué de 9 blocs Scicos. Chaque bloc Scicos correspond aux 9 états discrets de l'automate stochastique hybride du système. La première entrée de chaque bloc Scicos du descripteur de modes correspond au vecteur $[x,\ \dot{x}]$. Les autres entrées des blocs Scicos du descripteur de modes sont les valeurs aléatoires concernant les défaillances et les réparations des composants.

La figure 3.3 montre en détail les blocs Scicos de l'état 3 du descripteur de modes. La première entrée du bloc « Mux », située à l'extrême droite, correspond aux équations différentielles A_3 définissant le comportement dynamique du système dans l'état courant. La deuxième entrée correspond à la fonction de réinitialisation $R_3 = \{x = x_{crt}\}$ et la troisième aux conditions de garde G_3 associées aux transitions de sortie de l'état discret.

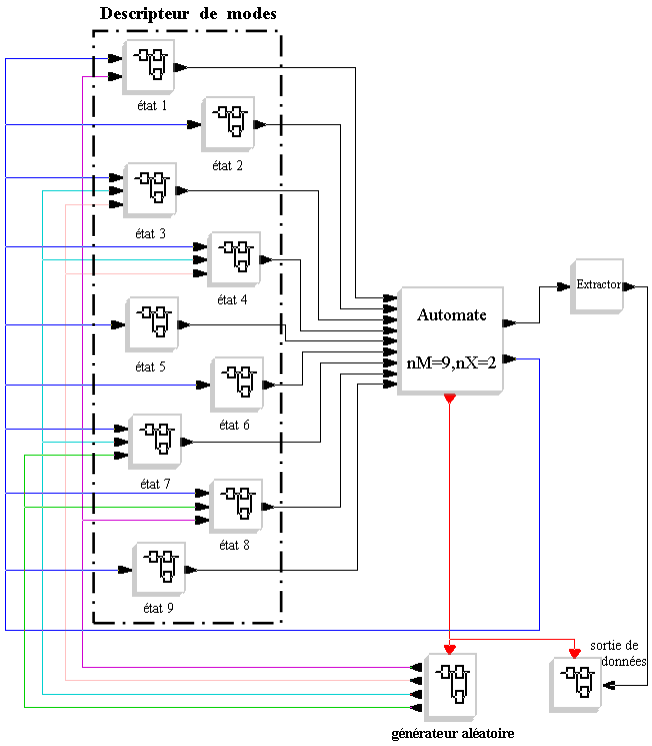


Figure 3.2 – Modèle Scicos du système hybride (cas test).

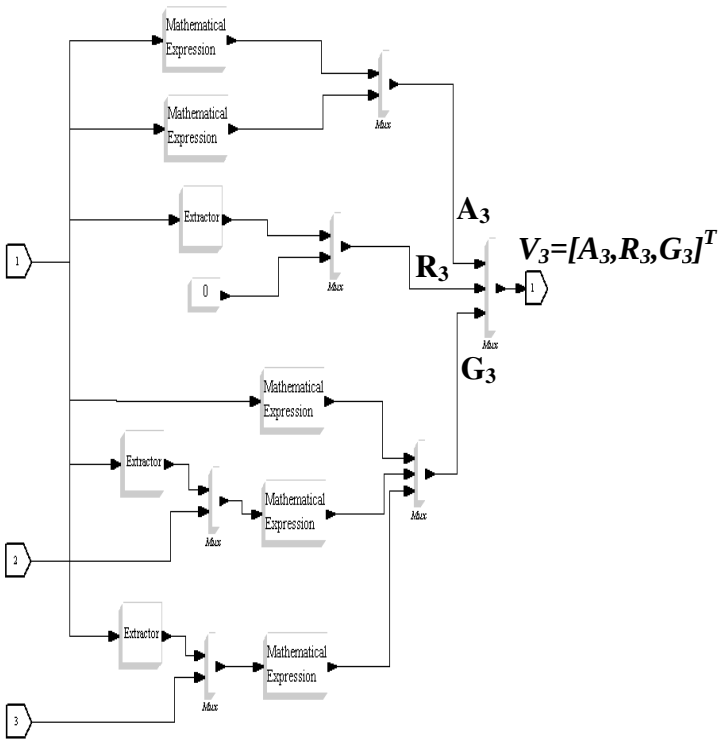


Figure 3.3 – État 3 discret du modèle Scicos du cas test.

Un échantillon de la simulation du système est montré par le graphique de la figure 3.4. La courbe du haut montre l'évolution de l'état courant du système au cours du temps. Cette évolution obéit au comportement défini par l'automate stochastique hybride du système. La courbe du bas présente l'évolution de la température du four. Cette dernière courbe montre la réponse à l'échelon de référence au démarrage, puis la défaillance de la boucle PI (à t_1) identifiée par le passage du seuil de danger (à t_2) et ensuite la régulation TOR (de t_2 à t_3). On peut aussi voir la défaillance du TOR (à t_3). La température monte à nouveau vers le seuil de danger (à t_4) détecté par le système de détection de franchissement de seuils qui bascule le relais. L'automate est alors dans l'état 7, où ni le contrôleur PI, ni le TOR ne contrôlent la température. Le four est débranché, la température chute vers la valeur de la température ambiante jusqu'à la réparation du contrôleur PI (à t_5) qui reprend le contrôle et ainsi de suite. Bien entendu, il aurait pu arriver que la réparation du TOR arrive avant celle du contrôleur PI.

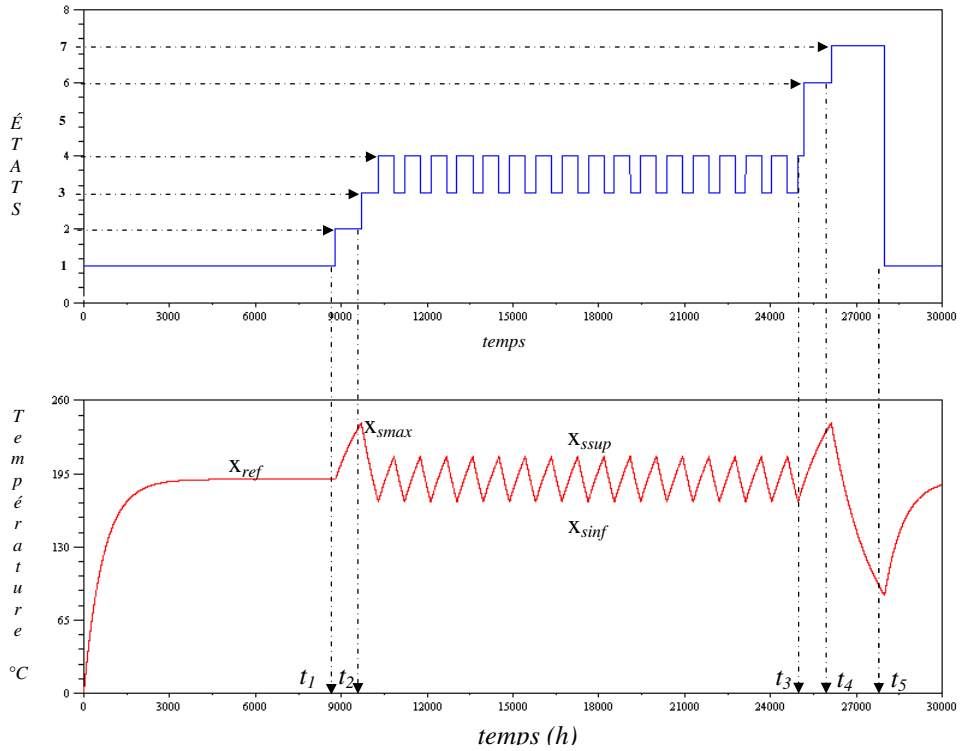


Figure 3.4 – Simulation du système hybride avec l'automate stochastique hybride.

3.2.1.4.2 Évaluation des grandeurs de la sûreté de fonctionnement

Pour obtenir les grandeurs de la sûreté de fonctionnement nous avons utilisé l'automate stochastique hybride implémenté sous Scicos – Scilab et effectué une simulation de Monte Carlo. La simulation est arrêtée si les deux conditions données par les équations (2.15) et (2.16) sont vérifiées.

3.2.1.4.3 Fiabilité du système

Pour l'étude de la fiabilité du système, nous nous sommes contentés d'évaluer le MTTF (mean time to failure). Nous avons effectué une simulation de Monte Carlo en rendant absorbant l'état 7 de l'automate stochastique hybride de la figure 3.1. La fiabilité du système est la probabilité qu'il soit dans les états 1, 2, 3, 4, 5 ou 6. Nous avons donc approché le MTTF par la moyenne du temps d'accès à l'état 7 absorbant (MoyTAEA) sur l'ensemble des histoires simulées (une histoire est le passage du système depuis l'état initial par une suite d'états de bon fonctionnement avant d'arriver à l'état de défaillance du système, l'état 7). La simulation est arrêtée si (2.15) et (2.16) sont vérifiées pour les valeurs de $\varepsilon = 0.1$ et de $\theta = 0.9$ pour la mesure considérée MoyTAEA qui approche de manière asymptotique le MTTF. On obtient ainsi :

$$MTTF = 2{,}191 \cdot 10^{6}\ h$$

Les résultats montrent qu'il n'est pas nécessaire de faire plus de 110 histoires pour satisfaire les critères. La durée de simulation étant de l'ordre de 7 minutes. La figure 3.5 présente ces résultats.

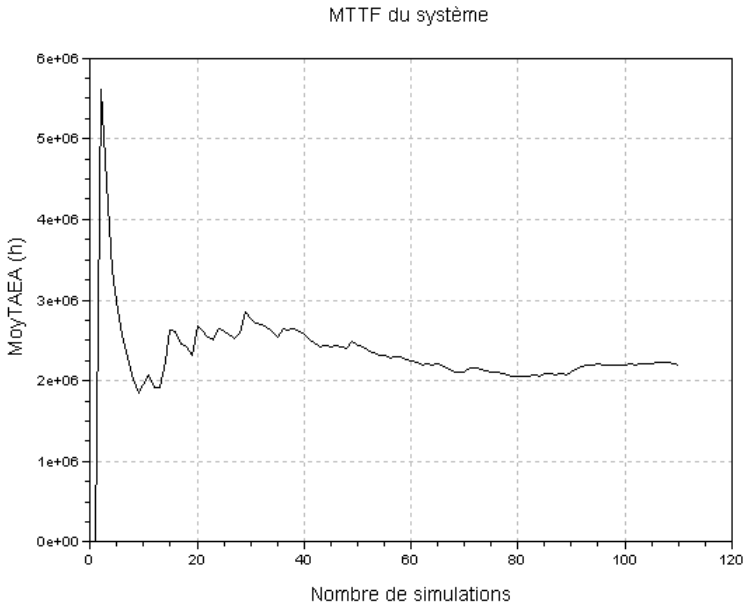


Figure 3.5 – Temps moyen d'accès à l'état de défaillance.

3.2.1.4.4 Disponibilité du système

Comme nous avons mentionné, §1.1.3, la disponibilité du système est la probabilité que le système soit en état d'accomplir une fonction requise dans des conditions données à un instant donné. Nous avons déterminé l'indisponibilité asymptotique $\bar{A}_{\infty} = 1 - A_{\infty}$. L'état 7 est l'état d'indisponibilité du système lorsque ni le contrôleur PI ni le contrôleur TOR ne contrôlent plus la température du four. Pour approcher la disponibilité asymptotique, nous considérons comme mesure le temps moyen de séjour dans l'état d'indisponibilité (TmoySEI) et nous lui appliquons les conditions d'arrêt donné par les équations (2.15) et (2.16) donc avec une précision de $\varepsilon = 0.1$ et une probabilité de convergence de $\theta = 0.9$. Chaque fois que le système passe dans l'état 7, on vérifie si les conditions sont satisfaites. Quand ces conditions sont vérifiées, nous considérons que le régime asymptotique est atteint et nous déterminons l'indisponibilité du système $\bar{A}$ comme le rapport entre le temps de séjour cumulé dans l'état d'indisponibilité (état 7) et le temps de séjour cumulé dans tous les états, y compris l'état d'indisponibilité. Ensuite, nous obtenons la disponibilité du système :

$$A_{\infty} = 1 - \bar{A}_{\infty} = 99{,}99\%$$

Les résultats montrent qu'il n'est pas nécessaire de faire plus de 80 histoires et la durée de simulation étant de l'ordre de 3 minutes. La figure 3.6 présente ces résultats.

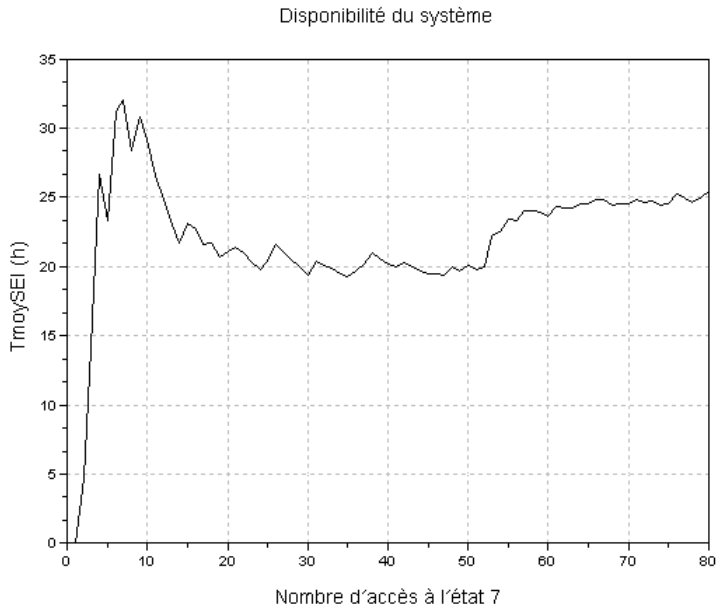


Figure 3.6 – Temps moyen de séjour dans l'état d'indisponibilité.

3.2.1.4.5 Maintenabilité du système

La maintenabilité est l'aptitude d'une entité à être rétablie à l'instant *t* dans un état dans lequel elle peut accomplir une fonction requise, sachant qu'elle est en panne depuis l'instant initial. Nous avons calculé le MTTR (mean time to repair) comme l'espérance mathématique de la durée de réparation. Nous avons donc approché le MTTR par la moyenne du temps d'accès aux états de fonctionnement (MoyTAEF) sur l'ensemble des histoires simulées (une histoire est le passage du système de l'état défaillant 7 vers les états de bon fonctionnement 1 et 4). Nous avons considéré une précision $\varepsilon = 0.1$, équation (2.15), et une probabilité de convergence $\theta = 0.9$, équation (2.16), pour la mesure considérée MoyTAEF qui approche de manière asymptotique le MTTR. On obtient ainsi :

$$MTTR = 28{,}48\ h$$

Les résultats montrent qu'il suffit de simuler 90 histoires. La durée de simulation étant de l'ordre de quatre minutes pour atteindre la précision désirée. La figure 3.7 présente ces résultats.

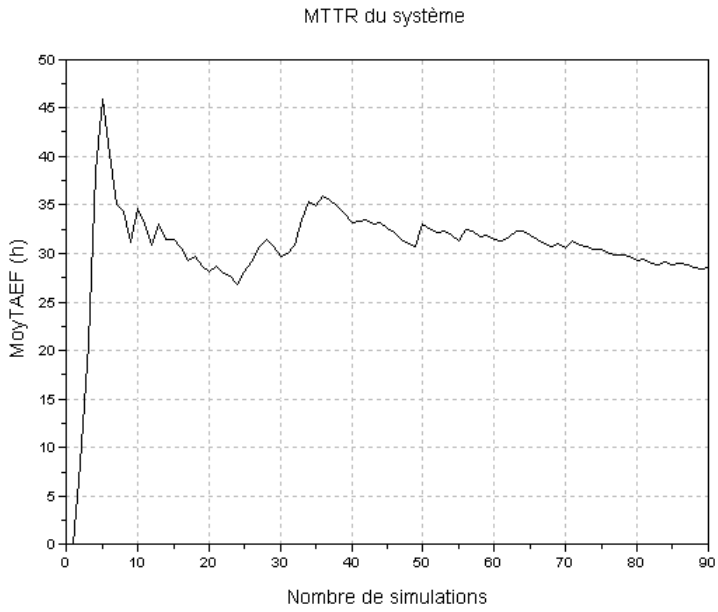


Figure 3.7 – Temps moyen d'accès aux états de fonctionnement.

3.2.1.4.6 Le MUT : temps moyen de disponibilité

Nous avons aussi déterminé l'espérance mathématique de la durée de disponibilité du système. La simulation que nous avons effectué pour déterminer la disponibilité du système nous a servi pour calculer le temps moyen de séjour du système dans les états de marche ou de bon fonctionnement. La simulation effectuée nous a donné :

$$MUT = 2{,}175 \cdot 10^{6}\ \text{h}$$

3.2.1.5 Processus de Markov et processus semi-markovien

Les processus markoviens et semi-markoviens sont souvent utilisés pour l'étude de la sûreté de fonctionnement des systèmes, notamment des systèmes réparables.

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

$$\{Z(t),\ t \in T\} \tag{3.10}$$

En pratique T représente l'espace de temps. Il peut être discret ou continu. Les variables $Z(t)$ prennent leurs valeurs dans un ensemble $\mathcal{X}$ 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 Z des variables aléatoires,

- le domaine T du paramètre *t*,
- des relations statistiques entre les Z(*t*) pour différentes valeurs de *t* définies par :

$$F_z(t)=Pr[Z(t_1)\leq z_1 ;\dots ;Z(t_n)\leq z_n] \qquad \forall z=(z_1,\dots,z_n),\ \forall t=(t_1,\dots,t_n),\ \forall n \tag{3.11}$$

Les processus de Markov sont des processus sans mémoire (la probabilité de transition dépend seulement de l'état courant), c'est-à-dire, qu'à chaque instant le temps restant à passer dans l'état courant est indépendant du temps qu'il y a déjà passé. La seule distribution continue qui vérifie cette hypothèse est la distribution exponentielle. Par ailleurs, les processus semi-markoviens sont des processus suivant des distributions générales [Brinzei, 2003], [Cocozza-Thivent, 1997] et [Niel et Craye, 2002]. L'automate stochastique hybride de la figure 3.1, du système de contrôle de la température du four, ne correspond pas à ces types de processus. Cela est dû au fait que l'automate stochastique hybride modélise un système dynamique lequel a des transitions déterministes qui ne dépendent pas uniquement d'une distribution exponentielle ou du temps écoulé depuis l'arrivée dans un état. Cependant, nous allons approcher l'automate stochastique hybride du système à un processus de Markov et à un processus semi-markovien afin de comparer et donner une estimation à nos résultats numériques obtenus par simulation avec les résultats analytiques apportés par ces approches.

3.2.1.5.1 Processus de Markov

Un processus stochastique est dit markovien si :

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

alors

$$Pr[Z(t_{n+1})=z_{n+1}/Z(t_n)=z_n ; Z(t_{n-1})=z_{n-1} ;\dots ;Z(t_1)=z_1] = Pr\,[Z(t_{n+1})=z_{n+1}/X(t)=z_n] \tag{3.12}$$

Ainsi un processus markovien est un processus sans mémoire. La connaissance de l'état aux instants $t_1<t_2<\dots<t_n<t_{n+1}$ *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.

Le vecteur des probabilités *P* d'être dans un état à l'instant *t* est solution de l'équation de Chapman Kolmogorov :

$$\dot{P}(t) = P(t).A \tag{3.13}$$

où *A* est la matrice de transition entre les états du système :

$$[A] = \begin{bmatrix} -\sum_{j=2}^{n} a_{1j} & a_{12} & . & a_{1i} & . & a_{1n} \\ a_{21} & -\sum_{j=1, j\neq 2}^{n} a_{2j} & . & . & . & a_{2n} \\ . & . & . & . & . & . \\ a_{i1} & . & . & -\sum_{j=1, j\neq i}^{n} a_{ij} & . & a_{in} \\ . & . & . & . & . & . \\ a_{n1} & a_{n2} & . & . & . & -\sum_{j=1}^{n-1} a_{nj} \end{bmatrix} \quad (3.14)$$

et $a_{ij}.dt$ est la probabilité de passer de l'état Z_i à l'état Z_j entre t et $t+dt$ sachant qu'à l'instant t le système est dans l'état Z_i.

Un processus de Markov est représentable graphiquement par un modèle à états-transitions appelé graphe de Markov [Schoenig, 2004]. Nous avons modifié l'automate stochastique hybride du système de contrôle de la température du four (figure 3.1) afin d'obtenir un processus de Markov du système représentée par le graphe de la figure 3.8. Pour cela, nous avons enlevé de l'automate stochastique hybride les transitions déterministes en agrégeant les états discrets but avec les états origine de ces transitions.

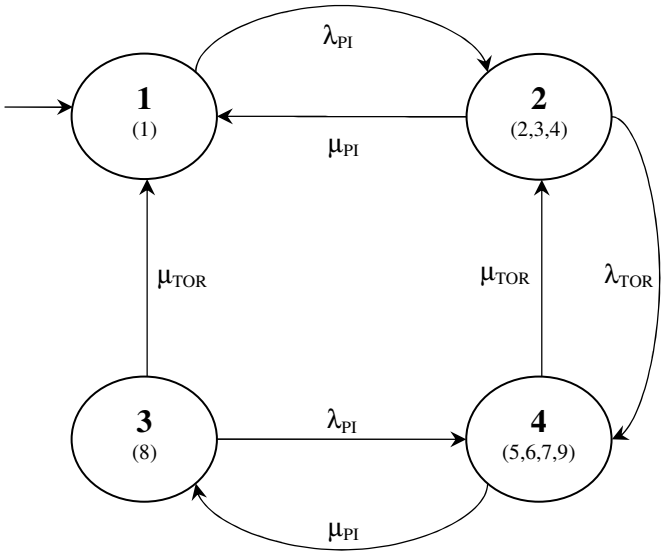


Figure 3.8 – Graphe de Markov équivalente à la matrice de transition *A*.
Les nombres entre parenthèses correspondent aux états agrégés de l'automate stochastique hybride.

Les états 1,2 et 3 sont les états de bon fonctionnement du système :

- état 1 : le contrôleur PI contrôle la température du four. Le contrôleur TOR est inactif,

- état 2 : le contrôleur PI devient défaillant et le contrôleur TOR contrôle maintenant la température du four. La réparation du contrôleur PI ou la défaillance du contrôleur TOR peuvent arriver,

- état 3 : le contrôleur PI est réparé et il est maintenant actif, mais le contrôleur TOR est encore défaillant,

- état 4 : les contrôleurs PI et TOR sont défaillances et ils sont en processus de réparation. Le système est défaillance.

La matrice des transitions du graphe de Markov de la figure 3.8 est donnée par l'équation suivante :

$$A = \begin{bmatrix} -\lambda_{PI} & \lambda_{PI} & 0 & 0 \\ \mu_{PI} & -(\mu_{PI} + \lambda_{TOR}) & 0 & \lambda_{TOR} \\ \mu_{TOR} & 0 & -(\mu_{TOR} + \lambda_{PI}) & \lambda_{PI} \\ 0 & \mu_{TOR} & \mu_{PI} & -(\mu_{PI} + \mu_{TOR}) \end{bmatrix} \tag{3.15}$$

Nous sommes intéressés à déterminer le MTTF, la disponibilité et le MTTR du système exprimé par le graphe de Markov de la figure 3.8.

En intégrant la fiabilité *R(t)* du système nous pouvons déduire le MTTF [Niel et Craye, 2002]. Cependant, il y a une façon plus pratique d'obtenir le même paramètre [Corazza, 1975], Osaki, 2002]. La matrice des transitions (3.15) est partitionnée en fonction des états de bon fonctionnement et des états de défaillance. C'est-à-dire, que les trois premières colonnes correspondent aux états de bon fonctionnement 1, 2 et 3 du graphe de Markov de la figure (3.8). La dernière colonne correspond à l'état de défaillance du système. La sous-matrice G_{11} (3.16) représente les états de bon fonctionnement, tandis que la matrice G_{22} (3.16) correspond à l'état de défaillance.

$$G_{11} = \begin{bmatrix} -\lambda_{PI} & \lambda_{PI} & 0 \\ \mu_{PI} & -(\mu_{PI} + \lambda_{TOR}) & 0 \\ \mu_{TOR} & 0 & -(\mu_{TOR} + \lambda_{PI}) \end{bmatrix} \tag{3.16}$$

$$G_{22} = \left[-(\mu_{PI} + \mu_{TOR}) \right] \tag{3.17}$$

Donc, le MTTF est déterminé par l'équation (3.18) :

$$\mathrm{MTTF} = \mathrm{P_F}(0).(-\mathrm{G_{11}})^{-1}.1_{\mathrm{nf}} \tag{3.18}$$

où P_F est le vecteur de probabilités des états de bon fonctionnent, G_{11} est la sous-matrice des taux de transitions entre états de bon fonctionnement. 1_{nf} est un vecteur de sommation sur tous les états de fonctionnement (3.19), dans notre cas nf=3.

$$1_{\mathrm{nf}}^{\mathrm{T}} = [\underbrace{1,1,...,1}_{\mathrm{nf}}] \tag{3.19}$$

Le MTTR est déterminé par l'équation (3.20) :

$$\mathrm{MTTR} = \mathrm{P_D}(0).(-\mathrm{G_{22}})^{-1}.1_{\mathrm{nd}} \tag{3.20}$$

avec P_D le vecteur de probabilités de l'état de défaillance. G_{22} est la sous-matrice des taux de transitions entre états de défaillance et 1_{nd} correspondant à l'état de défaillance.

Les valeurs des taux des défaillances et des réparations des contrôleurs PI et TOR sont les suivantes :

- $\lambda_{PI} = 13\cdot10^{-05}\ \mathrm{h}^{-1}$; $\lambda_{TOR} = 8\cdot10^{-05}\ \mathrm{h}^{-1}$; $\mu_{PI} = 21\cdot10^{-03}\ \mathrm{h}^{-1}$; $\mu_{TOR} = 14\cdot10^{-03}\ \mathrm{h}^{-1}$

Appliquant les équations (3.18) et (3.19) nous avons obtenu :

$$\mathrm{MTTF} = \frac{\lambda_{PI} + \lambda_{TOR} + \mu_{PI}}{\lambda_{PI}\lambda_{TOR}} = 2{,}039\cdot10^{6}\ h$$

$$\mathrm{MTTR} = \frac{1}{\mu_{PI} + \mu_{TOR}} = 28.57\ h$$

Les résultats obtenus montrent que le processus de Markov approché donne une bonne approximation de ceux obtenus par l'automate stochastique hybride.

La disponibilité du système est déterminée par l'équation (3.21). La matrice A_{m}^{-1} est obtenue en remplaçant la dernière colonne de la matrice *A* par des 1. $\prod$ est le vecteur des probabilités asymptotiques d'être dans chacun des états.

$$\Pi = [0,0,0,1].A_{\mathrm{m}}^{-1} \tag{3.21}$$

On obtient les probabilités asymptotiques suivantes :

$$\prod = [0.99382664237564 \quad 0.00613828172876 \quad 0.00002096765749 \quad 0.00001410823811]$$

Donc, la disponibilité est la somme des probabilités pour le système d'être dans un des états de fonctionnement. Ainsi,

Disponibilité asymptotique = 99.99867%

Le MUT du système est déterminé par l'équation (3.22)

$$MUT = \frac{\Pi_F.G_{11}.G_{11}^{-1}.1_{nf}}{\Pi_F.G_{12}.1_{nd}} = \frac{\Pi_F.1_{nf}}{\Pi_F.G_{12}.1_{nd}} = \frac{\Pi_F.1_{nf}}{\Pi_D.G_{21}.1_{nf}} \quad (3.22)$$

avec : $\Pi = [\Pi_F, \Pi_D]$; $G_{12}=[0 ;\lambda_{TOR} ; \lambda_{PI}]$; $1_{nf} = [1 ;1 ;1]$ et $1_{nd} = [1]$.

On trouve finalement :

$$MUT = \frac{\Pi_F 1_{nf}}{\Pi_F G_{12} 1_{nd}} = 2{,}025 \cdot 10^6 \text{ h}$$

3.2.1.5.2 Processus semi-markovien

Certains systèmes non markoviens sont tels qu'il existe des instants pour lesquels l'état du système résume toute son évolution passée. Ces instants sont appelés points de régénération. Si nous prenons comme ensemble d'indices du processus modélisant le système l'ensemble de ces instants de régénération, nous définissons un processus de Markov discret appelée chaîne immergée. L'étude de ce processus permet de se faire une idée du comportement du système. Un ensemble important en pratique de processus possédant cette propriété est constitué par les processus semi-markoviens. Pour de tels processus, les probabilités de transition d'un état vers un autre ne dépendent que d'une seule variable : le temps écoulé depuis l'arrivée dans cet état. Les processus semi-markoviens sont une généralisation des processus markoviens dans lesquels les temps de passage d'un état à l'autre sont des variables aléatoires suivant des distributions générales. Les processus semi-markoviens ont été introduits simultanément par Levy [Levy, 1954] et Smith [Smith, 1955].

Nous allons aussi déterminer le MTTF, la disponibilité, le MTTR et le MUT du cas test sous le contexte semi-markovien.

Pour déterminer le MTTF du système nous avons réordonné les états discrets de l'automate stochastique hybride de la figure 3.1 en distinguant les états discrets de marche et les états discrets de panne. La figure 3.9 montre les états discrets de marche 1 à 8 et l'état discret 9 de panne du système.

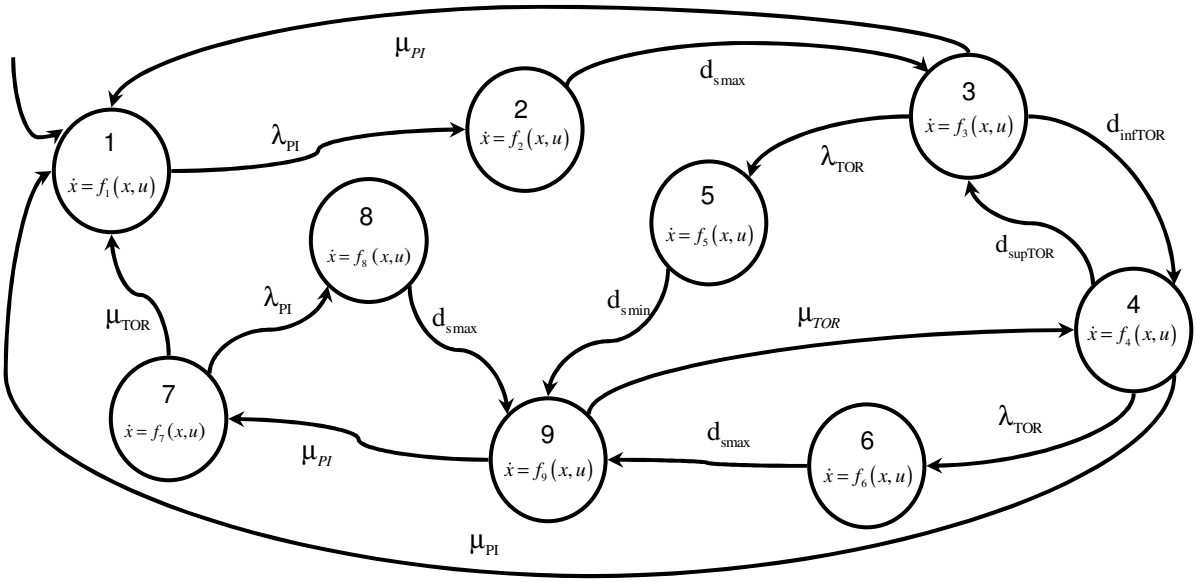


Figure 3.9 – Modèle semi-markovien déduit de l'automate stochastique hybride du système de contrôle de la température d'un four (états réordonnés).

Dans le modèle de la figure 3.9 le temps avant le franchissement d_{ij} (où i est l'état de départ et j est l'état d'arrivée) des certaines transitions (2→3, 3→4, 4→3, 5→9, 6→9 et 8→9) ne sont pas stochastiques mais sont déterminés à partir de la trajectoire de la température par rapport aux seuils prédéfinis. Pour les autres transitions les temps sont distribués exponentiellement. Les instants de franchissement des transitions non stochastiques sont des points de renouvellement du modèle semi-markovien à construire.

Dans le cas d'un processus semi-markovien, le MTTF est le même que celui du processus markovien de sauts qui a même chaîne de Markov immergée, même durée moyenne de séjour dans les états discrets et mêmes états discrets de marche que lui [Cocozza-Thivent, 1997]. Donc, l'équation (3.23) permet calculer le MTTF du système.

$$MTTF = -\mathrm{P}(0)^T \, \mathrm{G}_{11}^{-1} 1_m \tag{3.23}$$

où $\mathrm{P}(0) = [1, 0, 0, 0, 0, 0, 0, 0, 0]$ correspond à la loi initiale du processus, G_{11} à la matrice génératrice des états discrets de marche et 1_m un vecteur colonne de 1 de dimension m = 8 (8 états de marche).

On trouve donc :

$$\mathrm{MTTF} = 2{,}279 \cdot 10^6 \text{ h.}$$

Pour déterminer la disponibilité asymptotique du système nous avons repris l'automate stochastique hybride de la figure 3.1. Nous avons déterminé les probabilités d'états π_i lesquelles sont obtenues par l'équation :

$$\pi_i = \frac{\pi_{im_i}\eta_i}{\sum_{j=1}^{9}\pi_{im_j}\eta_j} \tag{3.24}$$

où π_{im_i} sont les probabilités d'états de la chaîne de Markov immergée et η_i est la durée moyenne de séjour dans l'état x_i.

Les probabilités d'états de la chaîne de Markov immergée sont déterminées par les équations suivantes :

$$\left[\pi_{im_i}\right] = \pi_{im_i} \cdot P \quad \text{et} \quad \sum_{i=1}^{9}\pi_{im_i} = 1 \tag{3.25}$$

P_{ij} correspond à la matrice des probabilités de transition de la chaîne de Markov immergée. Elle est obtenue à partir de la matrice $Q_{ij}(t)$ appelée noyau. $Q_{ij}(t)$ représente la probabilité de passer en une transition de l'état x_i à l'état x_j dans l'intervalle de temps [0,t].

Donc :

$$P_{ij} = Q_{ij}(\infty) \tag{3.26}$$

$$Q_{ij}(t) = \int_0^{\infty}\prod_{k\neq j}(1 - F_{ik}(u))dF_{ij}(u) \tag{3.27}$$

où $F_{ij}(t)$ est la fonction de répartition du temps de passage de l'état x_i vers l'état x_j.

η_i est donnée par la relation suivante :

$$\eta_i = \int_0^{\infty} td\left(\sum_j Q_{ij}(t)\right) = \sum_j P_{ij}\int_0^{\infty} tdF_{ij}(t) \tag{3.28}$$

Les matrices P et $Q(t)$ obtenues an appliquant les équations (3.26) et (3.27) du système représenté par la figure 3.1 sont :

$$Q_{ij}(t)=\begin{bmatrix}
0 & 1-e^{-\lambda_{PI}t} & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & \begin{cases}0\,;\,t<d_{23}\\1\,;\,t\geq d_{23}\end{cases} & 0 & 0 & 0 & 0 & 0 & 0\\
\begin{cases}\dfrac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)t}\right);t<d_{34}\\ \dfrac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d}\right);t\geq d_{34}\end{cases} & 0 & 0 & \begin{cases}0 & ;\,t<d_{34}\\ e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d} & ;\,t\geq d_{34}\end{cases} & \begin{cases}\dfrac{\lambda_{TOR}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-(\mu_{PI}+\lambda_{TOR})t}\right);t<d_{34}\\ \dfrac{\lambda_{TOR}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d_{34}}\right);t\geq d_{34}\end{cases} & 0 & 0 & 0\\
\begin{cases}\dfrac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)t}\right);t<d_{43}\\ \dfrac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d}\right);t\geq d_{43}\end{cases} & 0 & \begin{cases}0 & ;\,t<d_{43}\\ e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d} & ;\,t\geq d_{43}\end{cases} & 0 & 0 & \begin{cases}\dfrac{\lambda_{TOR}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)t}\right);t<d_{43}\\ \dfrac{\lambda_{TOR}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-\left(\mu_{PI}+\lambda_{TOR}\right)d_{43}}\right);t\geq d_{43}\end{cases} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & \begin{cases}0\,;\,t<d_{57}\\1\,;\,t\geq d_{57}\end{cases} & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & \begin{cases}0\,;\,t<d_{67}\\1\,;\,t\geq d_{67}\end{cases} & 0 & 0\\
0 & 0 & 0 & \dfrac{\mu_{TOR}}{\mu_{PI}+\mu_{TOR}}\left(1-e^{-\left(\mu_{PI}+\mu_{TOR}\right)t}\right) & 0 & 0 & 0 & Q_{78} & 0\\
\dfrac{\mu_{TOR}}{\lambda_{PI}+\mu_{TOR}}\left(1-e^{-\left(\lambda_{PI}+\mu_{TOR}\right)t}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Q_{89}\\
0 & 0 & 0 & 0 & 0 & 0 & \begin{cases}0\,;\,t<d_{97}\\1\,;\,t\geq d_{97}\end{cases} & 0 & 0
\end{bmatrix}$$

$$Q_{78}=\frac{\mu_{PI}}{\mu_{PI}+\mu_{TOR}}\left(1-e^{-\left(\mu_{PI}+\mu_{TOR}\right)t}\right)$$

$$Q_{89}=\frac{\lambda_{PI}}{\lambda_{PI}+\mu_{TOR}}\left(1-e^{-\left(\lambda_{PI}+\mu_{TOR}\right)t}\right)$$

$$P_{ij}(t)=\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-(\lambda_{TOR}+\mu_{PI})d_{34}}\right) & 0 & 0 & e^{-(\mu_{PI}+\lambda_{TOR})d_{34}} & \frac{\lambda_{TOR}}{\lambda_{TOR}+\mu_{PI}}\left(1-e^{-(\lambda_{TOR}+\mu_{PI})d_{34}}\right) & 0 & 0 & 0 & 0 \\
\frac{\mu_{PI}}{\mu_{PI}+\lambda_{TOR}}\left(1-e^{-(\lambda_{TOR}+\mu_{PI})d_{43}}\right) & 0 & e^{-(\mu_{PI}+\lambda_{TOR})d_{43}} & 0 & 0 & \frac{\lambda_{TOR}}{\lambda_{TOR}+\mu_{PI}}\left(1-e^{-(\lambda_{TOR}+\mu_{PI})d_{43}}\right) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\frac{\mu_{TOR}}{\mu_{TOR}+\lambda_{PI}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\lambda_{PI}}{\lambda_{PI}+\mu_{TOR}} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}$$

Nous avons obtenu :

Etat	π_{im_i}	η_i (h)	π_i
1	0.3320579	7692.30	0.8892100
2	0.3320579	909.20	0.1051012
3	0.3320579	47.43	0.0054825
4	0.0005223	47.43	0.0000086
5	0.0012618	359.78	0.0001580
6	5.30E-08	608.19	1,122E-08
7	0.0012708	34.59	0.0000153
8	0.0007625	75.19	0.0000200
9	0.0000070	1400.96	0.0000034

La disponibilité asymptotique du système est la somme des probabilités d'états π_i des états discrets de marche. Ainsi,

Disponibilité asymptotique = 99.99847%

Le MTTR est déterminé à partir du graphe de Markov de la figure (3.10), obtenue à partir de la figure 3.1 :

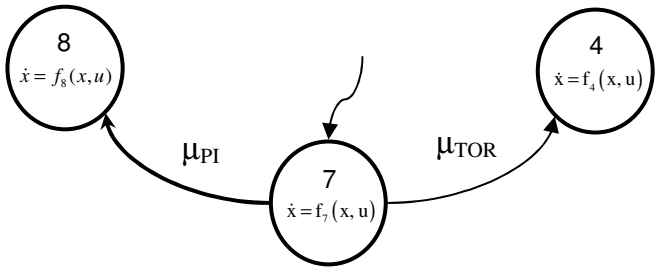


Figure 3.10 – Graphe de Markov pour déterminer le MTTR.

Nous avons appliqué l'équation (3.28) et nous avons obtenu :

$$\text{MTTR} = \frac{1}{\mu_{PI} + \mu_{TOR}} = 28{,}57 \text{ h}$$

Pour déterminer le MUT nous avons repris la figure 3.9. L'équation (3.29) [Cocozza-Thivent, 1997] nous permet obtenir son valeur :

$$MUT = \frac{\pi_1 1_m}{\pi_1 A_{12} 1_m} \tag{3.29}$$

où π_1 correspond aux probabilités d'états des états de marche et A_{12} à la matrice génératrice de transition des états de marche vers l'état de panne.

On a donc obtenu :

$$\text{MUT} = 2{,}264 \cdot 10^6 \text{ h}$$

Le tableur 3.2 montre une récapitulatif des résultats par les trois méthodes utilisées : simulation, chaîne de Markov et processus semi-markovien.

Paramètre	**ASH** ($\varepsilon = 0{,}1$; $\theta = 0{,}9$)	**Processus de Markov**	**Processus semi-markovien**
A	99,99998%	99,99867%	99,99847%
MTTF	$2{,}191 \cdot 10^6$ h	$2{,}039 \cdot 10^6$ h	$2{,}279 \cdot 10^6$ h
MTTR	28,48 h	28,57 h	28,57 h
MUT	$2{,}175 \cdot 10^6$ h	$2{,}025 \cdot 10^6$ h	$2{,}264 \cdot 10^6$ h

Tableau 3.2 – Récapitulatif des résultats par simulation, par chaîne de Markov et par le processus semi-markovien.

3.2.1.6 Conclusion

La modélisation et la simulation du système dynamique hybride simple, avec les taux de défaillances et de réparations constants, à travers l'automate stochastique hybride nous ont permis d'évaluer les grandeurs de la sûreté de fonctionnement. L'intérêt du formalisme est bien entendu vis-à-vis de l'impossibilité à trouver une solution analytique sans faire d'hypothèses réductrices. Nous avons pu prendre en compte les interactions entre fonctionnement et dysfonctionnement pour une évaluation fine de ces paramètres de sûreté de fonctionnement. Nous pouvons remarquer que le modèle Scicos du système est simple et l'approche est systématique et réutilisable. Par ailleurs, nous avons pu visualiser les

changements d'état de l'automate au cours de la simulation. Finalement, nous avons pu constater la capacité de l'automate stochastique hybride à piloter la simulation malgré les comportements déterministe et stochastique. Les temps de simulations sont raisonnables, ce qui permet de prendre en compte de nombreux systèmes technologiques. Les résultats obtenus étant relativement voisins de ceux obtenus par simplification des hypothèses (processus markovien et semi markovien), on pourrait se poser la question de l'intérêt de l'approche. Elle réside bien sûr dans sa capacité à traiter des aspects plus complexes de la fiabilité dynamique comme on le verra dans la suite.

3.2.2 Modes de vieillissement multiples des composants selon l'état du système et différentes lois de probabilité des instants de défaillance.

Dans cette partie nous présentons l'évaluation de la fiabilité du système dynamique dans lequel existent des reconfigurations impliquant des modes de vieillissement différents pour certains composants (changement de lois, changement des paramètres des lois, etc.). Les indices de performance du système, tels que la disponibilité, la fiabilité et la maintenabilité, sont déterminés par simulation de Monte Carlo.

3.2.2.1 Vieillissement des composants

Le modèle couramment adopté pour l'évolution des composants au cours de leur cycle de vie suit trois phases. La « courbe en baignoire » de la figure 3.9 représente ces phases. La partie inférieure de la courbe montre d'une façon plus réaliste chaque distribution de la défaillance d'un composant. Dans [Smith, 2001] on peut trouver une explication plus précise de chaque distribution.

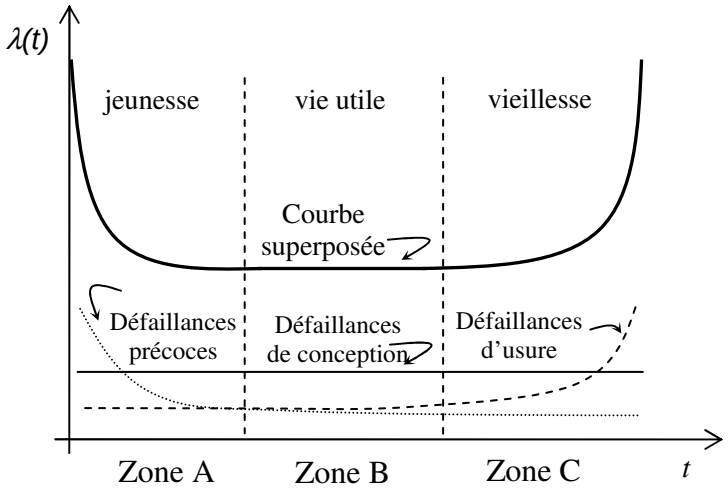


Figure 3.11 – Comportement typique du taux de défaillance d'un composant [Smith, 2001].

Les trois périodes de la courbe en baignoire sont :

- une période dite de "jeunesse" ou encore de "mortalité infantile" (des composants) qui se caractérise par un taux de panne relativement important, mais en décroissance, correspondant à l'élimination des défauts de jeunesse et au rodage (Zone A),

- une période de "vie utile" qui se caractérise par un taux de panne faible et constant. Les différents composants ont prouvés leur robustesse aux défauts de jeunesse, l'équipement est dans sa phase de maturité (Zone B),

- une période de vieillissement et/ou d'usure dans laquelle le taux de panne augmente rapidement en fonction du temps (Zone C).

La plupart des techniques utilisées en sûreté de fonctionnement sont valables pour la zone de vie utile. Une loi qui permet de représenter les trois phases de la durée de vie d'un matériel est la loi de Weibull. Elle est souvent utilisée dans le domaine de l'analyse de la durée de vie, grâce à sa flexibilité. Elle peut reproduire le comportement d'autres lois de probabilité, telle que la loi exponentielle.

Elle a trois paramètres : γ est le paramètre de localisation et il correspond à une simple translation sur l'échelle des temps ; η est le paramètre d'échelle lié au temps de bon fonctionnement et β est le paramètre de forme. Celui-ci est le paramètre le plus important de la loi de Weibull, car il intervient directement sur la variation du taux de défaillance et permet ainsi de modéliser alternativement les trois phases de la courbe en baignoire. Une valeur de $\beta<1$ correspond à la période de jeunesse, pour $\beta=1$ on retrouve la loi exponentielle, c'est la période de maturité, et $\beta>1$ correspond à la période de vieillisse ou d'usure des matériels.

Donc la fonction de répartition correspondant à cette loi est donnée par :

$$F(t) = 1 - \exp\left[-\left(\frac{t-\gamma}{\eta} \right)^{\beta} \right] \tag{3.30}$$

Le taux de défaillance est :

$$\lambda(t) = \frac{\beta (t-\gamma)^{\beta-1}}{\eta^{\beta}} \tag{3.31}$$

Nous nous sommes intéressés dans la suite à un moyen d'implémenter cette loi pour modéliser et simuler le vieillissement des composants, particulièrement de la partie de la commande du système dynamique hybride.

3.2.2.2 Les approches

Dans la littérature on retrouve différentes façons d'aborder le phénomène du vieillissement des composants. Il y a des approches qui se basent sur l'expérience et la connaissance du phénomène. Grellier, Oddi et Simon [Grellier *et al.,* 2006] présentent des actions pour maîtriser le vieillissement et l'obsolescence des matériels électroniques. Cela implique d'identifier et de comprendre les mécanismes de vieillissement des matériels, de mesurer les indicateurs du vieillissement, d'interpréter ces indicateurs et d'agir pour maîtriser le risque du vieillissement. D'autres approches qui utilisent la loi de Weibull afin de donner une estimation du vieillissement d'un ou plusieurs composants. Dans [Cipoire *et al.,* 2004] l'objectif est la construction d'un modèle qui utilise deux informations (distance et temps) transformées dans une seule variable permettant de modéliser au mieux la sollicitation subie par un composant. La loi de Weibull permet d'étudier la fiabilité de ce composant ou d'un groupe de composants pour obtenir une meilleure corrélation entre les incidents observés et la modélisation de la loi de défaillance. L'approche proposée par [Clarotti *et al.,* 2004] permet de distinguer entre une défaillance due à la mauvaise conception d'une pièce d'un équipement et une défaillance due au phénomène de vieillissement. Elle utilise le logiciel développé pour le traitement du vieillissement IBTV (Inférence Bayésienne pour le Traitement du Vieillissement). L'indicateur utilisé pour surveiller le comportement d'un composant est le taux de défaillance. Ainsi le vieillissement d'un composant est représenté par l'accumulation de ces défaillances.

Par ailleurs, comme nous avons déjà présenté dans §1.6.2.5, Soro, Nourelfath et Aït-Kadi [Soro *et al.,* 2006] proposent une stratégie d'amélioration des performances d'un système multi-états. La méthode considère que l'état du système se dégrade avec l'usage. Bien entendu, ces dégradations peuvent conduire à la diminution de l'efficacité du système. Les niveaux de dégradation ne peuvent pas excéder un certain seuil limite qui peut être considéré comme une défaillance complète. Le système peut défaillir aléatoirement à n'importe quel état opérationnel et faire l'objet d'une réparation minimale. Cette action de réparation ramène le système à son état opérationnel précédent sans affecter son taux de défaillance. L'approche markovienne est adoptée pour représenter le système et la modélisation analytique est donnée par l'établissement des équations de Chapman-Kolmogorov. Les transitions d'un état vers un autre s'effectuent selon la loi exponentielle, voir figure 1.11.

Pour évaluer les stratégies de maintenance d'un système sous des contraintes économiques Borgonovo, Marseguerra et Zio [Borgonovo *et al.,* 2000] proposent de prendre en compte la durée de vie passée d'un composant ainsi que la prévision de sa durée de vie restante. Dans le cas général, l'influence du vieillissement sur la fiabilité et la disponibilité d'un composant peuvent être illustrées par l'évolution du taux de défaillance dans le temps. Afin de caractériser le vieillissement, il y a deux paramètres importants : le moment où débute le vieillissement et la vitesse d'augmentation du taux de défaillance.

3.2.2.3 Approche proposée

Nous voulons évaluer, par une simulation de Monte Carlo, la fiabilité du même système dynamique du four contrôlé en température utilisé précédemment. On considère maintenant que les régulateurs ne seront remis à l'état neuf qu'après leur mort survenant après un certain nombre d'alternances de défaillances / réparations. A chaque réparation du PI ou du TOR, celui-ci se dégrade, c'est-à-dire qu'après chaque réparation il risque de tomber en panne plus vite que précédemment (diminution de l'espérance du temps de bon fonctionnement t_{BF}). Nous proposons alors de considérer que si le t_{BF} atteint une valeur trop faible le composant sera alors remplacé (remis à neuf). Nous avons donc modélisé le vieillissement des contrôleurs PI et TOR et la défaillance du four. Pour les contrôleurs nous avons utilisé la loi de Weibull. Le paramètre d'échelle de temps η (appelé aussi durée de vie caractéristique) a permis de définir t_{BF} et la dégradation des composants jusqu'à leur mort.

Le four quant à lui présente deux modes de vieillissement selon le type de commande : quand sa température est contrôlée par le PI le four a un comportement continu, sa défaillance suit une loi en fonction du temps. Par contre, quand sa température est contrôlée par le TOR il fonctionne de façon discontinue : il est allumé ou il est éteint en fonction des seuils imposés par le TOR. Sa probabilité de défaillance dépend alors du nombre de commutations. Dans notre modèle de simulation, nous avons utilisé la loi normale en fonction du temps pour déterminer le t_{BF} total du four quand il est couplé au PI et la loi uniforme en fonction du nombre de commutations subies par le four quand il est piloté par le TOR.

Tous ces aspects sont gérés sans difficulté par l'automate stochastique hybride.

Au début de la simulation, on fait les tirages aléatoires pour déterminer : le t_{BF} des contrôleurs (t_{BFPI} et t_{BFTOR}), le temps critique de fonctionnement en mode PI supporté par le four t_{crit} et le nombre critique de commutations provoquées par le fonctionnement en mode TOR supporté pour le four n_{ccrit}. Les paramètres β et γ de la loi de Weibull sont fixes. A chaque défaillance constatée du contrôleur concerné, on diminue le paramètre η comme l'indique l'équation suivante :

$$\eta = \alpha\eta, \quad \text{où} \quad 0<\alpha<1 \tag{3.32}$$

Quand le t_{BF} du contrôleur concerné est inférieur à une valeur prédéfinie t_{minBF} ($t_{minBFPI}$ et $t_{minBFTOR}$), on remplace le contrôleur et on réinitialise la valeur de η.

Pour le four, on compare le temps d'utilisation du four en mode PI t_c avec le t_{crit} et on compare aussi le nombre de commutations n_{cc} subies par le four en mode TOR avec le nombre de commutations autorisées n_{ccrit}. Lorsqu'une de ces conditions, équations (3.33) et (3.34), est vérifiée le four devient défaillant.

$$t_c \geq t_{crit} \tag{3.33}$$

$$n_{cc} \geq n_{ccrit} \tag{3.34}$$

Ces deux fonctions sont sans dimension et représentent un « pourcentage » d'utilisation: lorsque la valeur maxima est atteinte, elles valent 1 et donc la défaillance du four se produit.

Les réparations des contrôleurs PI et TOR ainsi que celle du four sont de caractère exponentiel.

3.2.2.4 Description du comportement du système

Le système du contrôle de la température fonctionne de la manière suivante : au démarrage la température *x* du four est contrôlée par le contrôleur PI. Au bout d'un certain temps aléatoire le contrôleur tombe en panne (avec un taux λ_{PI} déterminé par η_{PI} de la loi de Weibull) et la température du four augmente rapidement. Le système de détection de franchissement de seuils vérifie que la température du four a atteint une valeur dangereuse ($x \geq x_{s\max}$) déduisant que la température du four est hors contrôle. Il donne alors l'ordre au relais de basculer sur la boucle TOR. La boucle du contrôleur PI est maintenant ouverte et la boucle TOR fermée. La température du four est contrôlée maintenant par le TOR ($x_{\inf TOR} \leq x \leq x_{\sup TOR}$). Le processus de réparation du contrôleur PI est enclenché (on considère une réparation avec un taux μ_{PI} et un facteur de dégradation pour le contrôleur dû à sa défaillance). Cependant, la possibilité de défaillance du TOR existe après une durée également aléatoire (un λ_{TOR} déterminé par η_{TOR} de la loi de Weibull). Une fois que le contrôleur PI est réparé, le système de détection bascule le relais sur la boucle de celui-ci et ouvre la boucle du TOR. La température du four est maintenant à nouveau régulée par le contrôleur PI. On inclut également le processus de réparation du TOR μ_{TOR} et sa dégradation s'il devient défaillant. Après la réparation des contrôleurs PI et TOR, ils deviennent opérationnels mais avec une dégradation due aux réparations qu'ils ont subis. Le four vieillit jusqu'à sa défaillance à cause de trop fortes sollicitations : d'un côté, pour son fonctionnement continu quand il est couplé avec le contrôleur PI, et d'un autre côté pour son fonctionnement discontinu quand il est couplé avec le TOR.

3.2.2.4.1 États du système

Nous avons construit l'automate stochastique hybride pour le système ci-avant décrit. Il est présenté par la figure 3.12. Une brève description de celle-ci est donnée:

- Etat 1 : le contrôleur PI est actif et il contrôle la température du four. Le contrôleur TOR est inactif.

- Etat 2 : le contrôleur PI est défaillant mais toujours actif, sa défaillance n'est pas encore détectée.

- Etats 3 et 4 : la défaillance du contrôleur PI est bien détectée. Le contrôleur TOR est maintenant actif. Le processus de réparation du PI est déclenché. La défaillance du TOR peut arriver.

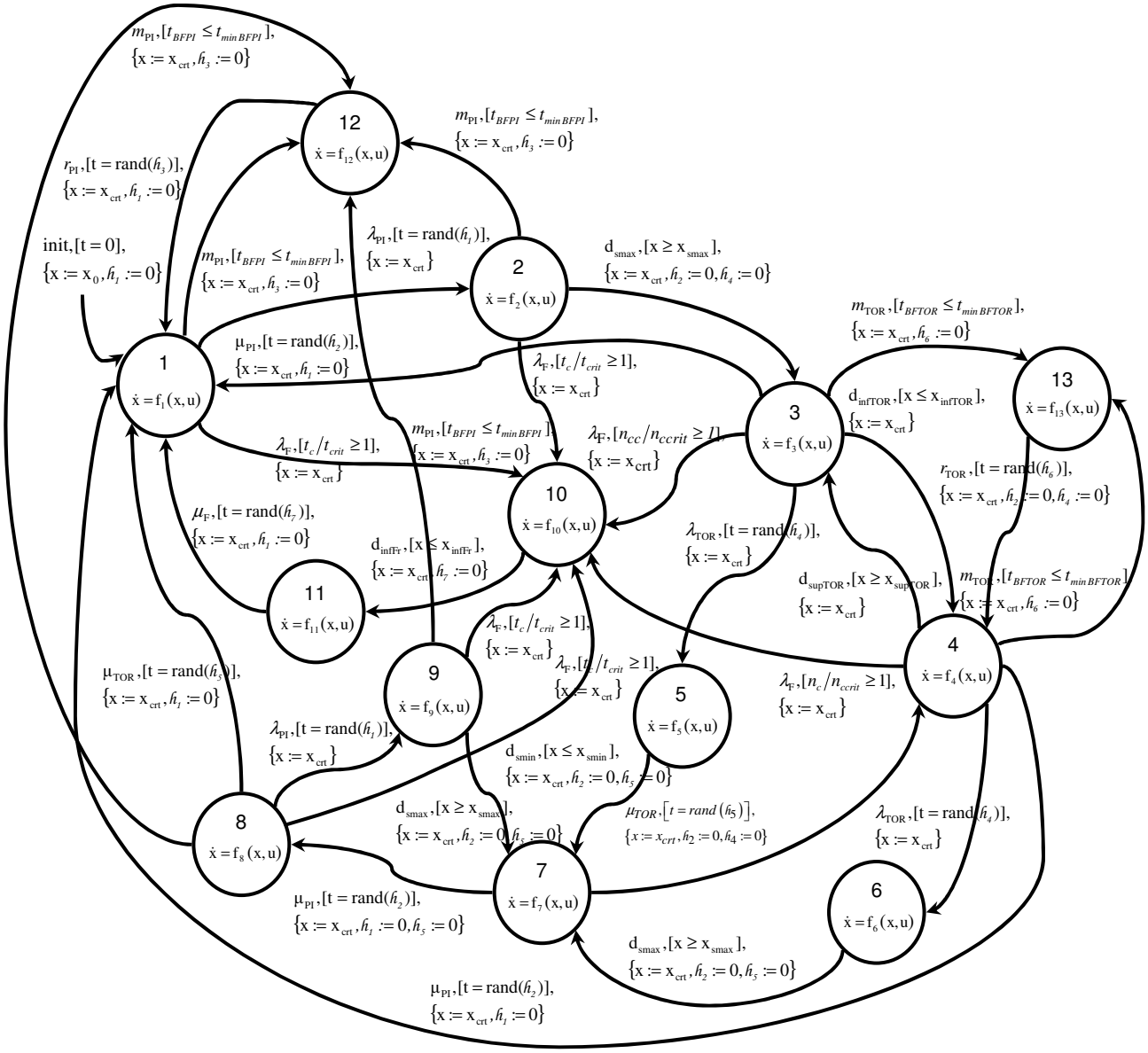


Figure 3.12 – Automate stochastique hybride du four contrôlé en température avec des modes de vieillissement multiples.

- Etats 5 et 6 : le contrôleur TOR est défaillant mais toujours actif, sa défaillance n'est pas encore détectée.

- Etat 7 : la défaillance du contrôleur TOR est bien détectée. Ni le contrôleur PI ni le contrôleur TOR ne fonctionnent plus, ils sont défaillants. Le processus de réparation des contrôleurs PI et TOR est initialisé. Un niveau de dégradation est subi pour ces composants.

- Etat 8 : le contrôleur PI est réparé, il contrôle de nouveau la température du four. Mais sa défaillance peut arriver. On attend la réparation du TOR.

- Etat 9 : la défaillance du contrôleur PI est arrivée, mais il est toujours actif. Sa défaillance n'est pas encore détectée.

- Etat 10 : le four est défaillant, mais toujours actif. Un des contrôleurs est toujours actif.

- Etat 11 : la défaillance du four est détectée. Il est réparé.

- Etat 12 : la mort du contrôleur PI est arrivée. Il est remplacé.

- Etat 13 : la mort du contrôleur TOR est arrivée. Il est remplacé.

3.2.2.4.2 Paramètres de l'automate stochastique hybride correspondant au cas test

En partant de la définition donnée de l'automate stochastique hybride nous avons les expressions suivantes pour le cas test :

$\mathcal{X} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$.

$\mathcal{E} = \{\lambda_{PI}, \lambda_{TOR}, \lambda_F, \mu_{PI}, \mu_{TOR}, \mu_F, d_{smin}, d_{smax}, d_{infTOR}, d_{supTOR}, d_{infFr}, r_{PI}, r_{TOR}, m_{PI}, m_{TOR}\}$

avec :

- λ_{PI}, λ_{TOR}, λ_F, μ_{PI}, μ_{TOR}, μ_F, r_{PI} et r_{TOR} sont respectivement les taux de défaillance des contrôleurs PI et TOR et du four, leurs taux de réparation ainsi que les taux de remplacements (remise à neuf, lois exponentielles) du PI et du TOR. Ces taux correspondent aux transitions stochastiques du système. Ils seront produits par le générateur aléatoire dans la simulation,

- d_{smin} et d_{smax} sont les événements de franchissement des seuils de température minimum et maximum au-delà de laquelle le système de détection de franchissement de seuils identifie la défaillance des contrôleurs PI et TOR,

- d_{infTOR} et d_{supTOR} sont les événements de franchissement des seuils du contrôleur TOR et quand ils ont franchis le four s'allume ou s'éteint,

- d_{infFr} est l'événement de franchissement du seuil de température associé à la défaillance du four,

- m_{PI} et m_{TOR} correspondent respectivement à la mort des contrôleurs PI et TOR. Ils seront remplacés.

G = $\{t=rand(\hbar_1)$; $t=rand(\hbar_2)$; $t=rand(\hbar_3)$; $t=rand\ (\hbar_4)$; $t=rand(\hbar_5)$; $t=rand(\hbar_6)$; $t=rand(\hbar_7)$; $t_{BFPI} \leq t_{minBFPI}$; $t_{BFTOR} \leq t_{minBFTOR}$; $t_c/t_{crit} \geq 1$; $n_{cc}/n_{ccrit} \geq 1$; $x \leq x_{infFr}$; $x \leq x_{smin}$; $x \geq x_{smax}$; $x \leq x_{infTOR}$; $x \geq x_{supTOR}\}$.

$X = \{x\}$, représente la variable physique du système : la température.

$A : \mathcal{X} \times X \rightarrow \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}, f_{12}, f_{13}\}$. Ces fonctions correspondent aux dynamiques du système dans chaque état discret.

$R = \{x = x_{crt}\}$. La valeur de la température x à l'entrée dans chaque état est la même quand le système a quitté l'état précédant (température courante x_{crt}). $R = \{h_1 := 0, h_2 := 0, h_3 := 0, h_4 := 0, h_5 := 0, h_6 := 0, h_7 := 0\}$ représentent les réinitialisations des horloges qui modélisent les temps de défaillance, de réparation et du remplacement des composants.

$\mathcal{H} = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7\}$, où h_1, h_2 et h_3 représentent respectivement le temps de bon fonctionnement, le temps de réparation et le temps de remplacement du contrôleur PI. h_4, h_5 et h_6 représentent de même le temps de bon fonctionnement, le temps de réparation et le temps de remplacement du contrôleur TOR. h_7 correspond au temps de réparation du four.

$F_1(h) = 1 - exp\left[\left(\left(h - \gamma\right)/\eta\right)^{\beta}\right]$ vient de l'équation (3.30) pour modéliser le vieillissement des contrôleurs PI et du TOR.

$F_2(h) = 1 - exp(-\lambda h)$. La loi exponentielle est utilisée pour déterminer les taux de réparation et les taux de remplacement de composants.

p n'est pas utilisée parce qu'il n'y a pas de transitions en conflit.

Le tableau 3.3 définit la fonction de transition de l'automate : pour chaque état, l'état discret futur dépend de l'événement associé à l'arc et de la condition de garde, l'état continu étant réinitialisé par la fonction R associée à cet arc.

Etat origine	***e***	**Condition de garde [G]**	**Réinitialisation {R}**	**Etat but**
1	λ_{PI} λ_F m_{PI}	$[t = rand(h_1)]$ $[t_c / t_{crit} \geq 1]$ $[t_{BFPI} \leq t_{minBFPI}]$	$\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_3 := 0\}$	2 10 12
2	d_{smax} λ_F m_{PI}	$[x \geq x_{smax}]$ $[t_c / t_{crit} \geq 1]$ $[t_{BFPI} \leq t_{minBFPI}]$	$\{x := x_{crt}, h_2 := 0, h_4 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_3 := 0\}$	3 10 12

3	μ_{PI} d_{infTOR} λ_{TOR} λ_F m_{TOR}	$[t = rand(h_2)]$ $[x \le x_{infTOR}]$ $[t = rand(h_4)]$ $[n_{cc} / n_{ccrit} \ge 1]$ $[t_{BFTOR} \le t_{minBFTOR}]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_6 := 0\}$	1 4 5 10 13
4	μ_{PI} d_{supTOR} λ_{TOR} λ_F m_{TOR}	$[t = rand(h_2)]$ $[x \ge x_{supTOR}]$ $[t = rand(h_4)]$ $[n_{cc} / n_{ccrit} \ge 1]$ $[t_{BFTOR} \le t_{minBFTOR}]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_6 := 0\}$	1 3 6 10 13
5	d_{smin}	$[x \le x_{smin}]$	$\{x := x_{crt}, h_2 := 0, h_5 := 0\}$	7
6	d_{smax}	$[x \ge x_{smax}]$	$\{x := x_{crt}, h_2 := 0, h_5 := 0\}$	7
7	μ_{PI} μ_{TOR}	$[t = rand(h_2)]$ $[t = rand(h_5)]$	$\{x := x_{crt}, h_1 := 0, h_5 := 0\}$ $\{x := x_{crt}, h_2 := 0, h_4 := 0\}$	8 4
8	μ_{TOR} λ_{PI} λ_F m_{PI}	$[t = rand(h_5)]$ $[t = rand(h_1)]$ $[t_c / t_{crit} \ge 1]$ $[t_{BFPI} \le t_{minBFPI}]$	$\{x := x_{crt}, h_1 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_3 := 0\}$	1 9 10 12
9	d_{smax} λ_F m_{PI}	$[x \ge x_{smax}]$ $[t_c / t_{crit} \ge 1]$ $[t_{BFPI} \le t_{minBFPI}]$	$\{x := x_{crt}, h_2 := 0, h_5 := 0\}$ $\{x := x_{crt}\}$ $\{x := x_{crt}, h_3 := 0\}$	7 10 12

10	d_{infFr}	$\left[x \le x_{infFr} \right]$	$\{x := x_{crt}, h_7 := 0\}$	11
11	μ_F	$\left[t = rand\left(h_7\right) \right]$	$\{x := x_{crt}, h_1 := 0\}$	1
12	r_{PI}	$\left[t = rand\left(h_3\right) \right]$	$\{x := x_{crt}, h_1 := 0\}$	1
13	r_{TOR}	$\left[t = rand\left(h_6\right) \right]$	$\{x := x_{crt}, h_2 := 0, h_4 := 0\}$	4

Tableau 3.3 – Définition de la fonction de transition de l'automate stochastique hybride.

3.2.2.4.3 Modélisation et simulation du cas test

Du point de vue fiabiliste, le système a maintenant trois composants : le contrôleur PI, le contrôleur TOR et le four. Les paramètres utilisés pour la simulation de Monte Carlo sont :

- vieillissement des contrôleurs PI et TOR :
  - loi de Weibull :

    $\gamma = 0$, $\beta = 2$, $\eta_{iPI} = 52700$ h, $\eta_{iTOR} = 5000$ h ;

- dégradation des contrôleurs après leur réparation :

  $\eta_{PI} = 0.8\,\eta_{PI}$, $\eta_{TOR} = 0.85\,\eta_{TOR}$;

- défaillance du four :
  - pour le temps critique t_{crit} quand le four est couplé au PI :
    - · la loi normale :

      moyenne 287600 h ; écart type = 20000 h ;
  - pour le nombre de commutations critiques n_{ccrit} quand le four est couplé au TOR :
    - · la loi uniforme :

      sur [70,85] ;

- x_{smax} = 240 °C ; x_{smin} = 140 °C ; x_{infTOR} = 170 °C ; x_{supTOR} = 210 °C ; x_{infFr} = 170 °C

- $t_{minBFPI}$ = 5270 h ; $t_{minBFTOR}$ = 500 h ; $\mu_{PI} = 14 \cdot 10^{-06}$ h^{-01} ; $\mu_{TOR} = 10 \cdot 10^{-06}$ h^{-01} ; $\mu_F = 8 \cdot 10^{-06}$ h^{-01} ; $r_{PI} = 14 \cdot 10^{-06}$ h^{-01} ; $r_{TOR} = 10 \cdot 10^{-06}$ h^{-01}

Les équations différentielles associées aux différents états discrets sont :

- État 1 et 8 :

$$\dot{x} + 0{,}0015x - 0{,}0015u_{ref} = 0 \tag{3.35}$$

- État 2, 4, 6 et 9 :

$$1500\dot{x} + x - u_{map} = 0 \tag{3.36}$$

- États 3 et 5 :

$$1500\dot{x} + x - u_{mip} = 0 \tag{3.37}$$

- État 7, 10, 11, 12 et 13 :

$$1500\dot{x} + \exp\left(1/1500\right)x - u_s = 0 \tag{3.38}$$

où :

- u_{ref} = 190°C (température de référence)
- u_{map} = 300°C (température asymptotique à max puissance)
- u_{mip} = 25°C (température asymptotique à min puissance)
- u_s = 25°C (température ambiante)

Les valeurs choisies pour les différents paramètres n'ont qu'une justification pédagogique, notre objectif, comme nous l'avons déjà dit, étant seulement de montrer la faisabilité de l'étude.

3.2.2.4.4 Résultats

3a. Le modèle et la simulation du système dynamique

La figure 3.13 présente le modèle Scicos du système dynamique du contrôle de la température du four dont le comportement a été donné par la figure 3.10. L'automate, le générateur aléatoire et le descripteur de modes sont montrés. Le descripteur de modes a 13 blocs Scicos : un bloc pour chaque état discret. Les états 10 à 13 correspondent à la défaillance du four ou à la « mort » des contrôleurs PI ou TOR. Les blocs qui se trouvent en dessous du descripteur de modes évaluent le vieillissement des composants.

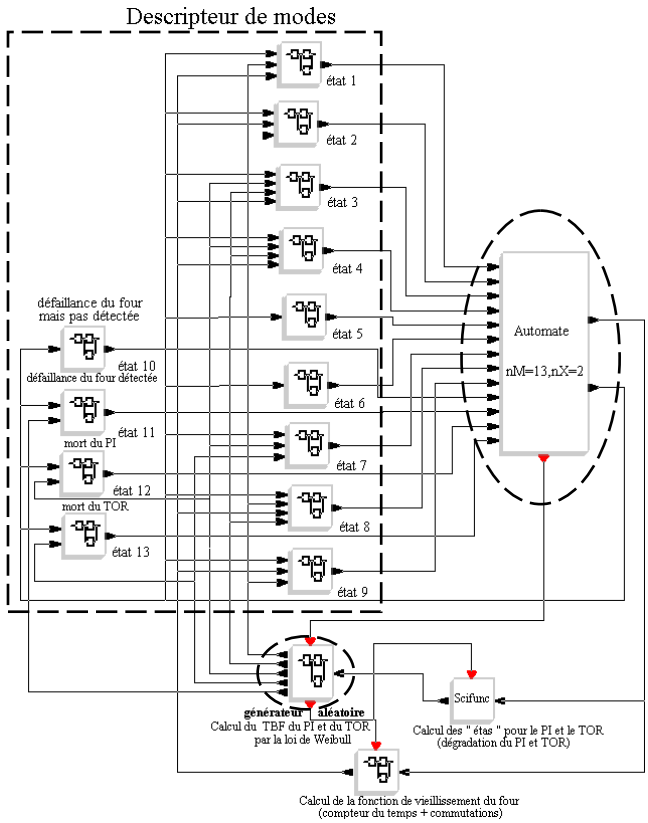


Figure 3.13 – Modèle Scicos du système dynamique hybride.

La figure 3.14 présente la simulation de l'automate stochastique hybride de la figure 3.12. La courbe en haut montre les différents états discrets dans lesquels le système se trouve. La courbe en bas présente le comportement de la température. Cette dernière courbe montre la réponse à l'échelon de la température de référence au démarrage, puis la défaillance de la boucle PI (à t_1) identifiée par le passage du seuil de danger et ensuite la régulation TOR jusqu'à la réparation du contrôleur PI (à t_2). Le contrôleur PI reprend à nouveau le contrôle de la température du four. Le temps de bon fonctionnement du contrôleur PI est maintenant inférieur au précédent dû à la dégradation qu'il a subi. Tout cela se répète jusqu'à sa mort (à t_3). Par ailleurs, la courbe montre aussi la défaillance du four (à t_4) due à ses deux types de modes de fonctionnement : continu par rapport au contrôleur PI et discontinu par rapport au contrôleur TOR. Aux instants t_3 et t_6 la mort du contrôleur PI s'est produite et à l'instant t_4 la défaillance du four.

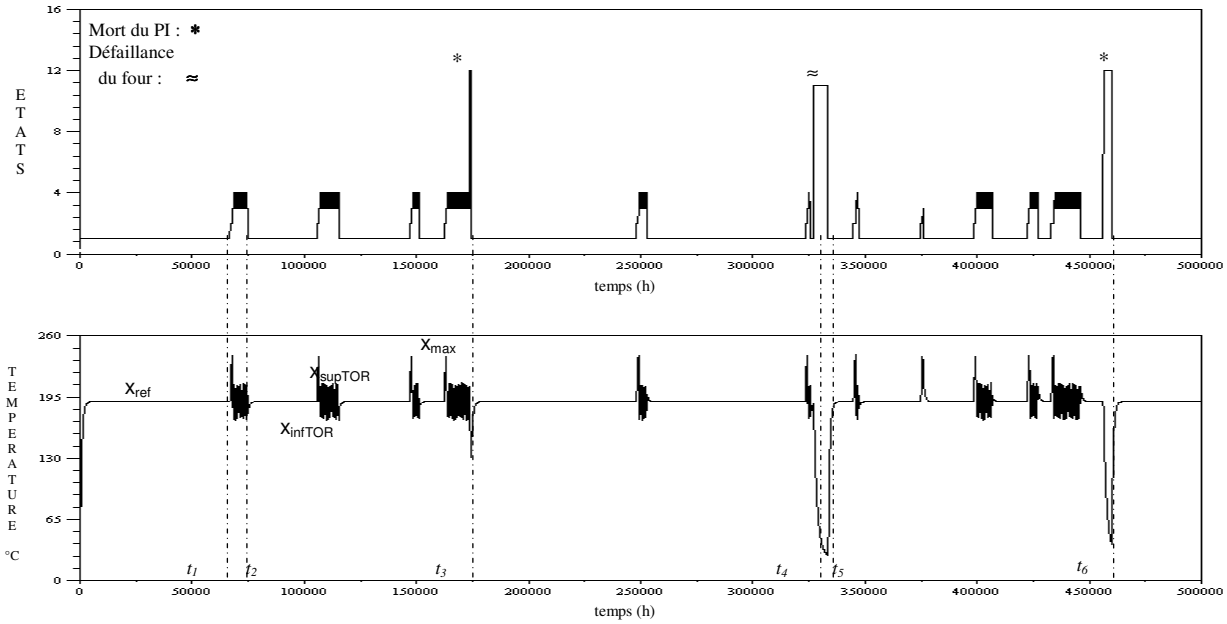


Figure 3.14 – Simulation du système hybride avec l'automate stochastique hybride.

3b. Fiabilité du système

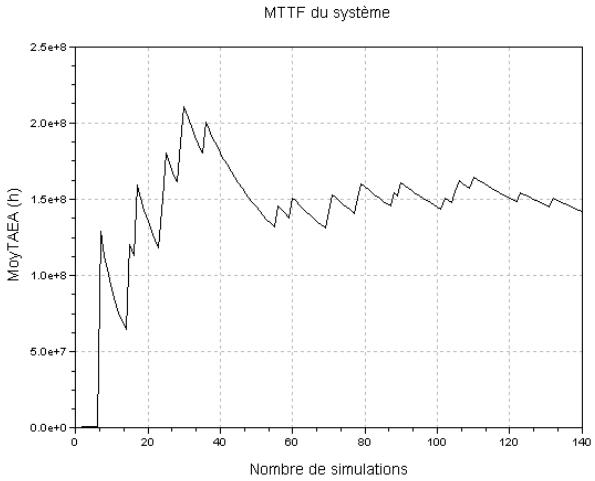
Les grandeurs moyennes de la sûreté de fonctionnement ont été obtenues par une simulation de Monte Carlo effectuée sur le modèle Scicos du système dynamique hybride proposé. Les mêmes conditions, équations (2.15) et (2.16), ont été appliquées à la simulation pour l'arrêter.

Pour l'étude de la fiabilité du système, nous nous sommes contentés d'évaluer le MTTF (mean time to failure). Nous avons effectué une simulation de Monte Carlo en rendant absorbants les états 7 et 11 de l'automate stochastique hybride de la figure 3.12. La fiabilité du système est la probabilité qu'il soit dans les états 1, 2, 3, 4, 5, 6,10, 12 et 13. Nous avons donc approché le MTTF (mean time to failure) par la moyenne du temps d'accès aux états absorbants (MoyTAEA) sur l'ensemble des histoires simulées (une histoire est le passage du système depuis l'état initial par une suite d'états de bon fonctionnement avant d'arriver aux états de défaillance du système (les états 7 et 11). Nous avons considéré une précision $\varepsilon = 0.1$ et une probabilité de convergence $\theta = 0.9$, équations (2.15) et (2.16), pour la mesure considérée MoyTAEA qui approche de manière asymptotique le MTTF. Nous obtenons obtient ainsi :

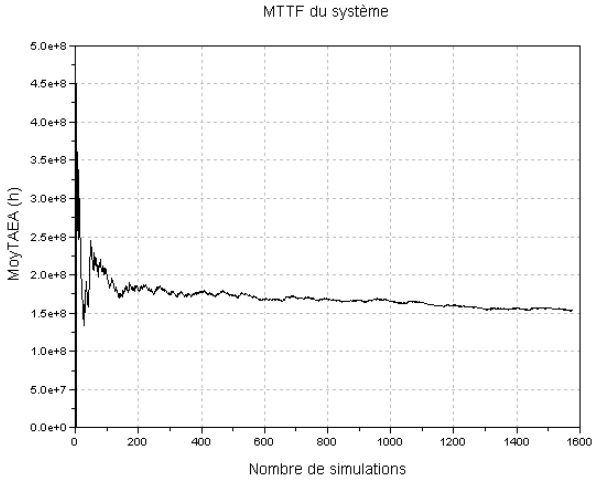
$$MTTF = 1{,}4 \cdot 10^{8}\ h$$

Les résultats montrent qu'il n'est pas nécessaire de faire plus de 140 histoires, la durée de simulation étant de l'ordre de 20 minutes. La figure 3.15a présente ces résultats.

a)



b)



c)

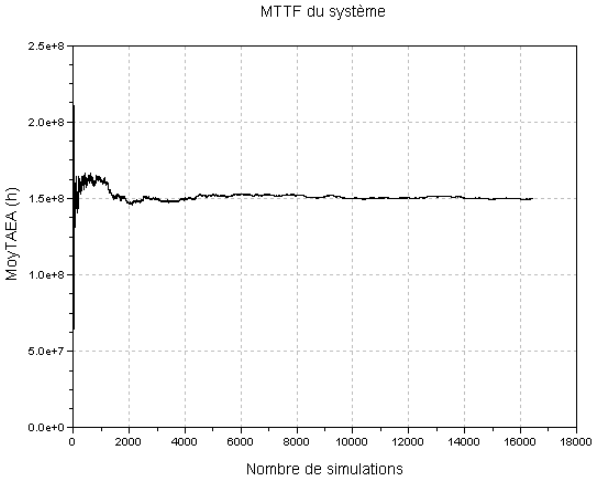


Figure 3.15 – Temps moyen d'accès à l'état de défaillance :
a) epsilon = 0.1, temps calcul = 0.1735min ;
b) epsilon = 0.01, temps calcul= 1.7159h ;
c) epsilon = 0.001, temps calcul = 17.12 h.

3c. Disponibilité du système

La disponibilité du système est la probabilité que le système soit en état d'accomplir une fonction requise dans des conditions données à un instant donné. Nous avons déterminé l'indisponibilité asymptotique $\overline{A}_{\infty} = 1 - A_{\infty}$. Les états 7 et 11 sont les états d'indisponibilité du système lorsque ni le contrôleur PI ni le contrôleur TOR ne contrôlent plus la température du four et le four est défaillant. Pour approcher la disponibilité asymptotique, on considère comme mesure le temps moyen de séjour dans les états d'indisponibilité (TmoySEI) et on lui applique les critères d'arrêt donné par les équations (2.15) et (2.16) avec une précision $\varepsilon = 0.1$ et une probabilité de convergence $\theta =$ 0.9 (chaque fois que le système passe dans les états 7 et 11, on vérifie si ces équations sont satisfaites). Quand ces critères d'arrêt de la simulation sont satisfaits, on considère que le régime asymptotique est atteint et on détermine l'indisponibilité du système comme le rapport entre le temps de séjour cumulé dans les états d'indisponibilité (états 7 et 11) et le temps de séjour cumulé dans tous les états, y compris les états d'indisponibilité. Ensuite, on obtient la disponibilité asymptotique du système :

$$A_{\infty} = 1 - \overline{A}_{\infty} = 99{,}80\ \%$$

Les résultats montrent qu'il n'est pas nécessaire de faire plus de 80 histoires, la durée de simulation étant de l'ordre de 6 minutes. La figure 3.16 présente ces résultats.

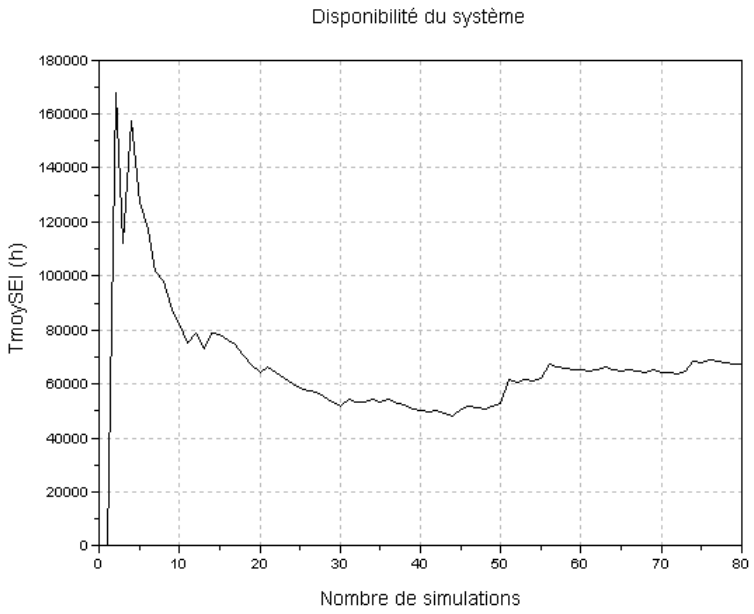


Figure 3.16 – Temps moyen de séjour dans l'état d'indisponibilité.

3d. Maintenabilité du système

La maintenabilité est l'aptitude d'une entité à être rétablie à l'instant *t* dans un état dans lequel elle peut accomplir une fonction requise, sachant qu'elle est en panne depuis l'instant initial. Pour l'étude de la maintenabilité du système, nous nous sommes contentés d'évaluer le MTTR (mean time to repair). Nous avons calculé le MTTR comme l'espérance mathématique de la durée de réparation. Nous avons donc approché le MTTR par la moyenne du temps d'accès aux états de fonctionnement (MoyTAEF) sur l'ensemble des histoires simulées (une histoire est le passage du système des états défaillants 7 et 11 vers l'état de bon fonctionnement). Nous avons considéré une précision $\varepsilon = 0.1$ et une probabilité de convergence $\theta = 0.9$ (équations (2.15) et (2.16) pour la mesure considérée MoyTAEF qui approche de manière asymptotique le MTTR. On obtient ainsi :

$$MTTR = \quad 25594{,}89\ h$$

Les résultats montrent qu'il suffit de simuler 90 histoires, la durée de simulation étant de l'ordre de 6 minutes pour atteindre la précision désirée. La figure 3.17 présente ces résultats.

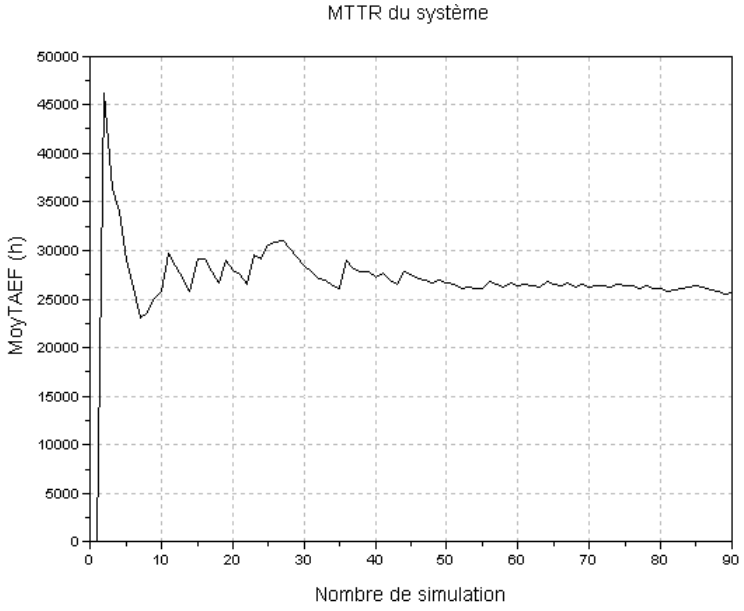


Figure 3.17 – Temps moyen d'accès à l'état de fonctionnement.

3.2.2.5 Conclusion

L'automate stochastique hybride nous a permis de modéliser et simuler le vieillissement et la défaillance des composants en utilisant différentes lois de probabilités. A travers la loi de Weibull nous a modélisé ce comportement et grâce au paramètre η nous avons pu dégrader les contrôleurs PI et TOR jusqu'à leur mort. La loi normale et la loi uniforme ont été utilisées pour déterminer la durée de bon fonctionnement du four. La défaillance de celui-ci intervient à cause de deux différents types de comportement : le

temps d'utilisation en régime normal (contrôleur PI), ou bien le nombre de commutations (fonctionnement en mode de secours : TOR). La loi exponentielle a été utilisée pour modéliser les temps de réparations et de remplacement des composants. Au moyen de l'automate stochastique hybride nous avons pu accéder aux grandeurs de la sûreté de fonctionnement : la fiabilité, la disponibilité et la maintenabilité du système dynamique hybride proposé. Nous avons pu montrer la faisabilité de l'approche et il reste possible de modifier les paramètres afin d'obtenir des résultats plus réalistes. Par rapport à la façon de modéliser la dégradation des contrôleurs nous avons dégradé le paramètre η de manière linéaire mais nous pourrons évidemment étudier des comportements non linéaires.

Nous avons vu également (cf. calcul de la fiabilité) que la précision exigée sur l'estimation d'un paramètre a une incidence directement proportionnelle sur le temps de calcul. En demandant des précisions peu élevées, on obtient néanmoins de bonnes indications de tendance.

3.3 Cas test 2 : système de contrôle de niveau du liquide d'un réservoir

Comme nous l'avons dit en introduction du chapitre, ce cas test correspond au benchmark présenté par de nombreux auteurs : [Aldemir, 1987], [Dutuit *et al.*, 1997], [Marseguerra et Zio, 1996], [Kermish et Labeau, 2000a] et [Zhang *et al.*, 2008]. Il consiste en un réservoir contenant un liquide dont le niveau *h* doit être maintenu à l'aide d'une pompe principale P1, d'une pompe de secours P2 et d'une vanne de vidange V. Chacun de ces trois composants est commandé par une boucle de contrôle contenant un détecteur de niveau, figure 3.18.

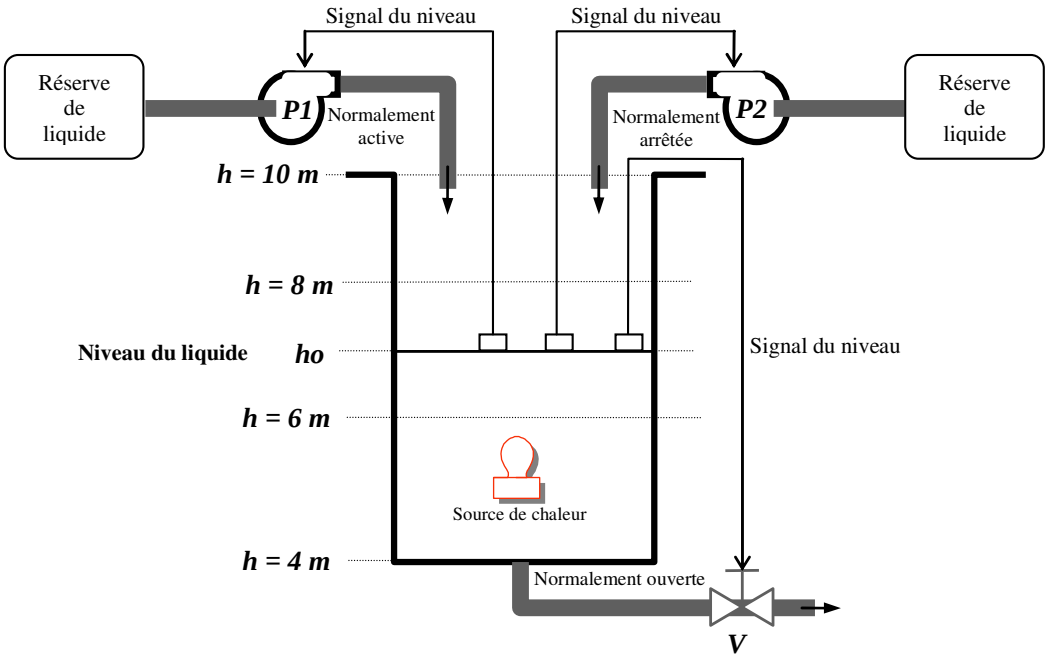


Figure 3.18 – Réservoir et sa régulation de niveau.

Une source de puissance thermique constante chauffe le liquide dont la température évolue ainsi avec les débits d'entré et de sortie. On suppose de plus que les taux de défaillance des composants peuvent dépendre de la température. Les expressions suivantes montrent cette dépendance :

$$\lambda^c = a(T)\hat{\lambda}^c, \quad c = \text{P1, P2, V} \tag{3.39}$$

$$a(T) = \frac{b_1 e^{b_c (T-20)} + b_2 e^{-b_d (T-20)}}{(b_1 + b_2)} \tag{3.40}$$

où *a(T)* est une fonction de la température, figure 3.19, et les paramètres respectifs sont :

- $\lambda_{P1} = 2{,}2831.10^{-3}h^{-1}$; $\lambda_{P2} = 2{,}8571.10^{-3}h^{-1}$; $\lambda_V = 1{,}5625.10^{-3}h^{-1}$

- $b_1 = 3{,}0295$, $b_2 = 0{,}7578$, $b_c = 0{,}05756$ et $b_d = 0{,}2301$

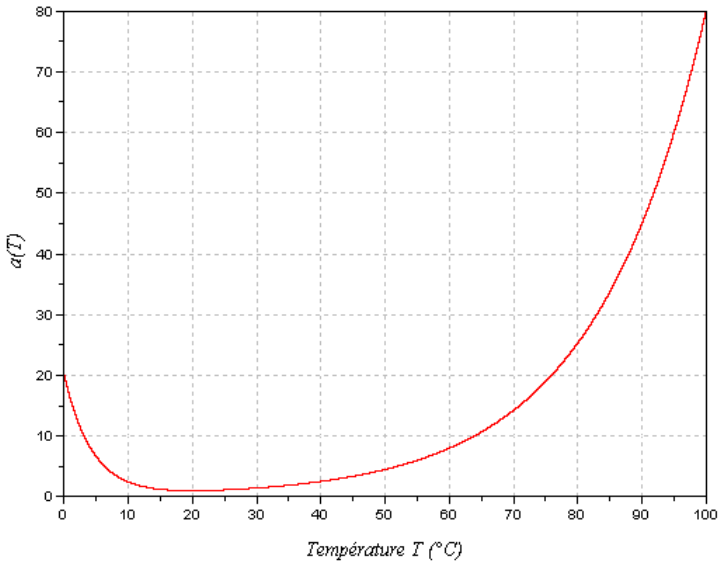


Figure 3.19 – La fonction *a(T)*.

La vanne V permet de vider le réservoir avec un débit donné, tandis que les pompes P1 et P2 en assurent le remplissage. La mission à remplir par cette installation est de maintenir le niveau de liquide à l'intérieur d'un intervalle h∈ [6,8], afin d'éviter deux situations extrêmes : l'assèchement et le débordement. L'un de ces cas, par exemple, est susceptible de se produire lorsque les débits des pompes P1 et P2 ne compensent plus celui de la vanne V.

Pour tenter d'éviter ces modes de défaillance, des capteurs de niveau indépendants sont associés à chacun des composants. Lorsque le niveau descend sous un seuil (h<6) les deux pompes sont normalement mises en marche, tandis que la vidange de l'eau par la vanne V

est arrêtée. Dans le cas contraire où le niveau excède le seuil (h>8), ce sont les pompes qui sont arrêtées alors que la vidange du réservoir est maintenue (tableau 3.4).

Niveau *h*	**Pompe 1**	**Pompe 2**	**Vanne**
$h < 6$ m	active	active	fermée
$6 \text{ m} \leq h \leq 8$ m	active	arrêtée	ouverte
$h > 8$ m	arrêtée	arrêtée	ouverte

Tableau 3.4 – Conditions de fonctionnement nominal.

Les trois composants sont mutuellement indépendants et non réparables. Les différents modes de défaillance pour chaque composant sont pris en compte, figure 3.20 : le comportement intempestif (état 0 → état 3, état 1 → état 2) et le blocage en l'état courant (état 0 → état 2, état 1 → état 3). Leurs durées de fonctionnement avant défaillance sont des variables aléatoires qui suivent des lois exponentielles.

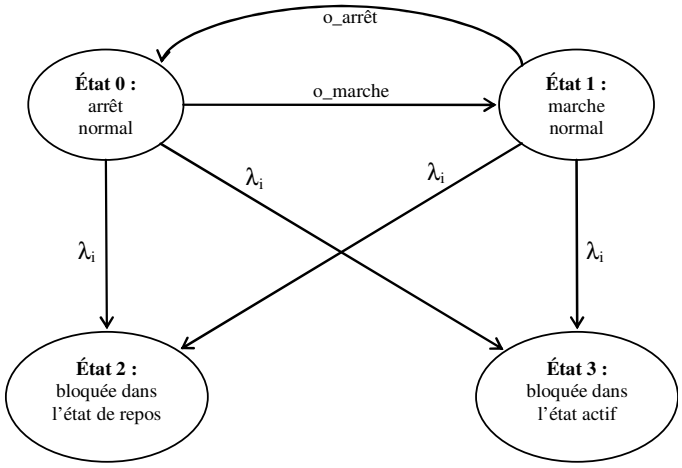


Figure 3.20 – États des composants et transitions entre états.

Les variables continues pour le système sont : le niveau du liquide *h* et la température *T* du fluide lesquelles sont en fonction de l'état des composants. Ainsi, les variables *h(t)* et *T(t)* satisfassent les équations différentielles suivantes :

$$\frac{dh(t)}{dt} = \gamma_1(\upsilon) \tag{3.41}$$

$$\frac{dT(t)}{dt} = (\gamma_2(\upsilon) - \gamma_3(\upsilon)T)/h \tag{3.42}$$

où $\upsilon = (\upsilon_{P1}, \upsilon_{P2}, \upsilon_V)$ et les composants $c \in$ {P1, P2, V}. Ainsi,

$$\upsilon_c = \begin{cases} 0 & \text{si c est OFF ou bloqué en OFF} \\ 1 & \text{si c est ON ou bloqué en ON} \end{cases} \tag{3.43}$$

avec

- $\gamma_1(\upsilon) = (\upsilon_{P1} + \upsilon_{P2} - \upsilon_V)G$
- $\gamma_2(\upsilon) = (\upsilon_{P1} + \upsilon_{P2})GT_{in} + 23{,}88915$
- $\gamma_3(\upsilon) = (\upsilon_{P1} + \upsilon_{P2})G$
- G = 1,5 m^3h^{-1} (débit des composants : $G_{P1} = G_{P2} = G_V = G$)
- T_{in} = 15°C (température du liquide quand il entre dans le réservoir)

Les équations généralisées (3.41) et (3.42) reflètent les différents modes opératoires possibles du processus. Elles font apparaître l'influence des phénomènes discrets sur l'évolution du processus au travers des termes υ_c. Ces derniers peuvent prendre la valeur 1 si l'actionneur associé est commandé en ouverture ou active ou bien si une défaillance de cet actionneur le bloque dans la position ouverte ou active. La valeur 0 est prise dans le cas contraire, comme l'exprime l'équation (3.43). En conditions nominales, les débits pour les trois composants est le même. Au temps $t = 0$, le niveau du liquide h est 7 m, la température est 30,9261 °C, la pompe P1 fonctionne, la pompe P2 est à l'arrêt et la vanne V est ouverte.

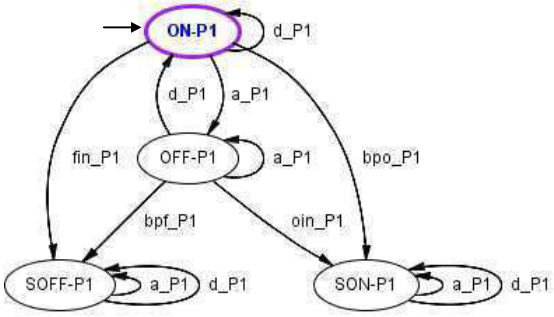
Le but est de déterminer la probabilité pour que le système arrive aux états redoutés : à l'assèchement $(h \leq 4\ m)$, au débordement $(h \geq 10\ m)$ et à la haute température $(T \geq 100\ °C)$.

3.3.1 Premier cas : système du réservoir sans prendre en compte la température

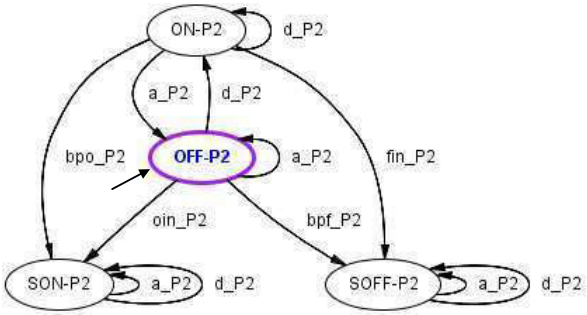
Ce cas test a été d'abord traité sans tenir compte de l'effet de la température sur les composants. La dépendance entre les variables T et h ne sera pas prise en compte. Ainsi, le modèle présenté du cas test a été simplifié en utilisant uniquement la variable du niveau du liquide h. Donc, dans l'équation (3.39) $a(T) = 1$ et les taux de défaillances des composants sont constants.

3.3.1.1 Implémentation

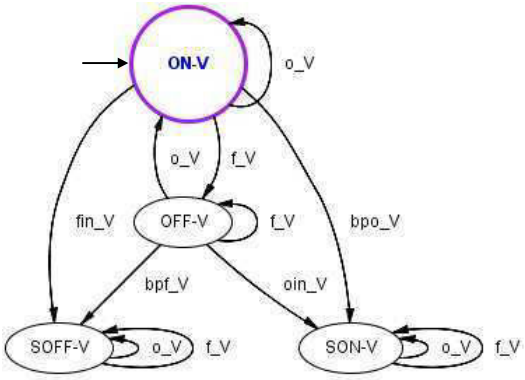
Pour implémenter l'automate stochastique hybride du système, nous avons défini d'abord cinq automates à états finis élémentaires (figure 3.21) : trois pour les composants, un pour la cuve et un dernier pour la commande. A l'aide du logiciel DESUMA [DESUMA], nous avons réalisé la composition synchronisée de ces cinq automates afin d'obtenir formellement l'automate global.



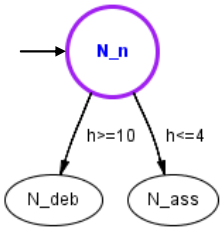
(a) Automate de la pompe 1



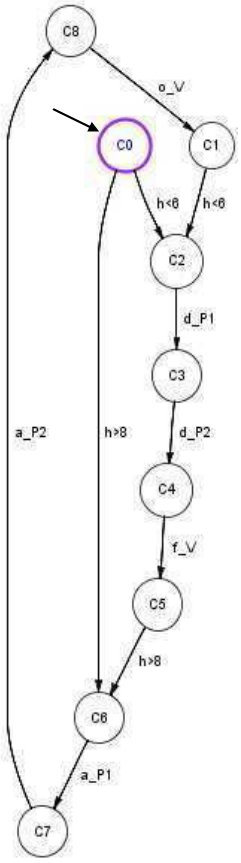
(b) Automate de la pompe 2



(c) Automate de la vanne



(d) Automate de la cuve



(e) Automate de la commande

Figure 3.21 – Automates à états finis élémentaires.

L'ensemble des événements associés à la pompe P1, à la pompe P2, à la vanne V et à la commande est :

- bpo_P1, bpo_P2 et bpo_V : pompes bloquées dans l'état actif et vanne bloquée dans l'état ouvert,

- bpf_P1, bpf_P2 et bpf_V : pompes bloquées dans l'état arrêt et vanne bloquée dans l'état fermé,
- oin_P1, oin_P2 et oin_V : démarrage intempestif des pompes et ouverture intempestive de la vanne,
- fin_P1, fin_P2 et fin_V : arrêt intempestif des pompes et fermeture intempestive de la vanne,
- a_P1 et a_P2 : arrêt des pompes P1 et P2,
- d_P1 et d_P2 : démarrage des pompes P1 et P2,
- o_V : ouverture de la vanne,
- f_V : fermeture de la vanne.

Nous distinguons pour les composants P1, P2, V les états suivants :

- ON-P1, ON-P2 et ON-V : pompes actives et vanne ouverte,
- OFF-P1, OFF-P2 et OFF-V : pompes en arrêt et vanne fermée,
- SON-P1, SONP2 et SON-V : pompes bloquées dans l'état actif et vanne bloquée dans l'état ouvert,
- SOFF-P1, SOFF-P2 et SOFF-V : pompes bloquées dans l'état arrêt et vanne bloquée dans l'état fermé.

Pour le réservoir nous avons les états ;

- N_n : niveau normal du réservoir ($6 \leq h \leq 8$),
- N_ass : niveau d'assèchement ($h \leq 4$) et
- N_deb : niveau de débordement ($h \geq 10$).

Les lois de contrôle de la commande (tableau 3.4) :

- initial : C0 – P1 active, P2 arrêtée et V ouverte,
- si h<6 : C2 → C3, P1 active ; C3 → C4, P2 active et C4 → C5, V fermée,
- si h>8 : C6 → C7, P1 arrêtée ; C7 → C8, P2 arrêtée et C8 → C1, V ouverte,
- C1 – état stable de la commande quand h>8 et
- C5 – état stable de la commande quand h<6.

3.3.1.2 Résultats

La composition synchronisée des cinq automates présentés ci-avant a donné comme résultat un automate global de 873 états. Nous avons simplifié cet automate à partir de règles simples, afin d'obtenir formellement l'automate global du système : d'abord, comme le but de notre application est de déterminer la probabilité d'arriver aux états redoutés, nous avons groupé dans un état tous les états qui ont comme niveau du réservoir la valeur correspondant à l'assèchement N_ass (un des états redoutés). Nous avons appliqué la même procédure pour l'autre état redouté N_deb. Après, nous avons regroupé les états de la séquence C2 $\rightarrow$ C3 $\rightarrow$ C4 $\rightarrow$ C5, laquelle correspond à l'action de la commande quand le niveau de liquide atteint le seuil h<6. De même pour la séquence C6 $\rightarrow$ C7 $\rightarrow$ C8 $\rightarrow$ C1 quand le niveau du liquide atteint h>8 (voir les lois de contrôle de la commande et l'automate de la commande de la figure 3.21e). Comme résultats de ces deux règles simples, nous avons obtenu un automate global final de 83 états. A partir de cet automate, nous avons implémenté l'automate stochastique hybride pour le système.

Pour évaluer la probabilité d'arriver aux états redoutés (assèchement et débordement du réservoir) nous avons procédé à une série de simulations de Monte Carlo. La simulation est arrêtée si les conditions données par les équations (2.15) et (2.16) sont vérifiées.

Nous avons utilisé une valeur de $\varepsilon = 0{,}0001$ et de $\theta = 0.8$ (80% des histoires ont vérifié le critère). Le tableau 3.5 montre les résultats que nous avons obtenus (ASH) ainsi que les résultats obtenus par Zhang *et al.* [Zhang *et al.*, 2008] (PDMP et RdP). Notons que les résultats obtenus par RdP (sous Moca RP) correspondent à une simulation de 10^6 histoires. La simulation par ASH avec le critère ci-dessus n'a jamais nécessité plus de 12000 histoires.

Temps (heures)	**Débordement**			**Assèchement**		
	PDMP	**RdP**	**ASH**	**PDMP**	**RdP**	**ASH**
200	0,213	0,202	0,1913	0,026	0,023	0,0253
400	0,368	0,364	0,3682	0,068	0,067	0,0669
600	0,439	0,438	0,4365	0,097	0,096	0,0970
800	0,472	0,471	0,4739	0,111	0,110	0,1199
1000	0,486	0,486	0,4869	0,118	0,118	0,1178

Tableau 3.5 – Probabilités d'accès aux états redoutés.

Nous avons également calculé la probabilité cumulée des événements de débordement et d'assèchement (sur 10^4 histoires) afin de la comparer à celles trouvées par PDMP (les

fichiers de données nous ayant été aimablement transmis par les auteurs). La figure 3.22 montre ces résultats. Les deux courbes sont assez proches.

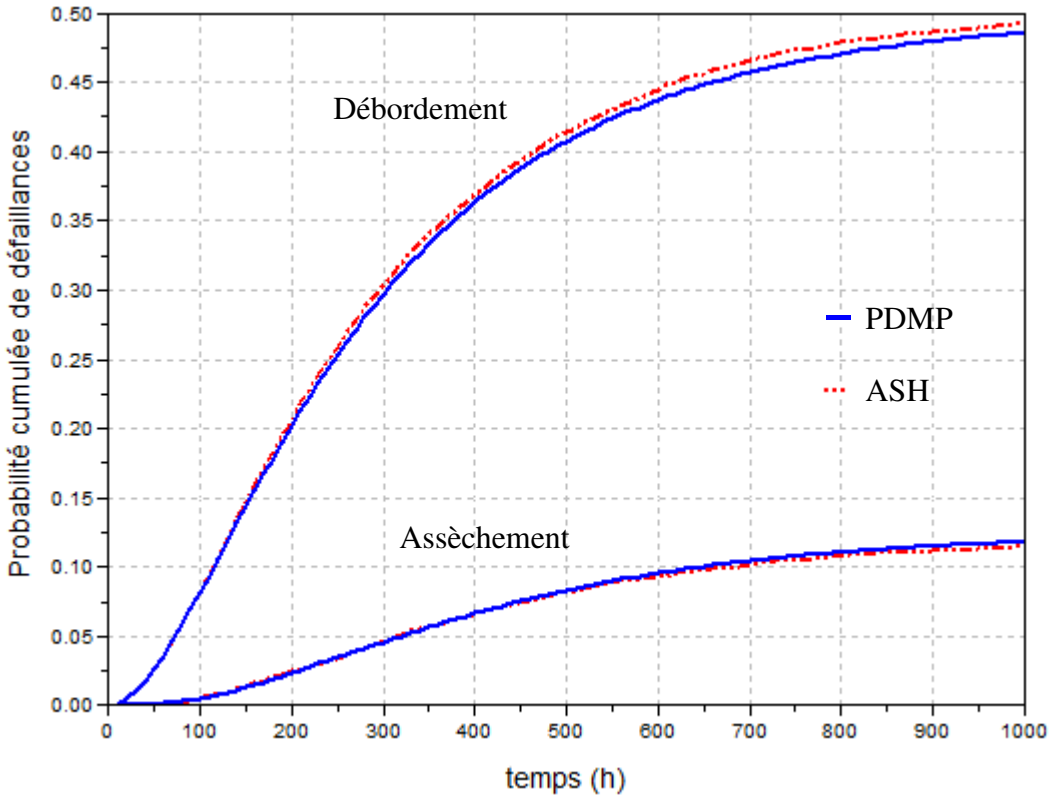


Figure 3.22 – Superposition des résultats des probabilités cumulées de défaillances par PDMP et ASH.

Nous avons effectué plusieurs simulations de Monte Carlo avec un nombre d'histoires par simulation de 1000, 5000, 10000 et 12000. Les résultats sont assez proches les uns des autres, ce qui a tendance à montrer qu'il n'est peut être pas nécessaire de faire un nombre très importants d'histoires. La figure 3.23 montre la superposition de ces résultats par référence toujours avec ceux obtenus par PDMP.

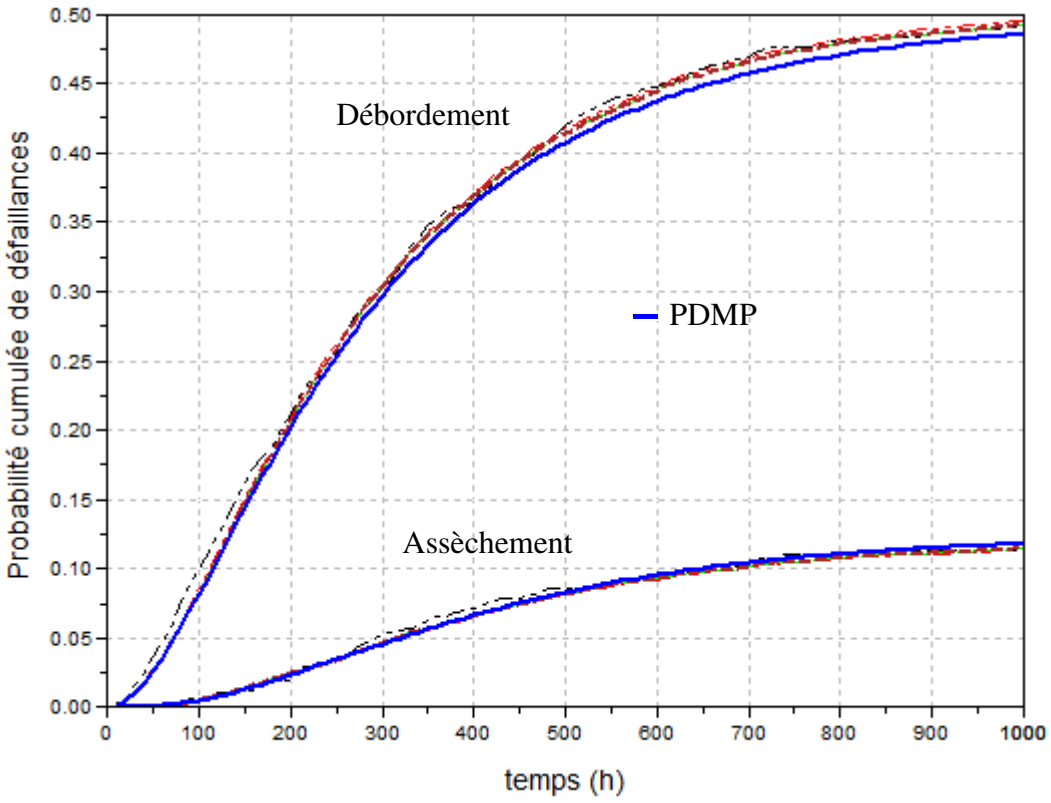


Figure 3.23 – Superposition des résultats des probabilités cumulées de défaillances avec 10^3, $5{\cdot}10^3$, 10^4 et $12{\cdot}10^3$ histoires par ASH et 10^6 histoires par PDMP.

La figure 3.24 montre pour rappel les résultats publiés par Marseguerra et Zio [Marseguerra et Zio, 1996] également par simulation de Monte Carlo qui sont également proches (la courbe extraite de l'article est peu précise).

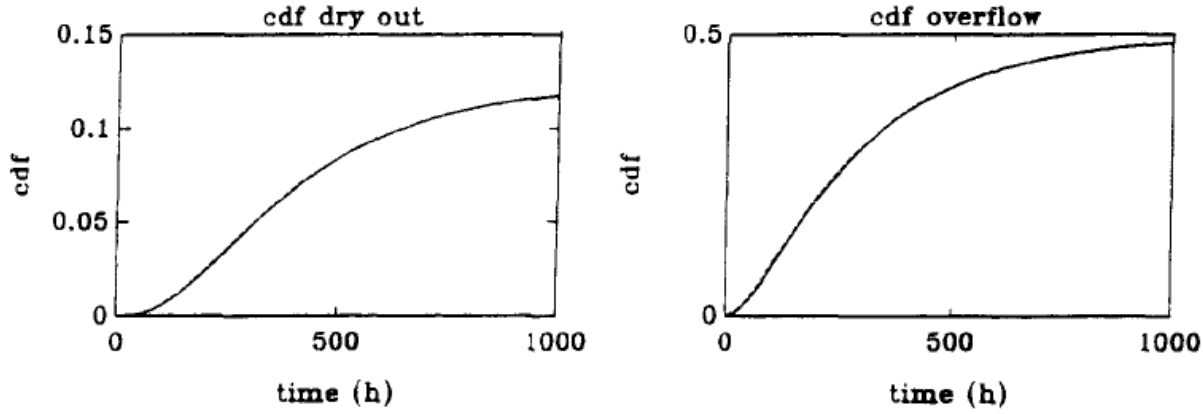


Figure 3.24 – Résultats publiés par Marseguerra et Zio [Marseguerra et Zio, 1996]

3.3.1.3 Conclusion

Notre approche nous a permis d'évaluer la probabilité d'occurrence des événements redoutés du cas test. Nous constatons une bonne convergence des résultats pour la probabilité de débordement et d'assèchement. Pour confirmer cette première impression, le pas suivant sera d'appliquer notre approche au même benchmark mais maintenant en prenant en compte la dépendance entre taux de défaillance λ_c et température T du liquide sous la forme $\lambda_c = f(T)$. Cela implique d'ajouter l'automate décrivant l'événement dépassement de température, de générer le nouvel automate global, de réduire cet automate et de modifier le descripteur de modes et le générateur aléatoire pour incorporer la dépendance de *f(T)*. Les temps de simulation ne devraient pas augmenter considérablement.

3.3.2 Deuxième cas : système du réservoir en prenant en compte la température

3.3.2.1 Automate de la température

Nous prenons maintenant en compte la dépendance entre taux de défaillance des composants (la vanne et les deux pompes) et température dans la cuve. Pour cela nous reprenons le modèle défini dans le paragraphe §3.3 en ajoutant un nouvel automate aux automates à états finis élémentaires de la figure 3.21. Cet automate modélise la détection de l'événement dépassement de température, comme l'indique la figure 3.25.

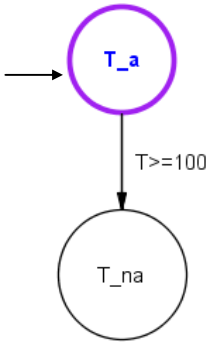


Figure 3.25 – Automate à état fini de la température.

Les états de l'automate de la température sont :

- T_a : état initial ; température acceptable pour le système,
- T_na : température non acceptable pour le système.

L'événement est :

- T>=100 : dépassement du seuil de la température permise (valeur prise par Marseguerra et Zhang).

De la même façon que pour le premier cas, nous nous sommes servis du logiciel DESUMA avec le but d'effectuer la composition synchronisée des six automates à états finis simples afin d'obtenir formellement l'automate global. L'automate résultant de la synchronisation présente 1746 états. Nous avons procédé pour l'état T_na de la même manière que nous avons fait pour les états associés au niveau du liquide N_ass et N_deb. Pour tous les trois états redoutés nous avons groupé dans un seul état tous les états pour lesquels on a un assèchement (N_ass) tous ceux pour lesquels on a un débordement (N_deb) et tous ceux pour lesquels on a un dépassement de température (T_na). Ainsi, le nombre d'états de l'automate global obtenu est ramené à 84 états. A partir de cet automate global nous avons complété le modèle pour obtenir l'automate stochastique hybride correspondant.

Précisons comment la simulation prend en compte l'action de la température sur les temps avant défaillance. A chaque entrée dans un nouvel état discret de l'automate, on tire aléatoirement une variable uniformément distribuée sur [0,1] pour chaque transition stochastique de sortie de cet état. Ces valeurs sont conservées pour toute la durée de séjour dans l'état. A chaque pas d'intégration numérique dans cet état discret, on recalcule les taux de transition en tenant compte de la température et on recalcule l'instant d'occurrence potentiel de chacun des événements aléatoires actifs dans l'état. On répète jusqu'à l'occurrence d'un de ces événements, à moins qu'un événement déterministe n'intervienne avant.

3.3.2.2 Résultats

Le tableau 3.6 et la figure 3.26 donnent les probabilités cumulées des événements redoutés que nous avons obtenues pour 10^4 histoires avec l'ASH comparativement aux résultats obtenus par les processus markovien déterministe par morceaux (PDMP) [Zhang *et al*., 2008].

Temps (heures)	Assèchement		Débordement		Haute température	
	PDMP	ASH	PDMP	ASH	PDMP	ASH
100	0.028463	0.0387	0.205090	0.1942	0.094659	0.0678
200	0.059185	0.0781	0.336688	0.3371	0.143356	0.1113
300	0.077217	0.1007	0.405502	0.4109	0.157475	0.1251
400	0.086785	0.1122	0.441659	0.4486	0.161640	0.129
500	0.091610	0.1184	0.461626	0.4697	0.162823	0.1299
600	0.093929	0.1208	0.473156	0.4809	0.163193	0.1307
700	0.095065	0.1221	0.479960	0.4882	0.163312	0.1308
800	0.095633	0.1228	0.484020	0.4925	0.163342	0.1308
900	0.095908	0.1232	0.486432	0.4951	0.163363	0.1308
1000	0.096032	0.1234	0.487753	0.4963	0.163368	0.1308

Tableau 3.6 – Les probabilités des événements redoutés.

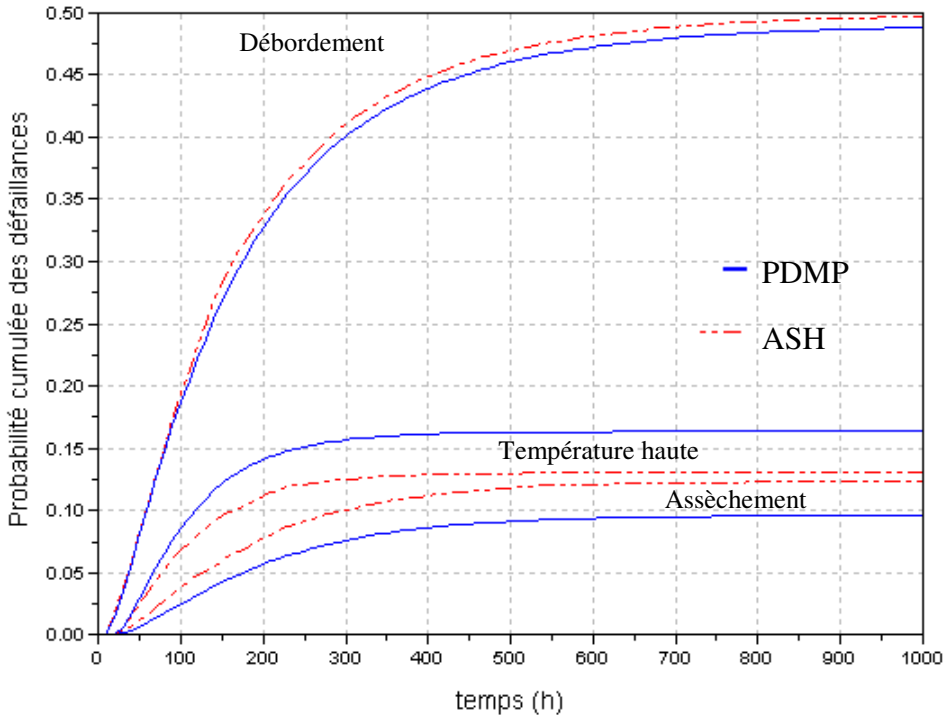


Figure 3.26 – Superposition des résultats des probabilités cumulées des défaillances par PDMP et ASH en tenant compte de la température.

Afin d'étudier la sensibilité du modèle à certains paramètres, nous avons effectué plusieurs simulations dans des conditions différentes.

La figure 3. 27 montre les résultats obtenus en modifiant les seuils de débordement et d'assèchement. Le seuil de débordement est diminué d'un mètre et le seuil d'assèchement est augmenté également d'un mètre.

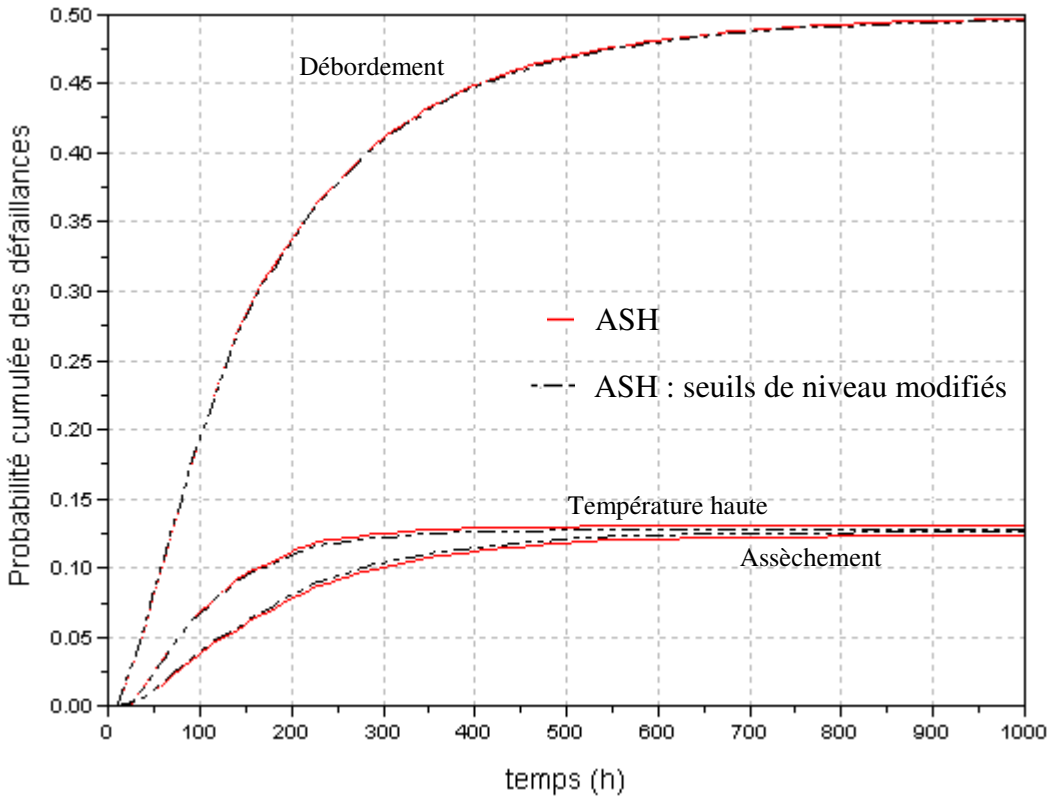


Figure 3.27 – Résultats des probabilités cumulées des défaillances avec modification des seuils de niveau dans la cuve.

La figure 3.28 montre les résultats correspondant à un seuil de dépassement de température de 100 °C, de 85°C et de 150°C en maintenant les seuils de niveau aux valeurs initiales (10m et 4m).

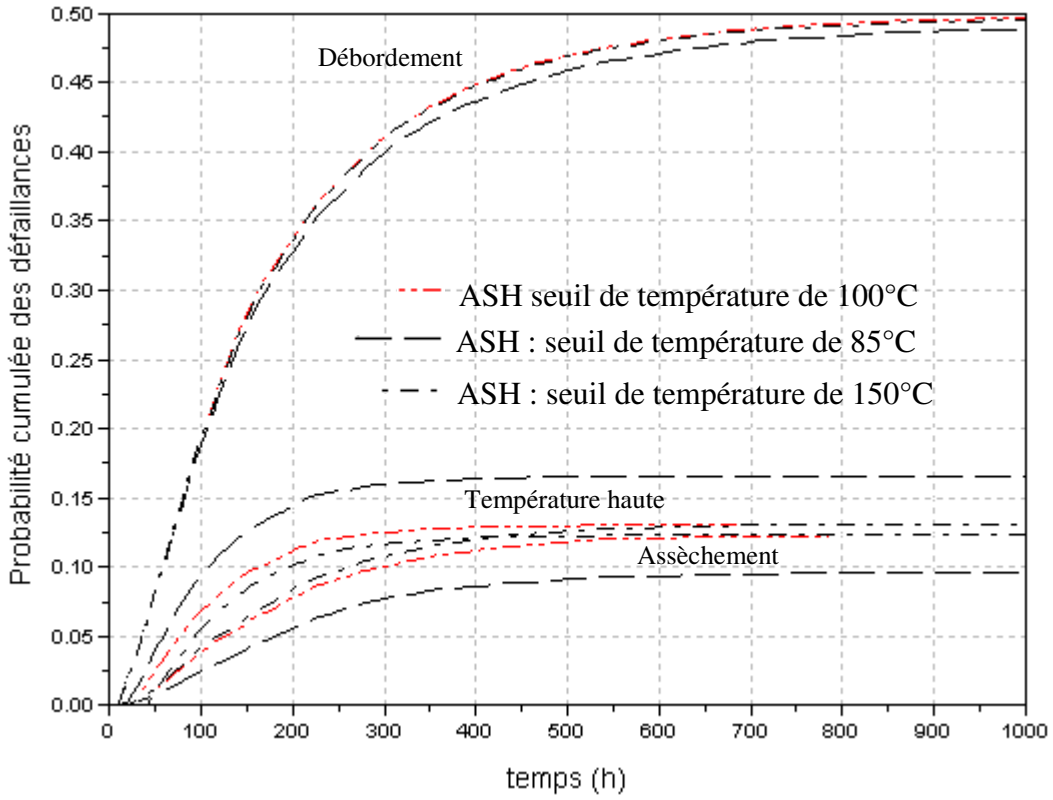


Figure 3.28 – Résultats des probabilités cumulées des défaillances avec modification du seuil de température de la cuve.

3.3.2.3 Conclusion

Comme on a pu le voir sur la figure 3.26, les résultats obtenus pour le débordement sont proches de ceux publiés par Marseguerra et Zhang. Par contre, les courbes correspondant aux événements d'assèchement et de température élevée divergent sensiblement. Nous ne sommes pas en mesure pour le moment d'expliquer ces différences. Nous pouvons cependant faire la remarque suivante : dans notre automate, nous ne considérons pas que la première occurrence d'un événement redouté soit un état puits puisque par construction, il est possible d'avoir successivement deux de ces événements, l'un n'interdisant pas l'occurrence d'un autre. Après assèchement ou débordement, l'occurrence du dépassement de température est possible et réciproquement (ce sont des variables physiques différentes et couplées qui génèrent ces événements). Par exemple, nous avons constaté qu'après assèchement, le test de température est quasiment toujours vérifié (ce qui est assez logique) et dans le cas du débordement, il ne l'est quasiment jamais. Nos courbes représentent les probabilités d'occurrence du premier événement redouté dans la séquence. Nous ne savons pas aujourd'hui comment les autres approches ont intégré cet aspect. Il sera intéressant d'en débattre prochainement avec les auteurs.

L'étude de la variation des paramètres du modèle montre une sensibilité faible aux paramètres de niveau. Par contre, on note une sensibilité assez importante au paramètre

température. Avec un seuil de 85°C, la probabilité d'assèchement diminue sensiblement du fait que la probabilité d'avoir d'abord un dépassement de température augmente.

Nous touchons là un problème important de la fiabilité dynamique. En effet, on voit qu'il est indispensable de raisonner en termes de séquences possibles d'événements (dans notre modèle, le langage de l'automate) et non plus en termes de coupes. En effet, certains événements étant associés à des variables continues couplées, ils peuvent avoir lieu dans n'importe quel ordre (il n'y a pas d'occurrence simultanée dans un automate) et il convient de ne pas arrêter la simulation sur la détection du premier. Les probabilités à retenir seraient sans doute celles correspondant aux différentes séquences possibles. Par la construction de notre automate, nous pourrons identifier les états terminaux associés à chacune de ces séquences et par conséquent en déduire leur probabilité. Nous nous proposons de développer cet aspect dans une proche perspective.

3.3.3 Durée de la simulation

Nous voulons présenter quelques remarques par rapport à la durée des simulations effectuées sur le cas du réservoir.

Nous avons effectué une simulation de Monte Carlo du système du réservoir en utilisant la fonction scicosim en mode batch (§2.2.5) sous Scilab. Nous nous sommes rendu compte que les simulations effectuées sur la plate-forme Linux sont considérablement plus rapides que sur la plate-forme Windows. Malgré que nous ayons utilisé la plate-forme Linux, la durée de la simulation de Monte Carlo appliquée pour chaque temps spécifié sur les tableaux 3.5 et 3.6 est de l'ordre de 5 heures. Cela peut donner l'impression que notre approche est peu performante comme nous l'avons indiqué avant (§2.5). Cependant, nous avons analysé cette situation et nous pouvons faire au moins trois remarques :

- La durée de la simulation d'une histoire du modèle Scicos du réservoir exécutée directement sur la boîte à outils Scicos est de l'ordre d'environ de 0.4 seconde. Or, une simulation de Monte Carlo faite directement sur Scicos n'est pas possible. Celle-ci n'est possible que via Scilab à travers la fonction scicosim.

- La durée de la simulation de la même histoire du modèle Scicos sous Scilab à travers la fonction scicosim est de l'ordre de 2 minutes ! Pour raccourcir le temps nécessaire, nous avons alors ajouté au modèle Scicos le bloc STOP pour forcer l'arrêt de la simulation lorsque le système arrive à un état redouté. Cependant, cela entraîne la nécessité d'une recompilation et d'une réinitialisation du modèle Scicos pour une deuxième histoire. Cela entraîne une augmentation considérable de la durée de simulation.

- La durée de la simulation d'une histoire du même modèle Scicos depuis Scilab à travers la fonction scicos_simulate est de l'ordre de 58 seconds, si la simulation n'est pas arrêtée par le bloc STOP dans le modèle Scicos. L'avantage de tout cela, c'est qu'il n'y aura pas besoin d'une recompilation pour une deuxième histoire du modèle Scicos. Cela réduit la durée de la simulation de Monte Carlo pour le système.

Finalement, la simulation de Monte Carlo sans bloc STOP et en utilisant scicosim dans l'environnement Scilab prend en moyenne 5 heures pour 10000 histoires.

Si nous pouvions effectuer directement la simulation de Monte Carlo sur Scicos nous aurions : 10000 histoires x 0.4 seconds de durée de simulation par histoire soit 66 minutes.

3.4 Conclusion

Nous avons montré l'efficacité de notre approche de simulation pour l'évaluation de la sûreté de fonctionnement en contexte dynamique. Elle permet de répondre à tous les problèmes posés et notamment la dépendance entre l'état discret et l'état continu tant au point de vue déterministe qu'au point de vue stochastique. D'autres aspects peuvent également être pris en compte comme par exemple les caractéristiques probabilistes du diagnostic et l'impact du contrôle par supervision sur la sûreté de fonctionnement. Ces points seront abordés au prochain chapitre.

Il reste néanmoins qu'un effort doit être fait pour rechercher les modalités d'accélération de la simulation. On devra par exemple chercher à optimiser les algorithmes numériques pour la détection des passages par zéro plutôt que pour la détermination précise des trajectoires de l'état continu. On pourra aussi mettre à profit les différentes échelles de temps présentes entre les aspects fonctionnels et dysfonctionnels. La construction d'un module ASH directement exécutable sous Scilab permettrait aussi d'accélérer la simulation de Monte Carlo.

Chapitre 4

Contrôle par supervision

4.1 Introduction

La théorie de la supervision des systèmes à événements discrets (SED) a été initiée par Wonham en 1987 [Ramadge et Wonham, 1987]. Dans cette approche le temps n'intervient pas et l'étude se place à un niveau qualitatif (validation des trajectoires de commande). Le système à traiter est considéré comme un système à événements discrets qui évolue seulement au gré de l'occurrence événements. Son fonctionnement peut être décrit par un ensemble de séquences de ces événements, ensemble qui constitue le langage formé sur l'alphabet des événements. Un superviseur (qui est aussi un SED), observe les événements qui se succèdent dans un système à événements discrets et est capable d'autoriser ou d'interdire l'occurrence des prochains événements attendus du système pour en modifier globalement le comportement (réduction du langage initial à un langage autorisé).

[Cassandras et Lafortune, 1999] présentent la théorie de la supervision dont le système à événements discrets est modélisé au niveau logique (non temporisé) par un automate $\mathcal{G}$ dont le comportement n'est pas satisfaisant. Nous reprenons ici les concepts introduits par ces auteurs. $\mathcal{E}$ est l'ensemble d'événements de $\mathcal{G}$ dont l'espace des états n'est pas nécessairement fini. Le principe est que ce comportement n'est pas satisfaisant et qu'il doit par conséquent être modifié. Modifier le comportement du système à événements discrets modélisé par $\mathcal{G}$ signifie restreindre le langage de ce système à un sous ensemble de $\mathcal{L}(\mathcal{G})$. Pour altérer le comportement de $\mathcal{G}$, on introduit un superviseur $\mathcal{S}$. Le comportement de $\mathcal{G}$ est modifié par l'équivalent d'une sorte de « boucle de réaction » (figure 4.1) pour obtenir un nouveau comportement conforme à un ensemble de spécifications.

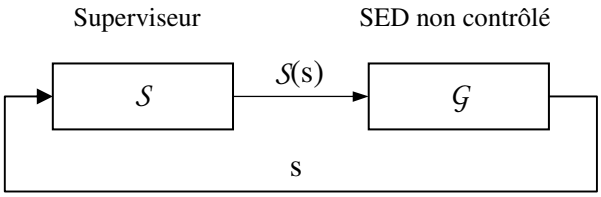


Figure 4.1 – Le superviseur et le SED [Cassandras et Lafortune, 1999].

Nous avons développé et implémenté dans l'environnement Scicos un modèle de système de contrôle par supervision. Ce système est composé de trois blocs Scicos : un identificateur d'événements observables, un automate simple et un superviseur. Le bloc automate a été modifié en ajoutant une entrée. Le superviseur apportera à cette entrée la liste des événements à invalider. Pour identifier l'occurrence des événements observables nous nous sommes servi des blocs identificateur d'événements observables et du bloc automate simple.

L'objectif est de disposer d'un outil informatique capable de simuler le problème du diagnostic par supervision. C'est-à-dire, qu'à partir de l'observation d'une séquence occurrence d'événements observables nous cherchons à savoir si un événement non observable, comme la défaillance d'un composant, s'est produit.

4.2 Principe général du contrôle par supervision

Un système à événements discrets couplé à un superviseur peut être perçu comme recevant à tout instant la liste des événements dont l'occurrence est autorisée. Ce système est appelé système à événements discrets supervisé, c'est-à-dire, que dans ce système l'occurrence d'un événement ne peut avoir lieu que si elle est autorisée par le superviseur. Fondamentalement, l'observation du système par le superviseur est asynchrone, c'est-à-dire non mesurée par une horloge. L'occurrence d'un événement observable conduit le superviseur dans un nouvel état qui donne immédiatement la liste des événements contrôlables dont l'occurrence est validée. Modifier le comportement du système modélisé par $\mathcal{G}$ c'est donc restreindre le langage de ce système à événements discrets à un sous ensemble de $\mathcal{L}(\mathcal{G})$.

Certains événements générés par le système ne peuvent pas être interdits, un tel événement sera appelé événement non contrôlable. En revanche, nous appellerons événement contrôlable tout événement qui peut être interdit à tout moment. Un événement non contrôlable est un événement non prévisible, comme une défaillance ou un aléa. Il peut aussi être dû à une contrainte matérielle, comme une limitation de capteurs, une saturation ou un débordement d'opérations. Les tops d'horloge ou les interruption prioritaires dans un système informatique sont des événements non contrôlables, de même que le changement asynchrone d'une entrée. Par ailleurs, les phénomènes de limitation de capteurs et les

événements locaux non transmis dans les systèmes distribués, peuvent être des exemples d'événements non observables.

4.3 Automates non déterministes

Au niveau d'abstraction logique (non temporisé), le comportement d'un système à événements discrets est décrit par les suites ordonnées des événements qui le caractérisent (pas de dates, pas de durée d'intervalle de temps). Ce « langage » peut être vu comme la trace des différents chemins de l'automate à états finis modélisant ce système.

Généralement, un système à événements discrets démarre dans un état bien spécifié et est déterministe : l'occurrence (réception et/ou production) d'un événement correspond au passage d'un état dans un autre sans ambiguïté. Le fait de considérer plusieurs états finaux (marqués) permet de modéliser l'existence des différentes tâches à exécuter par le système à événements discrets. Si un état final est atteint, la tâche correspondante est achevée.

Pour un automaticien le problème n'est pas de représenter un langage par un automate, mais plutôt l'inverse. C'est-à-dire, on identifie les états du système à événements discrets et les transitions possibles et on étudie les propriétés du langage de cet automate. L'intérêt des automates à états finis pour modéliser les systèmes à événements discrets est de permettre une résolution aisée des problèmes d'analyse de leurs comportements.

La présence d'événements non observables (invisibles pour un observateur externe) pouvant causer des changements dans l'état interne d'un SED (la défaillance d'un composant, par exemple) amène à définir les automates non déterministes.

Un automate non déterministe est défini comme :

$$\mathcal{A}_{nd} = (X, E \cup \{\varepsilon\}, f_{nd}, x_0, x_f) \tag{4.1}$$

avec la fonction de transition $f_{nd} : X \times E \cup \{\varepsilon\} \rightarrow 2^X$ (2^X est l'ensemble de tous les sous ensembles de X) et ε un événement non observable.

La notion d'automate non déterministe est intéressante pour modéliser certains problèmes et en particulier celui qui nous intéresse : la présence d'un événement non observable (potentiellement associé à différents arcs du graphe).

4.4 Automate observateur

On appelle automate observateur $\mathcal{A}_{obs}$ d'un automate non déterministe $\mathcal{A}_{nd}$, l'automate déterministe générant et marquant les mêmes langages.

L'automate observateur est déterministe et nous avons :

$$\mathcal{L}(\mathcal{A}_{obs}) = \mathcal{L}(\mathcal{A}_{nd}) \tag{4.2}$$

Cassandras et Lafortune [Cassandras et Lafortune, 1999] ont montré l'existence de cet automate et donné un algorithme pour le construire. Cet observateur est un automate dont les états sont des sous ensembles de l'ensemble des états de l'automate non déterministe et dont les transitions ne sont associées qu'à des événements observables. Si l'automate non déterministe est fini, l'automate déterministe équivalent l'est aussi.

Revenons sur la notion d'événements non observables (absence de capteur, non transmission d'information, panne…). Plutôt que d'étiqueter les transitions avec ε et de considérer un automate non déterministe, considérons plutôt deux classes d'événements : $\mathcal{E}_o$ événements observables et $\mathcal{E}_{no}$ événements non observable et que l'automate est déterministe avec $\mathcal{E}$ partitionné en ces deux classes d'événements. L'algorithme cité précédemment peut alors être appliqué à la construction de l'observateur de cet automate en traitant tous les événements non observables comme l'a été précédemment ε. Ces algorithmes sont implémentés dans le logiciel DESUMA.

4.5 Diagnostic

Dans de nombreuses applications où le modèle du système contient des événements non observables, il peut être intéressant de déterminer si certains d'entre eux ont pu intervenir ou sont intervenus avec certitude lorsque le système se trouve dans un état d'observation donné. C'est le problème du diagnostic. Pour le réaliser, on construit un automate dit de diagnostic en modifiant l'algorithme de construction de l'automate observateur afin d'affecter, dans chaque sous ensemble d'états obtenu, un statut à tous les états qui le constituent. Ce statut indique si, dans le chemin d'accès à cet état, l'événement non observable à diagnostiquer est présent. Si tous les états d'un ensemble ont acquis ce statut, on peut affirmer que l'événement s'est produit.

4.6 Contrôle par supervision

Pour modifier le comportement d'un SED noté $\mathcal{G}$, l'observateur noté $\mathcal{S}$ observe les événements de G (si possible tous mais par forcément). $\mathcal{S}$ indique à $\mathcal{G}$ quels sont les événements qui seront autorisés parmi l'ensemble des événements actifs courant de G. Donc, la décision est prise à l'observation par $\mathcal{S}$ des événements de $\mathcal{G}$. Le rôle du superviseur consiste alors à valider ou autoriser (ou invalider ou interdire) l'occurrence d'événements dans un système à événement discrets. Il ne peut en aucun cas forcer des événements à se produire. Le superviseur ne peut que restreindre le fonctionnement du système à événements discrets.

L'action de contrôle ne peut donc changer qu'après occurrence d'un événement observable ; elle est instantanément mise à jour. Il faut préciser que cette mise à jour intervient avant occurrence de tout événement non observable ; c'est un point important car il est possible que l'action de mise à jour porte sur des événements non observables. En

effet, aucune hypothèse de lien éventuel entre observabilité et contrôlabilité d'un événement n'est faite. Un événement non observable peut être contrôlable, un événement non contrôlable peut être observable et ainsi de suite.

Définissons $\mathcal{S}$, un superviseur devant agir sur $\mathcal{G}$ en « contre réaction » comme le montre la figure 4.1. Soient :

- $\mathcal{E}_c$ le sous ensemble des événements contrôlables (ceux dont $\mathcal{S}$ peut empêcher l'occurrence)

- $\mathcal{E}_{nc}$ le sous ensemble des événements non contrôlables.

Ils constituent une partition de E :

$$\mathcal{E}_c \cup \mathcal{E}_{nc} = \mathcal{E} \tag{4.3}$$

Supposons que tous les événements de $\mathcal{G}$ sont observés par $\mathcal{S}$ et que s représente la suite des occurrences des événements « traités » par $\mathcal{G}$ au fil du temps.

La fonction de transition de $\mathcal{G}$ peut être contrôlée par $\mathcal{S}$ en ce sens que $\mathcal{S}$ peut valider ou invalider tous les événements contrôlables de $\mathcal{G}$.

Formellement, $\mathcal{S}$ est une fonction du langage généré par $\mathcal{G}$ dans l'ensemble puissance de $\mathcal{E}$ (ensemble des sous ensembles de $\mathcal{E}$ ou ensemble des parties de $\mathcal{E}$).

$$\mathcal{S} : \mathcal{L}(\mathcal{G}) \rightarrow 2^{\mathcal{E}} \tag{4.4}$$

Pour chaque suite $s \in \mathcal{L}(\mathcal{G})$ traitée par $\mathcal{G}$, sous le contrôle de $\mathcal{S}$, $\mathcal{S}(s) \cap \Gamma(f(x_0,s))$ est l'ensemble des événements validés que $\mathcal{G}$ peut traiter dans sont état courant $f(x_0,s)$: $\mathcal{G}$ ne peut traiter un événement de $(f(x_0,s))$ si cet événement n'est pas aussi contenu dans $\mathcal{S}(s)$.

On dit qu'un superviseur S est admissible si :

$$\mathcal{E}_{nc} \cap \Gamma(f(x_0,s)) \subseteq \mathcal{S}(s) \tag{4.5}$$

$\mathcal{S}$ n'est pas autorisé à invalider un événement non contrôlable. $\mathcal{S}(s)$ est l'action de contrôle à s. $\mathcal{S}$ est la politique de contrôle.

Nous volons préciser au moins trois aspects : d'abord, que le domaine de définition de $\mathcal{S}$ est $\mathcal{L}(\mathcal{G})$ et non $\mathcal{X}$ ensemble des états (différence avec le contrôle des systèmes continus) ; en plus, l'action de contrôle peut aussi changer d'un passage à un autre dans le même état et finalement, on ne parle pas ici de la manière d'implanter $\mathcal{S}$.

4.7 Approche proposée

4.7.1 Système de contrôle par supervision

Dans la mesure où il peut modifier l'accès aux états dangereux d'un système, le contrôle par supervision fait partie des moyens de la sûreté de fonctionnement. Néanmoins, comme tous les autres, son efficacité doit être évaluée avant toute mise en œuvre. On doit donc pour cela construire le modèle adéquat. En contexte de fiabilité dynamique, nous devons encore procéder par simulation. Nous avons donc cherché comme dans les chapitres précédents à construire les outils nécessaires dans l'environnement Scicos Scilab. Le système dynamique hybride et son contrôle par supervision sont décrits par le schéma de la figure 4.2. Dans l'automate stochastique hybride représentant le système à contrôler, le bloc automate (défini au chapitre 2) a été modifié. Nous lui avons ajouté une entrée spécifique placée en dessous des autres entrées à gauche du bloc (rappel : dans le bloc ASH initial, il y a autant d'entrées que d'états discrets du système, chacune porte la dynamique continue, la réinitialisation et la condition de garde). Par cette entrée, le superviseur indiquera à tout moment la liste des événements à invalider afin que le comportement de la partie discrète du système soit modifié. La figure 4.3 montre le modèle Scicos du système de contrôle correspondant à la figure 4.2. Dans ce modèle nous n'avons pas fait figurer le générateur aléatoire pour ne pas surcharger la figure.

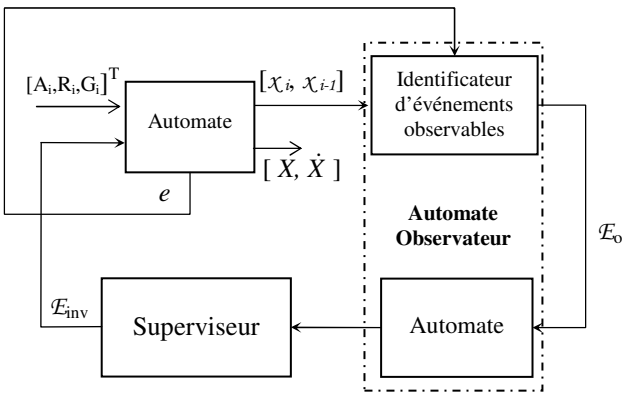


Figure 4.2 – Système dynamique hybride et son contrôle par supervision.

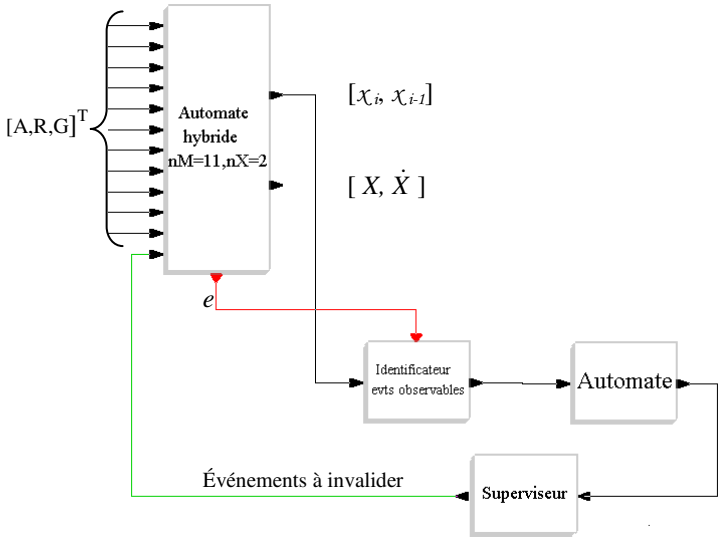


Figure 4.3 – Modèle Scicos du système de contrôle par supervision.

Le fonctionnement du système de contrôle par supervision est simple. Le système initial est modélisé par l'automate stochastique hybride. Dans cet ASH, le bloc automate gère le séquencement des événements liés aussi bien à la partie continue (franchissement de seuils) qu'à la partie discrète (défaillance, réparations…). Le bloc automate observateur est constitué de deux blocs, l'identificateur d'événements observables et le modèle automate du système observé. Le premier identifie les événements de l'ASH à partir de la connaissance de l'état précédent et de l'état courant à chaque occurrence de *e* événement composite de tous les événements de l'ASH et ensuite, ignore les événements déclarés non observables (initialisation). Le second bloc noté automate est le modèle observable du système qui évolue au fur et à mesure de l'occurrence des événements observables. Il fournit sur sa sortie le numéro de l'état d'observation. A partir de cette information le Superviseur détermine les événements contrôlables à invalider qu'il transmet au bloc automate afin de modifier le comportement du système.

Il est possible bien entendu de remplacer l'automate observateur par un automate de diagnostic d'un événement non observable pour conditionner la liste des événements contrôlables à valider.

4.7.2 Cas test

Pour illustrer l'action du système de contrôle par supervision sur un système dynamique nous reprenons le système du contrôle de la température d'un four par les contrôleurs PI et TOR défini au §3.2.1.2. Considérons maintenant que les capteurs de détection de dépassement de température consécutif à une défaillance PI ou TOR soient susceptibles de défaillance. Dans les états 2, 5, 6 et 9, on ajoute alors des transitions de

sortie représentant ces défaillances vers un nouvel état (n° 10) représentant une situation de danger comme le montre la figure 4.4.

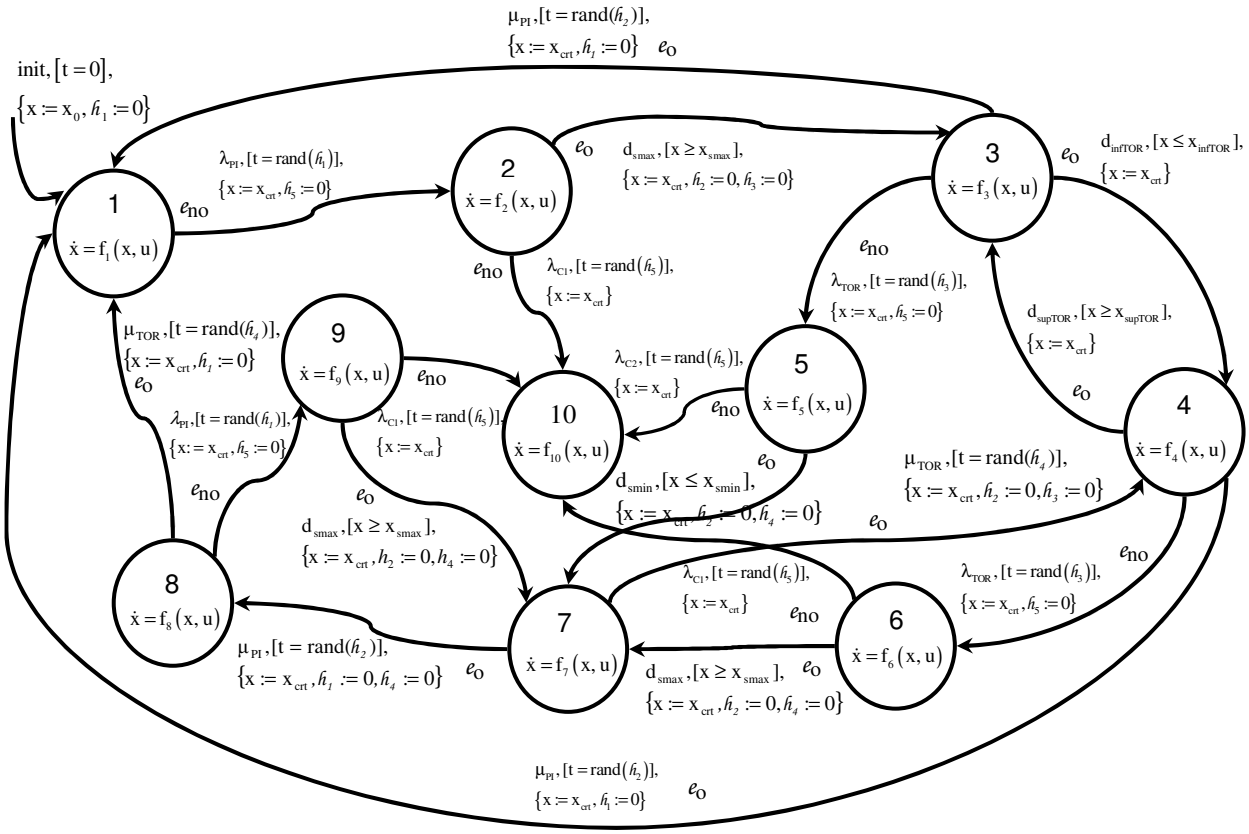


Figure 4.4 – Automate stochastique hybride du système de contrôle de la température du four avec défaillance des capteurs.

Soit $\mathcal{E}$ l'ensemble d'événements du système dynamique :

$$\mathcal{E} = \{\lambda_{PI}, \lambda_{TOR}, \lambda_{c1}, \lambda_{c2}\, \mu_{PI}, \mu_{TOR}, d_{smin}, d_{smax}, d_{infTOR}, d_{supTOR}\}$$

Soit $\mathcal{E}_o$, ensemble d'événements observables du système contient les franchissements des seuils et les réparations des composants PI et TOR:

$$\mathcal{E}_o = \{\mu_{PI}, \mu_{TOR}, d_{smin}, d_{smax}, d_{infTOR}, d_{supTOR}\}$$

et $\mathcal{E}_{no}$ l'ensemble d'événements non observables :

$$\mathcal{E}_{no} = \{\lambda_{PI}, \lambda_{TOR}, \lambda_{c1},\ \lambda_{c2}\}$$

La simulation de ce modèle (10 000 histoires sur une durée de 500 000h) nous permet de calculer la probabilité cumulée d'accès à l'état de danger comme le montre la figure 4.5. Le temps de simulation est d'environ 7mn sous Windows et 3mn sous Linux.

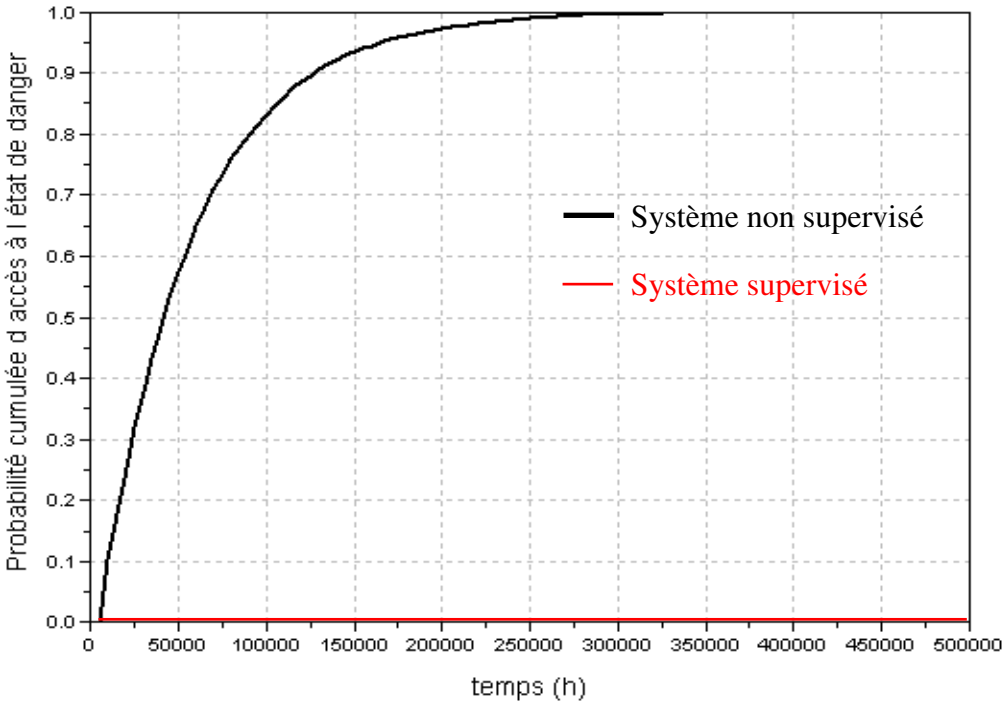


Figure 4.5 – Probabilité cumulée d'accès à l'état de danger.

Nous allons construire le superviseur qui empêche l'accès à cet état de danger.

Dans un premier temps, construisons l'automate observateur de l'automate de la figure 4.4. Obtenu à partir du logiciel DESUMA, il est représenté sur la figure 4.6.

Dans cet automate, les états **1, 2, 4** et **5** de l'automate observateur contiennent les états 2, 5, 6 ou 9 de l'automate observé. Lorsque ces états sont observés, le superviseur doit interdire les événements de défaillance des capteurs. Cela suppose qu'ils soient contrôlables. Dans la pratique cela peut être rendu possible par exemple en ajoutant d'autres capteurs en redondance ou d'autres moyens de détection. Déclarons donc ces événements contrôlables. Les autres événements peuvent indifféremment être déclarés contrôlables ou non puisque de toutes façons, on ne demandera pas au superviseur de les invalider.

Le modèle Scicos complet de ce cas test supervisé est présenté sur la figure 4.7. Nous pouvons distinguer l'automate stochastique hybride du système ainsi que l'identificateur d'événements observables, l'automate et le superviseur.

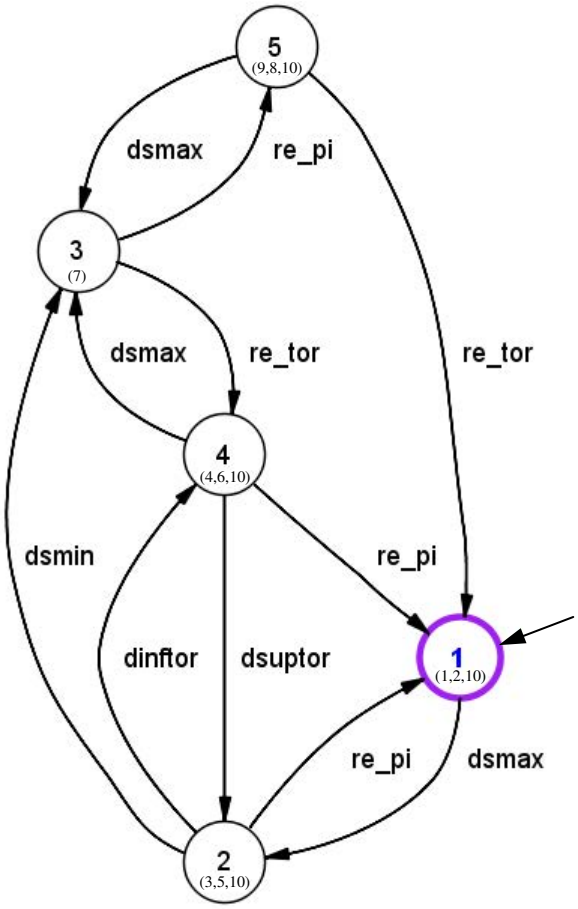


Figure 4.6 – Automate observateur du système.
(entre parenthèses : les états de l'automate stochastique hybride de la figure 4.7)

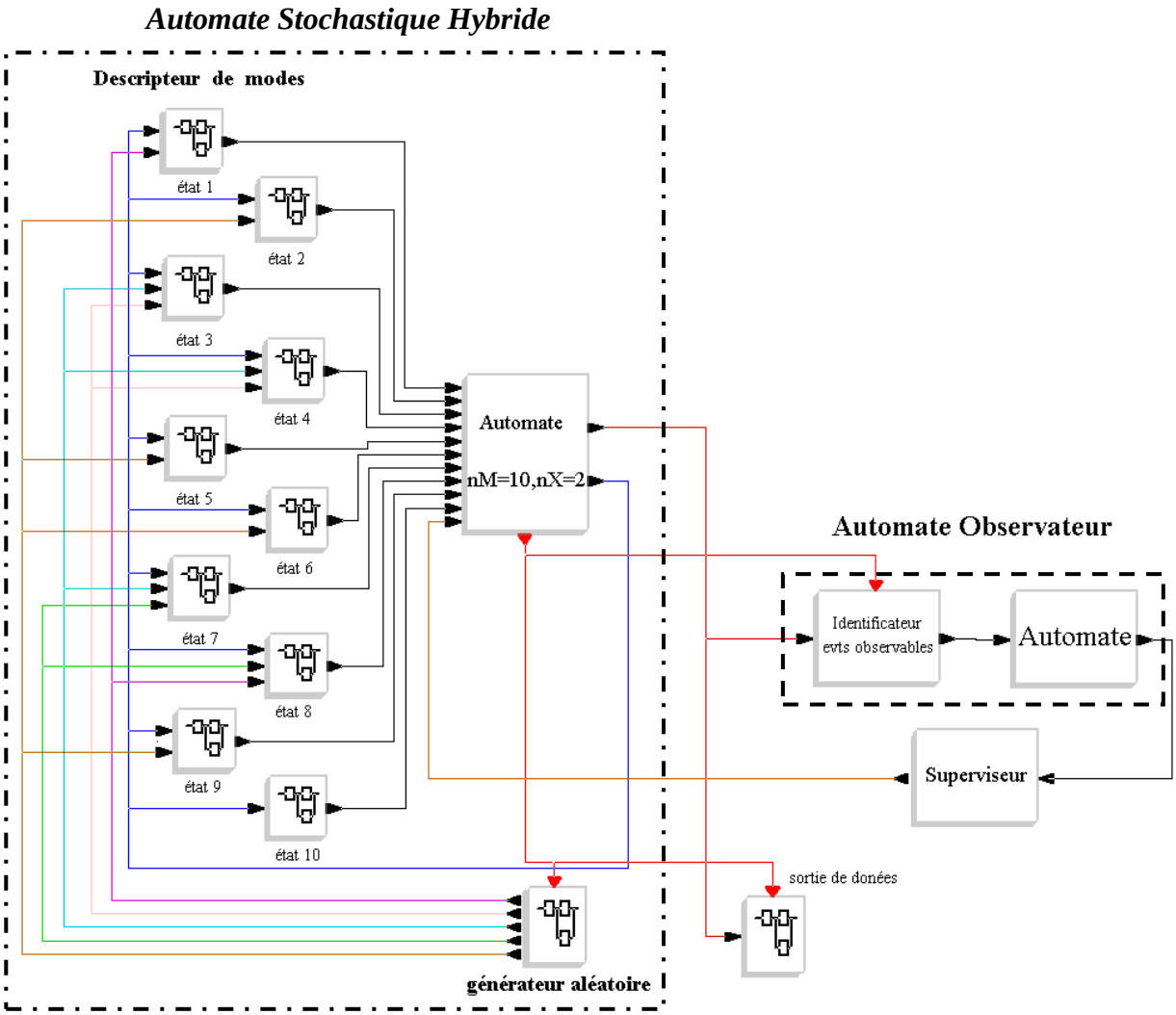


Figure 4.7 – modèle Scicos du système contrôlé par supervision.

La simulation de ce modèle sur 500000 d'heures montre bien entendu que la probabilité d'accès à l'état de danger est rendue nulle.

L'intérêt de l'approche est de permettre de vérifier formellement ce résultat sans avoir à reconstruire l'automate complet du système modifié, par combinaison de l'automate initial avec l'automate modélisant les modifications du système (les capteurs et ou détecteurs additionnels nécessaires) et la stratégie de contrôle (redondance active, passive…).

A titre documentaire, nous présentons (figures 4.8 à 4.10) les fenêtres Scicos relatives à la définition des blocs constituant le contrôle par supervision que nous avons définis.

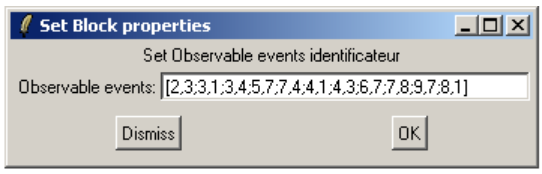


Figure 4.8 – Interface du bloc Identificateur des événements observables.

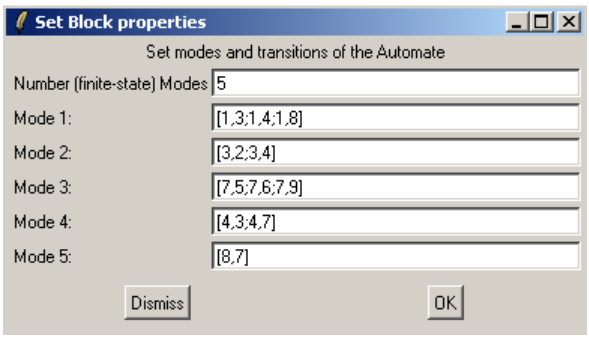


Figure 4.9 – Interface du bloc Automate.

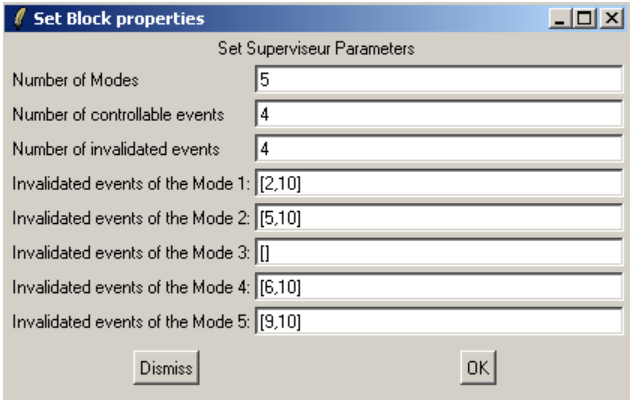


Figure 4.10 – Interface du bloc Superviseur.

4.8 Conclusion

Nous avons donné dans ce court chapitre une idée de l'intérêt du contrôle par supervision comme moyen de la sûreté de fonctionnement. Pour rendre cet intérêt encore plus évident, il faudrait pouvoir manipuler dans un même environnement les modèles pour la simulation développés avec Scicos Scilab et ceux développés pour la construction de l'automate observateur avec DESUMA.

Nous avons considéré dans notre exemple que les événements contrôlables l'étaient complètement. En effet, la théorie de la supervision utilisée ici suppose l'invalidation totale d'un événement contrôlable. En pratique, les éléments matériels mis en œuvre pour réaliser ce contrôle sont eux-mêmes sujets à défaillance. Le rôle du superviseur serait alors d'agir sur les événements contrôlables en réduisant leur probabilité d'occurrence et non en les empêchant totalement. Nous entrons alors dans le domaine de la supervision probabiliste des SED dont l'application à la sûreté de fonctionnement est encore à construire.

Conclusion générale

Dans notre premier chapitre nous avons montré, à travers une présentation de l'état de l'art, que les approches tendant à résoudre analytiquement les problèmes de la fiabilité dynamique s'appliquent dans des contextes restrictifs (cas simples) en invoquant un outillage mathématique lourd. La simulation du comportement du système nous est apparue comme étant le seul moyen accessible d'une approche complète de l'évaluation probabiliste. Pour cela, nous devions établir un modèle simulable du système de type état transition, capable de résoudre les équations différentielles associées aux états. Nous n'avons pas retenu les réseaux de Petri comme modèle car nous n'avons pas trouvé d'outil logiciel intégrant dans un même environnement l'exécution interactive d'un RdP et d'un puissant solveur numérique d'équations différentielles. Cela nous a orientés vers le concept d'automate stochastique hybride.

Dans le chapitre 2, nous avons défini ce concept d'ASH capable de prendre en compte tous les problèmes posés par la fiabilité dynamique et d'accéder à l'évaluation des grandeurs de la sûreté de fonctionnement par une simulation de Monte Carlo. Une première évaluation de ce concept dans l'environnement Scilab Scicos nous a permis d'en démontrer la faisabilité sur le cas test du système de contrôle de la température d'un four au moyen de régulateurs en redondance passive hétérogène.

Dans le chapitre 3, nous avons montré l'efficacité de notre approche de simulation pour l'évaluation de la sûreté de fonctionnement en contexte dynamique. Elle permet de répondre aux problèmes posés, notamment celui de la prise en compte de l'influence de l'état discret, de l'état continu et de leur interaction dans l'évaluation probabiliste des performances d'un système dans lequel en outre, les caractéristiques fiabilistes des composants dépendent eux-mêmes des états continu et discret. Nous avons vu qu'il reste néanmoins un effort à faire pour accélérer la simulation de Monte Carlo en recherchant l'optimisation des algorithmes de calcul numériques et l'exploitation des différentes échelles de temps entre aspects fonctionnels et dysfonctionnels. La construction d'un

module ASH directement exécutable sous Scilab permettrait aussi d'accélérer la simulation de Monte Carlo.

Du point de vue de la mise en œuvre, il sera intéressant d'intégrer dans l'environnement Scilab Scicos les algorithmes de construction des automates afin de libérer l'analyste du souci de la construction exacte et précise du modèle complet. Cependant, il conviendra bien entendu de théoriser cette approche dans le cadre hybride et stochastique. C'est un aspect important qui pourra être développé dans le cadre de travaux futurs au sein de l'équipe de recherche.

Dans le chapitre 4, nous avons cherché à montrer que d'autres aspects peuvent également être pris en compte dans notre approche de la simulation pour la fiabilité dynamique. Nous pouvons citer par exemple les caractéristiques probabilistes du diagnostic et l'impact du contrôle par supervision sur la sûreté de fonctionnement. Nous avons tenté de donner dans ce chapitre une idée de l'intérêt du contrôle par supervision comme moyen de la sûreté de fonctionnement. Les concepts d'automate observateur, et de contrôleur ont été introduits et illustrés sur notre cas test afin de montrer leur potentialité. Nous avons considéré dans notre exemple que les événements contrôlables étaient contrôlés de manière déterministe, la théorie de la supervision appliquée supposant l'invalidation totale d'un événement contrôlable. Bien entendu dans la pratique, la mise en œuvre de ce contrôle exige l'introduction d'éléments matériels caractérisés par leur propre fiabilité. Le rôle du superviseur est alors d'agir sur les événements contrôlables en réduisant leur probabilité d'occurrence et non en les empêchant totalement. Il est donc nécessaire de développer cet aspect de la supervision probabiliste des SED pour l'application à la sûreté de fonctionnement.

Enfin, sur ce dernier point, on peut reprendre la nécessité déjà évoquée de disposer d'un environnement intégrant les modèles pour la simulation développés avec Scicos Scilab et ceux développés pour la construction des observateurs et des contrôleurs avec DESUMA.

Bibliographie

[Acosta et Siu, 1993] Acosta C., Siu N. Dynamic event trees in accident sequence analysis: application to steam generator tube rupture. *Reliability Engineering and System Safety* 41, 1993, p. 135-154.

[Aldemir, 1987] Aldemir T. Computer-Assisted Markov Failure Modeling of Process Control Systems. *IEEE Transactions on reliability*, vol. R-36, No. 1, 1987 April.

[Arnold, 1992] Arnold A. Systèmes de transitions et sémantique des processus communicants. *Masson*, editor, 1992.

[Alur *et al.,* 1993] Alur, R., Courcoubetis C., Henzinger T. A., Ho P. H. *Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems*. In Grossman R. L., Nerode A., Ravn A. P., Rischel H., editors, Hybrid Systems I, Lecture Notes in Computer Science 736, p. 209 – 229. Spinger-Verlag, 1993.

[Alur *et al.,* 1995] Alur, R., Courcoubetis C., Halbwachs N., Henzinger T. A., Ho P. H., Nicollin X., Olivero A., Sifakis J., Yovine S. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138; pp. 3 – 34, 1995.

[Alur *et al.,* 1997] Alur R., Henzinger T. A., Wong-Toi H. Symbolic analysis of hybride systems. *36th Annual IEEE Conference on Decision and Control* 5CDC 1997, pp. 702-707.

[Alla *et al.,* 2000] Alla A., David R., Di Mascolo M., Ferrier J.-L. *Analyse et commande des systèmes à événements discrets*. Hermès Ed., 2000.

[Becker *et al.,* 2002] Becker G., Camarinopoulos L., Kabranis D. Dynamic reliability under random shocks. *Reliability Engineering System Safety*. 77, 2002, p. 239 – 251.

[Belhadj et Aldemir, 1996] Belhadj M., Aldemir T. Some computational improvements in process system reliability and safety analysis using dynamic methodologies. *Reliability Engineering System Safety* 52, 1996, p. 339 – 347.

[Borgonovo *et al.,* 2000] Borgonovo E., Marseguerra M., Zio E. A Monte Carlo methodological approach to plant availability modelling with maintenance, aging and obsolescence. *Reliability Engineering and System Safety* 67, 2000, p. 61 – 73.

[Brams, 1983] Brams G. W. *Réseaux de Petri : théorie et pratique*. Masson, 1983.

[Brinzei, 2003] Brinzei N. *Contribution à la modélisation et à la commande modulaire des systèmes de production, par une approche UML et réseaux de Pétri*. Thèse de doctorat présentée à l'Université de Technologie de Belfort Montbéliard, 19 septembre 2003.

[Cabarbaye et Laulheret, 2005] Cabarbaye A., Laulheret R. Evaluation de la sûreté de fonctionnement des systèmes dynamiques par modélisation récursive. *6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005*, Bordeaux, 2005.

[Campbell *et al.,* 2006] Campbell S. L., Chancelier J.-P., Nikoukhah R. *Modeling and simulation in Scilab/Scicos*. Springer, 2006.

[Cassandras et Lafortune, 1999] Cassandras C. G., Lafortune S. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.

[Chabot *et al.,* 1998] Chabot J. L., Ducamp F., Mattei J.-M., Dutuit Y., Hutinet T. Simulation hybride méthode de modélisation intégrant phénomènes continus et discrets. *11eme Colloque National de Fiabilité et Maintenabilité* - Lambda-mu 11, Arcachon, 1998.

[Chevalier *et al.,* 2004] Chevalier M., Garnier R., Chang P., Lusson B. *La Sûreté de Fonctionnement*. Intersections, le magazine Schneider Electric de l'enseignement technologique et professionnel. Novembre 2004.

[CEI 50 (191] CEI 50 191 *Vocabulaire Electrotechnique International*, Chapitre 191 – Sûreté de fonctionnement et qualité des services – 1990.

[CEI 61069] CEI 61069 *Norme international – Mesure et commande dans les processus industriel – Appréciation des propriétés d'un système en vue de son évaluation* – Parti 5 Évaluation de la sûreté de fonctionnement d'un système.

[Cipoire, 2006] Cipoire Y. Estimation d'une loi de Weibull pour des données de défaillance en âge et en distance. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*, 2006.

[Clarotti *et al.,* 2004] Clarotti C., Lannoy A., Odin S., Procaccia H., Detection of equipment aging and determination of the efficiency of a corrective measure. *Reliability engineering & system safety*, 84, 2004, p. 57 – 64.

[Corazza, 1975] Corazza M. *Techniques mathématiques de la fiabilité prévisionnelle*. Cepadues Éditions, 1975.

[Cocozza-Thivent, 1997] Cocozza-Thivent C. *Processus stochastiques et fiabilité des systèmes*. Mathématiques & Applications 28. Springer, 1997.

[Cocozza-Thivent et Eymard, 2006a] Cocozza-Thivent C., Eymard R. Algorithmes de fiabilité dynamique. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*. Lille, 2006.

[Cocozza-Thivent *et al.,* 2006b] Cocozza-Thivent C., Desgrouas M., Mercier S. Algorithme de calcul de disponibilité asymptotique en fiabilité dynamique. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*. Lille, 2006.

[Cojazzi et Cacciabue, 1993] Cojazzi G., Cacciabue P. C. The DYLAM approach for the reliability analysis of systems with dynamic interactions. EUR. 15266 *Report, JCR Ispra*, 1993.

[Cojazzi et Cacciabue, 1996] Cojazzi G., Cacciabue P. C. The DYLAM approach for the reliability analysis of systems with dynamic interactions. *Reliability Engineering and Systems Safety* 52, 1996, p. 279-296.

[David et Alla, 1989] David R., Alla H. *Du grafcet aux réseau de Petri*. Paris. Hermès, 1989.

[Dejean *et al.,* 2005] Dejean J. P., Averbuch D., Gainville M., Doux F. Développement de modèles pour la gestion intégrée des risques pour les systèmes de production pétroliers offshore. Utilisation de réseaux de Petri Stochastiques interprétés. *6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005*, Bordeaux, 2005.

[Desgrouas et Mercier, 2005] Desgrouas M., Mercier S. Comportement asymptotique d'un exemple de fiabilité dynamique. *6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005*, Bordeaux, 2005.

[DESUMA] http://www.eecs.umich.edu/umdes/toolboxes.html.

[Devooght et Smidts, 1992a] Devooght J., Smidts C. Probabilistic reactor dynamics – I: The theory of continuous event trees. *Nuclear science and engineering*, 111, 1992, p. 229 – 240.

[Devooght, 1997] Devooght J. *Advances in nuclear science and technology*. Volumen 25. Edited by Lewis J., Becker M. Plenum Press. New York, 1997.

[Devooght et Labeau, 1995] Devooght J., Labeau P. E. Moments of the distributions in probabilistic dynamics. *Annals of Nuclear Energy* 22 (2), 1995, p. 97-108.

[Dubi, 2000] Dubi A. *Monte Carlo Applications in systems engineering*. John Wiley & Sons, Ltd. England, 2000.

[Dufor et Dutuit, 2002] Dufour F., Dutuit Y. Dynamic Reliability: A new model, Fiabilité dynamique: un nouveau modèle. *Lambda mu 13–ESREL, European Conférence*, 2002, p. 350–353.

[Dutuit *et al.,* 1997] Dutuit Y., Rauzy A., Signoret J.-P., Thomas P. Dependability modelling and evaluation by using stochastic Petri nets: application to two test cases. *Reliability Engineering and System Safety* 55, 1997, p. 117-124.

[EN 292 – 1, 1991] *Sécurité de machines – Notions fondamentales, principes, généraux de conception* – Partie 1 : Terminologie de base – Méthodologie – 1991.

[German, 2000] German R. *Performance analysis of Communication systems: modeling with non – markovian stochastic Petri nets*. John Wiley & Sons Ltd., 2000.

[Henzinger *et al.,* 1995] Henzinger T. A., Ho P. H., Wong-Toi H. Algorithmic analysis of nonlinear hybrid systems. *Proceedings of the 7th International Conference on computer aided verification*. P. Wolper, Ed., vol. 939. Liege, Belgium: Springer Verlag, 1995, pp. 225 – 238.

[Henzinger, 1996] Henzinger, T. A. The theory of hybrid automata. *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pp. 278 – 292, 1996.

[Humbert, 2008] Humbert S. *Déclination d'exigences de sécurité, du niveau système vers le niveau logiciel, assistée par des modèles formels*. Thèse de doctorat présentée à l'Université Bordeaux I, le 11 avril 2008.

[Huseh et Mosleh, 1996] Hsueh K. S., Mosleh A. The development and application of the accident dynamic simulator for probabilistic risk assessment of nuclear power plants. *Reliability Engineering and System Safety* 52, 1996, p. 297-314.

[Incera, 2001] Incera Dieguez J. A. *Contributions à la modélisation et à la simulation accélérée de réseaux de communication*. Thèse Informatique. Université de Rennes I. 2001.

[Grelier *et al.,* 2006] Grellier J. M., Oddi M., Simon G. Maîtrise du vieillissement et de l'obsolescence des matériels électroniques à EDF. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*, 2006.

[Johansson, 2000] Johansson K. L. *Hybrid systems cours*. Lectures. Berkeley University.

[Kermisch et Labeau, 2000a] Kermisch C., Labeau P. E. *Approche dynamique de la fiabilité des systèmes*. Project 6/2000 de l'ISdF. Tâche n° 1 : établissement de l'état de l'art en fiabilité dynamique. Service de Métrologie Nucléaire. Université Libre de Bruxelles, 2000.

[Kermisch et Labeau, 2000b] Kermisch C., Labeau P. E. *Approche dynamique de la fiabilité des systèmes*. Project 6/2000 de l'ISdF. Tâche n° 2 : choix de méthodes d'analyse. Service de Métrologie Nucléaire. Université Libre de Bruxelles, 2000.

[Kaufmman *et al.,* 1975] Kaufmman A., Gronchko G., Cruon R. *Modèles mathématiques pour l'étude de fiabilité des systèmes*. Masson et Cie.

[Labeau et Devooght, 1995] Labeau P. E., Devooght J. Synthesis of multivariate distributions from their moments for probabilistic dynamics. *Annals of Nuclear Energy* 22 (2), 1995, p. 109-124.

[Labeau *et al.,* 2000] Labeau P. E., Smidts C., Swaminathan S. Dynamique reliability: towards an integrated plarform for probabilistic risk assessment. *Reliability Engineering and System Safety, Elsevier.* 68, 2000, p. 219 – 254.

[Labeau et Izquierdo, 2005] Labeau P. E., Izquierdo J. M. The stimulus-driven theory of Petri Nets. *6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005*, Bordeaux, 2005.

[Labeau et Kermisch, 2001] Labeau P. E., Kermisch C. *Approche dynamique de la fiabilité dynamique des systèmes*. Rapport MNFD 2002-10, Service de Métrologie Nucléaire, Université Libre de Bruxelles, novembre 2002.

[Lannoy et Procaccia, 1994] Lannoy A., Procaccia H. *Méthodes avancées d'analyse des bases de données du retour d'expérience industriel*. Editions Eyrolles, 1994.

[Lannoy, 1996] Lannoy A. *Analyse quantitative et utilité du retour d'expérience pour la maintenance des matériels et la sécurité*. Edition Eyrolles. 1996.

[Laprie *et al.,* 1995] Laprie J.-C., Arlat J., Blanquart J.-P., Costes A., Crouzet Y., Deswarte Y., Fabre J.-C., Guillermain H., Kaâniche M., Kanoun K., Mazet C., Powell D., Rabéjac C., Thévenod P. *Guide de la Sûreté de fonctionnement*. Cépaduès Éditions. 1995.

[Laprie, 2004] Laprie J.-C. Sûreté de fonctionnement informatique : concepts, défis, directions. CNRS, LAAS. *ACI Sécurité et Informatique* – Toulouse, 15 – 17 novembre 2004.

[Levy, 1954] Levy P. Processus semi-markoviens. *Proc. Int. Cong. Math.* (Amsterdam), pp. 416-426.

[Lewins et Becker, 1997] Lewins J., Becker M. *Advances in nuclear science and technology*. Plenum Press, New York, 1997.

[Marseguerra et Zio, 1995] Marseguerra M., Zio E. The cell-to-boundary method in Monte Carlo-based dynamic PSA. *Reliability Engineering and System Safety* 48, 1995, p. 199-204.

[Marseguerra et Zio, 1996] Marseguerra M., Zio E. Monte Carlo approach to PSA for dynamic process systems. *Reliability Engineering and System Safety* 52, 1996, p. 227-241, 1996.

[Marseguerra *et al.,* 1998] Marseguerra M., Zio E., Devooght J., Labeau P. E. A concept paper on Dynamic Reliability via Monte Carlo Simulation. *Mathematics and Computers in Simulation* 47, 1998, p. 371-382.

[Medjoudj, 2004] Medjoudj, M. Extraction des scénarios critiques pour l'évaluation de la sûreté de fonctionnement des systèmes mécatroniques. *5ème Congrès des Doctorants, Ecole Doctorale Systèmes (EDSYS)*, Toulouse, 2004.

[Mercier, 2006] Mercier S. Encadrement de quantités fiabilistes pour un gros système markovien. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*. Lille, 2006.

[MIL-HDBK-217F, 1991] *Military Handbook Reliability Prediction of Electronic Equipment*. Department of Defense, Washington DC 20301, 1991.

[Murata, 1989] Murata T. Petri nets : properties, analysis and applications. *Proc. IEEE*, vol. 77, n° 4, p. 541-580, avril 1989.

[Najafi, 2005] Najafi M. *Numerical solver for hybrid Differential-Algebraic Equations (DAE) (Solveur Numérique pour les Systèmes d'équation Algébro-Différentiels hybrides)*. Ph.D. Thesis, Paris XII (Val de Marne) University, November 2005.

[Najafi et Nikoukhah, 2007] Najafi, M., Nikoukhah R. Modeling Hybrid Automata in Scicos. *Multi-conference on Systems and Control (MSC)*, Singapore, 1 – 3 October, 2007.

[Niel et Craye, 2002] Niel E., Craye E. *Maîtrise des risques et sûreté de fonctionnement des systèmes de production. Productique : information, commande, communication*. Lavoisier, 2002.

[Osaki, 2002] Osaki S. *Stochastic models in reliability and maintainability*. Springer, 2002

[Pagès et Gondran, 1980] Pagès A., Gondran M. *Fiabilité des systèmes*. Edition Eyrolles. 1980.

[Pérez *et al.,* 2007a] Pérez Castaneda G. A., Aubry J-F., Brinzei N. Modélisation et simulation d'un système dynamique hybride pour calculer sa fiabilité dynamique en utilisant le toolbox Scicos de Scilab. *7ème édition du Congrès International pluridisciplinaire Qualita 2007* – Tanger (Maroc), p. 311 – 318, 2007.

[Pérez *et al.,* 2007b] Pérez Castaneda G. A., Aubry J-F., Brinzei N. Etat de l'art en fiabilité dynamique. *2èmes Journées Doctorales du GDR MACS JDMACS*, Reims, France, 2007.

[Pérez *et al.,* 2008a] Pérez Castaneda G. A., Aubry J. F., Brinzei N. Automate stochastique hybride appliqué à la fiabilité dynamique. *7ème Conférence International de Modélisation et Simulation MOSIM 2008*. Du 31 mars au 2 avril Paris, France, 2008.

[Pérez *et al.,* 2008b] Pérez Castaneda G. A., Marzat J., Aubry J. F., Brinzei N. *Simulation de Monte Carlo prenant en compte le vieillissement des composants. Lambda mu 16,* Avignon 2008.

[Ramadge et Wonham, 1987] Ramadge P. J., Wonham W. M. Supervisory control of a class of discret event processes. *SIAM Journal on Control, and Optimisation*, vol. 25, n° 25, p. 206 – 230, 1987.

[Sadou, 2007] Sadou N. *Aide à la conception des systèmes embarqués sûrs de fonctionnement.* Thèse de Doctorat du Laboratoire d'analyse et d'Architecture des systèmes du CNRS , 2007.

[Sadou *et al.,* 2006] Sadou N., Demmou H., Pascal J. C., Valette R. Fiabilité dynamique des systèmes hybrides : approche basée scénarios. *Conférence International Francophone d'Automatique (CIFA 2006)*, Bordeaux, France, 30 mai – 1er juin 2006.

[Saleh et Marais, 1991] Saleh J. H., Marais K. Highlights from the early (and pre-) history of reliability engineering. *Reliability Engineering and System Safety* 91, 2006, p. 249-256.

[Schoenig, 2004] Schoenig R. *Définition d'une méthodologie de conception des systèmes mécatroniques surs de fonctionnement*. Thèse de doctorat présentée à l'Institut National Polytechnique de Lorraine, le 26 octobre 2004.

[Schoenig *et al.,* 2006] Schoenig R., Aubry J.-F., Cambois T., Hutinet T. An aggregation method of Markov graphs for the reliability analysis of hybrid systems. *International Journal on Reliability Engineering and System Safety RESS*, Elsevier, Vol 91/2, 2006, p. 137 – 148.

[Scicos, 1989] Scicos http://www.scicos.org.

[Scilab, 1989] Scilab http://www.scilab.org.

[Siu, 1994] Siu, N. Risk assessment for dynamic systems: An overview. *Reliability Engineering and System Safety* 43, 1994, p. 43 – 73.

[Smidts et Devooght, 1992] Smidts C., Devooght J. Probabilistic reactor dynamics – II: A Monte Carlo study of a fast reactor transient. *Nuclear science and engineering,* 111, 1992, p. 241 – 256.

[Smidts, 1994] Smidts C. Probabilistic dynamics: a comparison between continuous event trees and discrete event tree model. *Reliability Engineering System Safety* 4, 1994, p. 189 – 206.

[Smith, 2001] Smith D. J. *Reliability, maintainability and risk. Practical methods for engineers*. Sixth Edition. Butterworth Heinemann, 2001.

[Smith, 1955] Smith W. L. Regenerative stochastic Process. *Proc. Roy. Soc.* Série A., Vol. 232, pp. 6-31.

[Soro *et al.,* 2006] Soro W. I., Nourelfath M., Aït-Kadi D. Evaluation des indices de performance d'un système multi-etats degradable. *6ème Conf. Francophone de Modélisation et Simulation, MOSIM*, Rabat, Maroc, 2006, p. 184 – 193.

[Tchangani et Noyes, 2005] Tchangani A. P., Noyes D. Attempt to modeling dynamic reliability using dynamic Bayesian networks. *6ème Congrès International pluridisciplinaire, qualité et sûreté de fonctionnement, Qualita 2005*, Bordeaux, 2005.

[Tombuyses et Aldemir, 1997] Tombuyses B., Aldemir T. Computational efficiency of the continuous cell-to-cell mapping technique as a function of integration schemes. *Reliability Engineering and System Safety* 58, 1997, p. 215-223.

[Trivedi et Kulkarni, 1993] Trivedi K. S., Kulkarni V. G. FSPNs: Fluid Stochastic Petri Nets. *14th International Conference on Applications and Theory of Petri Nets*, 1993, p. 24-31.

[Veach, 1997] Veach E. *Robust Monte Carlo methods for light transport simulation*. A dissertation submitted to the department of computer science and the committee on graduate studies of Stanford University, for the degree of doctor of philosophy. 1997.

[Villemeur, 1988] Villemeur A. *Sûreté de fonctionnement des systèmes industriels*. Edition Eyrolles. 1988.

[Zaytoon, 2001] Zaytoon J. (sous la direction de). *Systèmes dynamiques Hybrides*. Hermes Science Eurpope Ltd, 2001.

[Zhang *et al.,* 2006] Zhang H., Dufour F., Innal F., Dutuit Y. Fiabilité dynamique et processus déterministe par morceaux. *Maîtrise de risques et sûreté de fonctionnement, lambda mu 15*. Lille, 2006.

[Zhang *et al.,* 2008] Zhang H., Gonzalez K., Dufour F., Dutuit Y. Piecewise deterministic Markov processes and dynamic reliability. *Proceeding of the Mechanical Engineers, Part 0, Journal of Risk and Reliability*. Volume 222, number 4/2008, Professional Engineering Publishing, ISSN 1748-006X (Print), 1748-0078 (Online), 2008.

Annexe

Codes des blocs Scicos

```
function [x,y,typ]=AUTOMAT(job,arg1,arg2)
  x=[];y=[];typ=[]
  select job
   case 'plot' then
    standard_draw(arg1)
   case 'getinputs' then
    [x,y,typ]=standard_inputs(arg1)
   case 'getoutputs' then
    [x,y,typ]=standard_outputs(arg1)
   case 'getorigin' then
    [x,y]=standard_origin(arg1)
   case 'set' then
    x=arg1;
    graphics=arg1.graphics;exprs=graphics.exprs
    model=arg1.model;ipar=model.ipar;
    NMode=ipar(1)
    NX=ipar(3)
    while %t do
     CX='C1';
     MSG0='"Jump from Mode '; MSG2=':[..;M_final(Guard=In(';MSG3=').i);..]"'
     MSG=MSG0+'1'+MSG2+'1'+MSG3;
     VEC='"mat",[-1,1]';
     for i=2:NMode
       CX=CX+','+'C'+string(i);
       MSG=MSG+';'+MSG0+string(i)+MSG2+string(i)+MSG3;
       VEC=VEC+','+'"mat",[-1,1]';
     end
     //=============================================
```

```
        GTV='[ok,NMode,Minitial,NX,X0,XP,'+CX+',exprs]=getvalue("Set Finite state
machine model",..
         ["Number (finite-state) Modes";"Initial Mode";"Number of continuous-time
states";"Continuous-time states intial values";"Xproperties of continuous-time states in
each Mode";'+MSG+'],..
          list("vec",1,"vec",1,"vec",1,"mat",[-1,-1],"mat",[-1,-1],'+VEC+'),exprs)'
        execstr(GTV); if ~ok then break,end
        NMode_old=size(exprs,'*')-5;//-number of fileds before CX
        ModifEncore=%f;

        if (NMode_old>NMode) then
          exprs(NMode+6:NMode_old+5)=[];// number of fileds
          ModifEncore=%t;
        end
        if (NMode_old<NMode) then
          exprs(NMode_old+6:NMode+5)=exprs(NMode_old+4);// number of fileds
          ModifEncore=%t;
        end
        if (NX<>size(X0,'*')) then
          x_message('the size of initial continuous-time states should be NX='+string(NX));
          ModifEncore=%t;
        end

        [rXP,cXP]=size(XP)
        if cXP<>NX then
          x_message('Xproperty matrix is not valid: it should have NX='+string(NX)+'
columns');
          ModifEncore=%t;
        elseif ((rXP<>NMode) & (rXP>1))
          x_message('Xproperty matrix is not valid: it should have
NMode='+string(NMode)+' or 1 row(s)');
          ModifEncore=%t;
        elseif (rXP==1)
          for i=1:NMode-1
              XP=[XP;XP(1,:)];// xproprties are identical in modes.
          end
        end

        if (NMode_old==NMode)&(~ModifEncore) then
          XP=matrix(XP',NMode*NX,1);// put XP in column vector to be stocked in ipar
          ipar=[NMode;Minitial;NX;XP];
          rpar=matrix(X0,NX,1);// put X0 in a column vector;
          INP=ones(NMode,1);
    //      EVT=ones(NMode-1,1);    //sorties d'activations, gapercas, 24/01/2007
    //          if NX>0 then OUT=[2;2*NX;1;1];else OUT=[4];end          // deux sorties
ajoutÃ©es, gapercas, 22/01/2007
```

```
            if NX>0 then OUT=[2;2*NX];else OUT=[2];end
          MaxModes=1;
          nzcross=0;
          for i=1:NMode
              Ci=evstr(exprs(5+i));// number of fileds
              ipar=[ipar;Ci];
              INP(i,1)=2*NX+length(Ci);
              if (nzcross<length(Ci)) then
                nzcross=length(Ci);
              end
              if (MaxModes<max(Ci)) then
                MaxModes=max(Ci);
                imax=i;
              end
          end

          if MaxModes>NMode then
          x_message(['Attention!: Number of Modes should be '+string(MaxModes);..
                    'A destination Mode in Mode#'+string(imax)+'"s targets is invalid!']);
          ModifEncore=%t;
          end
          if MaxModes<NMode then
          x_message(['Attention!: There is an unused Mode or the Number of Modes should
be '+string(MaxModes)]);
          ModifEncore=%t;
          end
         end

         if ~ModifEncore then
          [model,graphics,ok]=check_io(model,graphics,INP,OUT,[],[1])
          if ~ok then break,end
          model.nzcross=nzcross;
          model.state=ones(2*NX,1);
          graphics.gr_i(1)(1)='txt=["                                                    Automate";"
hybride";"nM='+string(NMode)+',nX='+string(NX)+'"];'
          graphics.exprs=exprs;
          x.graphics=graphics;
          model.ipar=ipar;
          model.rpar=rpar;
          x.model=model;
          break
         end
       end;//while
       //---------------------------------------------
     case 'define' then
       NMode=2; //number of Modes
```

```
        Minitial=1; //initial Mode
        NX=1; //number of states (NX is identical for all Modes)
        X0=[0.0];
        XP=[1;1];//xproperties for each Mode
        C1=[2];//final_Mode after Jump du to in(2*Nx+i)
        C2=[1];

exprs=[string(NMode);string(Minitial);string(NX);sci2exp(X0);sci2exp(XP);sci2exp(C1);s
ci2exp(C2)];
        ipar=[NMode;Minitial;NX;XP;C1;C2];
        rpar=[X0];

        model=scicos_model();
        model.sim=list('automa',10004
        model.in=[2*NX+1;2*NX+1];//number of Modes
        model.out=[2;2*NX];// [Mode;(x,xd)]
        model.state=ones(2*NX,1);// [x;xd]
        model.nzcross=1;// max(taille_zc(Mode_i))
        model.blocktype='c';
        model.evtout=1;
        model.firing=-1;
        model.dep_ut=[%f %t];
        model.ipar=ipar;
        model.rpar=rpar;
        gr_i=['txt=["Automate";"hybride";"nM='+string(NMode)+',nX='+string(NX)+'"];'..
            ;'xstringb(orig(1),orig(2),txt,sz(1),sz(2),"fill")'];

        x=standard_define([4 3],model,exprs,gr_i);
      end

    endfunction
```

```
#include <scicos/scicos_block.h>
#include <math.h>
#include <stdio.h>

void automa(scicos_block *block,int flag)
{
  double **y=block->outptr;
  double *g=block->g;
  double *x=block->x;
  double *xd=block->xd;
  double *res=block->res;
  void **work=block->work;
  double *rpar=block->rpar;
  double *evout=block->evout;

  int *ipar=block->ipar;
  int *jroot=block->jroot;
  int  nevprt=block->nevprt;
  int  nin=block->nin;
  int *insz=block->insz;
  int ng=block->ng;

  int *Mode;
  int  NMode,NX,Minitial,i,j,k,Mi,Mf,indice;
  int *property;
  int *iparXp;
  int *iparCx;
  double *rparX0;
  int test;
  NMode=ipar[0];
  Minitial=ipar[1];
  NX=ipar[2];
  iparXp=ipar+3;
  iparCx=iparXp+NX*NMode;
  rparX0=rpar;
                                                                    //
printf("\n\r>>                x1=%g          x2=%g          x3=%g          flag=%d
t=%g",x[0],x[1],x[2],flag,get_scicos_time());
//-------------------------------------------------------------
  if (flag ==4)
  {
    if ((*work=scicos_malloc(sizeof(int)*(2+NX)))== NULL )
    {
          set_block_error(-16); return;
```

```
        }
        Mode=*work;
       property=Mode+2;
       Mode[0]=Minitial;                                    //Current Mode;
       Mode[1]=Minitial;                                    // Previous Mode
       for (i=0;i<NX;i++) property[i]=0;            // xproperties
       for (i=0;i<NX;i++) x[i]=rparX0[i];
     }
   //-----------------------------------------------------------
     else if (flag ==5)
     {
      scicos_free(*work);
     }
   //-----------------------------------------------------------
     else if  (flag ==1 || flag ==6)
     {
      Mode=*work;
      Mi=Mode[0];
      y[0][0]=Mi;                                                  //current
Mode
       y[0][1]=Mode[1];                                    //prevous Mode
       for (i=0;i<NX;i++)
        {
        y[1][i]=x[i];
        y[1][i+NX]=xd[i];
        }
     }
   //-----------------------------------------------------------
     else if (flag==0)
     {
      Mode=*work;
      Mi=Mode[0];
      for (i=0;i<NX;i++)
        res[i]=block->inptr[Mi-1][i];
     }
   //-----------------------------------------------------------
     else if (flag==7)
     {
      Mode=*work;
      property=Mode+2;
      Mi=Mode[0];
      for (i=0;i<NX;i++)
       {
       property[i] = iparXp[(Mi-1)*NX+i];
        set_pointer_xproperty(property);
       }
```

```
      }
    //-----------------------------------------------------------
      else if (flag==9)
      {
       Mode=*work;
       Mi=Mode[0];
       for (j=0;j<ng;j++)
          g[j]=0;
       for (j=0;j<insz[Mi-1]-2*NX;j++)
        g[j]=block->inptr[Mi-1][j+2*NX];

        //printf("\n\r Mi=%d, g0=%g, g1=%g, g2=%g",Mi,g[0],g[1],g[2]);
      }
    //-----------------------------------------------------------
      else if ((flag==3)&&(nevprt<0))
      {
       Mode=*work;
       Mi=Mode[0];
       indice=0;
       for (i=1;i<Mi;i++)
        indice+=insz[i-1]-2*NX;                          //number of modes
before Mi_th Mode
       for (k=0;k<insz[Mi-1]-2*NX;k++)
        if(jroot[k]==1)
         {
              evout[0]=0.0;                                  // on obtient un
signal d'activation
              break;
         }
      }
    //-----------------------------------------------------------
      else if ((flag==2)&&(nevprt<0))
      {
       Mode=*work;
       Mi=Mode[0];
       indice=0;
       Mf=Mi;                                                  // in case
where the user has defined a wrong mode destination or ZC direction.
       for (i=1;i<Mi;i++)
        indice+=insz[i-1]-2*NX;                          //number of modes
before Mi_th Mode
                                                                //
printf("\n\r Mi=%d,Mf=%d,j0=%d, j1=%d, j2=%d",Mi,Mf,jroot[0],jroot[1],jroot[2]);
       test=0;
       for (k=0;k<insz[Mi-1]-2*NX;k++)
        {
```

```
          if(jroot[k]==1)
           {
                  Mf=iparCx[indice+k];
                  Mode[0]=Mf;                                                    //  saving
the new Mode
                  Mode[1]=Mi;                                                    //  saving
the previous Mode
                  test=1;
                  break;
           }
         }
         if (test==0)
          {
          for (k=0;k<insz[Mi-1]-2*NX;k++) if(jroot[k]==-1) break;

//printf("\n\r Warning!: In Mode=%d, the jump condition #%d has crossed zero in negative
dierction",Mi,k+1);
         }
           for (i=0;i<NX;i++)
          x[i]=block->inptr[Mf-1][i+NX];                    //reinitialize the states
       }
     }
```

```
function [x,y,typ]=gen3(job,arg1,arg2)
// Copyright INRIA
x=[];y=[];typ=[];
select job
 case 'plot' then
  graphics=arg1.graphics;base=evstr(graphics.exprs(2))
  standard_draw(arg1)
case 'getinputs' then
  [x,y,typ]=standard_inputs(arg1)
case 'getoutputs' then
  [x,y,typ]=standard_outputs(arg1)
case 'getorigin' then
  [x,y]=standard_origin(arg1)

case 'set' then
  x=arg1;
  graphics=arg1.graphics;exprs=graphics.exprs
  model=arg1.model;
  while %t do
    [ok,lampi,lamtor,mupi,mutor,loi,deviation,nu1,nu2,beta1,beta2,exprs]=getvalue('Set
generator parameters',['lambda pi';'lambda tor';'mu pi';'mu tor';'exp 0 nor 1 weib
2';'deviation si nor';'si weib nu1';'si weib nu2';'si weib beta1';'si weib
beta2'],list('vec',1,'vec',1,'vec',1,'vec',1,'vec',1,'vec',1,'vec',1,'vec',1,'vec',1,'vec',1),exprs)
    if ~ok then break,end
    if lampi<0|lamtor<0|mupi<0|mutor<0 then
      message('valeurs positives !')
    else
      graphics.exprs=exprs
      model.rpar=[lampi;lamtor;mupi;mutor;loi;deviation;nu1;nu2;beta1;beta2];
      x.graphics=graphics;x.model=model
      break
    end
  end
case 'define' then
  lampi=10D-06
  lamtor=5D-06
  mupi=14D-05
  mutor=8D-05
  loi=0
  deviation=10000;
```

```
      nu1=0.8;
      nu2=0.9;
      beta1=3;
      beta2=4;
      model=scicos_model()
      model.sim=list('generateur',5)
      model.evtin=1
      model.out=[1;1;1;1]
      model.dstate=1
      model.rpar=[lampi;lamtor;mupi;mutor;loi;deviation;nu1;nu2;beta1;beta2]
      model.blocktype='c'
      model.dep_ut=[%f %f]

exprs=[string(lampi);string(lamtor);string(mupi);string(mutor);string(loi);string(deviation);
string(nu1);string(nu2);string(beta1);string(beta2)]
      gr_i=['xstringb(orig(1),orig(2),[" Generateur ";"Aleatoire "],sz(1),sz(2),"fill");']
      x=standard_define([3 2],model,exprs,gr_i)
    end
    endfunction
```

```
function block=generateur(block,flag)
block.z=1;
lampi=block.rpar(1)
lamtor=block.rpar(2)
mupi=block.rpar(3)
mutor=block.rpar(4)
loi=block.rpar(5)
deviation=block.rpar(6)
nu1=block.rpar(7)
nu2=block.rpar(8)
beta1=block.rpar(9)
beta2=block.rpar(10)

if flag==2 then
block.z=1;
elseif flag==1 then

//loi exponentielle
if loi==0 then
block.outptr(1)=block.z*(-log(1-rand(1,1))/lampi);
block.outptr(2)=block.z*(-log(1-rand(1,1))/lamtor);
block.outptr(3)=block.z*(-log(1-rand(1,1))/mupi);
block.outptr(4)=block.z*(-log(1-rand(1,1))/mutor);

//loi normale
elseif block.rpar(5)==1 then
block.outptr(1)=block.z*grand(1,1,'nor',1/lampi,deviation);
block.outptr(2)=block.z*grand(1,1,'nor',1/lamtor,deviation);
block.outptr(3)=block.z*grand(1,1,'nor',1/mupi,deviation);
block.outptr(4)=block.z*grand(1,1,'nor',1/mutor,deviation);

//loi weibull pour dÃ©faillances,exp pour rÃ©parations
elseif block.rpar(5)==2 then
block.outptr(1)=block.z*nu1*(-log(1-rand(1,1))/lampi)^(1/beta1);
block.outptr(2)=block.z*nu2*(-log(1-rand(1,1))/lamtor)^(1/beta2);
block.outptr(3)=block.z*(-log(1-rand(1,1))/mupi);
block.outptr(4)=block.z*(-log(1-rand(1,1))/mutor);
end
end
endfunction
```

Résumé

La recherche de solutions analytiques pour l'évaluation de la fiabilité en contexte dynamique n'est pas résolue dans le cas général. Un état de l'art présenté dans le **chapitre 1** montre que des approches partielles relatives à des hypothèses particulières existent. La simulation de Monte Carlo serait le seul recours, mais il n'existait pas d'outils performants permettant la simulation simultanée de l'évolution discrète du système et de son évolution continue prenant en compte les aspects probabilistes. Dans ce contexte, dans le **chapitre 2**, nous introduisons le concept d'automate stochastique hybride capable de prendre en compte tous les problèmes posés par la fiabilité dynamique et d'accéder à l'évaluation des grandeurs de la sûreté de fonctionnement par une simulation de Monte Carlo implémentée dans l'environnement Scilab-Scicos. Dans le **chapitre 3**, nous montrons l'efficacité de notre approche de simulation pour l'évaluation de la sûreté de fonctionnement en contexte dynamique sur deux cas test dont un est un benchmark de la communauté de la Sûreté de Fonctionnement. Notre approche permet de répondre aux problèmes posés, notamment celui de la prise en compte de l'influence de l'état discret, de l'état continu et de leur interaction dans l'évaluation probabiliste des performances d'un système dans lequel en outre, les caractéristiques fiabilistes des composants dépendent eux-mêmes des états continu et discret. Dans le **chapitre 4**, nous donnons une idée de l'intérêt du contrôle par supervision comme moyen de la sûreté de fonctionnement. Les concepts d'automate observateur et de contrôleur ont été introduits et illustrés sur notre cas test afin de montrer leur potentialité.

Mots-clés : Fiabilité dynamique, sûreté de fonctionnement, automate stochastique hybride, simulation de Monte Carlo, automate observateur, contrôle par supervision.

Abstract

The research of analytical solutions for reliability assessment in dynamic context is not solved in the general case. A state of the art presented in **chapter 1** shows that partial approaches exist in the case of particular hypothesis. The Monte Carlo simulation would be the only recourse, but there were no tools allowing the simultaneous simulation of the discrete evolution of the system and its continuous evolution taking into account the probabilistic aspects. In this context, in **chapter 2**, we introduce the concept of hybrid stochastic automaton capable of taking into account all the problems posed by dynamic reliability and to accede to the assessment of dependability parameters by a Monte Carlo simulation implemented in Scicos-Scilab environment. In **chapter 3**, we show the effectiveness of our approach of simulation for dependability assessment in dynamic context through two test cases of which case one is a benchmark of dependability community. Our approach responds to the posed problems, notably the consideration of the influence of the discrete state, of the continuous state and their interaction, in the probabilistic assessment of the performances of a system in which besides, the reliability characteristics of components depend themselves of the continuous and discrete states. In **chapter 4**, we give an idea of the interest of control by supervision as a means of dependability. The concepts of observer automaton and of controller have been introduced and illustrated on our test case in order to show their potential.

Keywords: dynamic reliability, dependability, hybrid stochastic automaton, Monte Carlo simulation, observer automaton, control by supervision.

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur Christophe BERENGUER, Professeur, Université de Technologie de Troyes, Troyes
Monsieur Dimitri LEFEBVRE, Professeur, Université du Havre, Le Havre

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur PÉREZ CASTANEDA Gabriel Antonio

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Evaluation par simulation de la sûreté de fonctionnement de systèmes en contexte dynamique hybride"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : « **Automatique et traitement du signal** »

Fait à Vandoeuvre, le 17 mars 2009
Le Président de l'I.N.P.L.,
F. LAURENT

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE - NANCY -

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55