ALGORITHME DE CALCUL DE DISPONIBILITE ASYMPTOTIQUE EN FIABILITE DYNAMIQUE

ALGORITHM FOR THE COMPUTATION OF ASYMPTOTIC AVAILABILITY IN DYNAMIC RELIABILITY

Cocozza-Thivent C., Desgrouas M., Mercier S.
Université de Marne-la-Vallée
Laboratoire d'Analyse et de Mathématiques Appliquées
CNRS UMR 8050
5 Bd Descartes
77454 Marne-la-Vallée Cedex 2

Résumé

Les industries s'intéressent de plus en plus à la maîtrise des risques pour contrôler les défaillances de leur matériel d'exploitation, et cela au meilleur coût. Dans ce sens, des études de sûreté de fonctionnement sont réalisées pour calculer différents critères afin d'optimiser le programme de maintenance tout en minimisant les risques. Cependant, pour certains systèmes parfois très simples (par exemple deux composants réparables en parallèle et en interaction ayant des taux de panne et de réparation non constants), il n'existe aucune méthode analytique pour calculer des quantités pourtant usuelles comme la disponibilité ou des fonctions de coûts. Comme dans L'article [4] du $\lambda\mu$ 14, nous proposons d'utiliser le formalisme de la fiabilité dynamique pour modéliser ces systèmes en nous intéressant ici aux quantités asymptotiques. Ce formalisme nous permet de décrire simplement de tels systèmes, mais ne donnent pas non plus de solutions analytiques. La méthode la plus courante pour obtenir des résultats approchés consiste à utiliser des simulations de Monte Carlo qui sont souvent gourmandes en temps de calcul. Nous proposons ici un algorithme numérique déterministe permettant d'évaluer les quantités asymptotiques. Plus précisément, il permet de déterminer la loi stationnaire qui, elle-même permettra de calculer les quantités asymptotiques souhaitées, non pas en tant que limite du régime transitoire mais directement. Nous comparons notre algorithme aux méthodes par simulation de Monte Carlo sur de petits exemples.

Summary

Industrials are more and more interested in risk management in order to control failures of their materials at the best cost. In this sense, safety assessment is done to evaluate various quantities in order to optimize maintenance policies and to minimize risk. Nevertheless, even for some very simple systems (such as two dependent repairable interacting components in parallel with non constant failure rates), no analytical method is available for the computation of even very usual quantities such as availability or cost functions. As in [4] from $\lambda\mu$ 14, we here propose to use the formalism of dynamic reliability to model such systems, with interest in asymptotic quantities. This formalism allows for simple modelling but does not provide analytical solutions either. The most current method for getting approximated results is Monte Carlo simulation, which however requires long computation times. In this paper, we propose a deterministic numerical algorithm wich provides us with direct estimation of asymptotic quantities. More precisely, it provides an approximation for the stationary distribution of the underlying process, from where the goal quantities are directly derived and not as limits of the transient ones. We compare our algorithm with Monte Carlo simulation on a few small examples.

Introduction

Les industries s'intéressent de plus en plus à la maîtrise des risques pour contrôler les défaillances de leur matériel d'exploitation, et cela au meilleur coût. Dans ce sens, des études de sûreté de fonctionnement sont réalisées pour calculer différents critères afin d'optimiser le programme de maintenance tout en minimisant les risques. Cependant, pour certains systèmes parfois très "simples", il n'existe aucune méthode analytique pour calculer des quantités pourtant usuelles en fiabilité comme la disponibilité ou des fonctions de coûts. Par exemple, si l'on considère un système formé de deux composants réparables, en parallèle et ayant des taux de panne et de réparation non constants, avec un mode commun, on ne sait pas donner d'expression analytique de sa disponibilité même asymptotique. Lorsque l'on sort ainsi du cadre indépendant ou du cadre markovien, l'évaluation de diverses quantités comme la disponibilité asymptotique ou des fonctions de coûts asymptotiques pose de gros problèmes et les industriels ne disposent souvent que de deux méthodes : soit utiliser un modèle approché pas très réaliste avec les risques d'erreurs souvent non évalués que cela comporte, soit utiliser de la simulation de Monte-Carlo sur un grand intervalle de temps jusqu'à ce que les quantités calculées soient stabilisées, ce qui engendre souvent de grands temps de calcul. Ce contexte nous a amené à réfléchir à de nouvelles méthodes et à proposer un nouvel algorithme permettant d'évaluer de telles quantités.
Lors du congrès Lambda-Mu 14, dans l'article [4], les auteurs ont proposé un algorithme basé sur une méthode de volumes finis permettant d'évaluer diverses quantités en régime transitoire pour de tels systèmes " simples " car de petite taille, mais " complexes " par leur fonctionnement. Lorsque l'on se place sur un horizon infini, l'utilisation d'un tel algorithme ne s'avère pas toujours pertinente tant d'un point de vue de la précision des résultats que des temps de calcul, cas les quantités asymptotiques sont alors calculées comme limite des quantités transitoires.
Nous proposons ici un algorithme numérique déterministe permettant cette fois-ci d'évaluer directement les quantités asymptotiques (sans devoir le faire comme limite du régime transitoire). Plus précisément, il permet de déterminer la loi stationnaire dont on en déduit directement les quantités asymptotiques recherchées.

Dans cet article, comme dans les documents [2], [4] et [5], nous décrivons ces systèmes par des modèles de fiabilité dynamique que nous présentons dans la première partie et illustrons à l'aide de deux exemples. Nous proposons un algorithme qui est décrit dans la troisième partie. Nous précisons ensuite les méthodes par simulations de Monte Carlo que nous utilisons. Enfin, nous comparons numériquement notre algorithme à ces méthodes de Monte Carlo sur de petits exemples.

Modélisation

En fiabilité dynamique, l'évolution d'un système est modélisée à l'aide d'un processus de Markov (I_t, X_t) à valeurs dans un espace non dénombrable de la forme $E \times \mathbb{R}^d$ où E est un ensemble fini.

Le matériel, qui peut se trouver dans un nombre fini d'états (différents états de marche, de panne), est décrit par I_t. Le taux de transition de l'état i vers l'état j à l'instant t dépend d'une part de ces deux états, d'autre part de variables continues représentées par le vecteur X_t. Ce taux est noté $a(i, X_t, j)$. L'évolution de la variable continue X_t est décrite par une équation différentielle qui dépend de l'état I_t :

$$\frac{dX_t}{dt} = \mathbf{v}(i, X_t), \qquad \text{sur } \{I_t = i\}. \tag{1}$$

On note $g(i, x, t)$ l'unique solution de cette équation différentielle dont la valeur initiale est x ($g(i, x, 0) = x$). $\mathbf{v}$ correspond à la vitesse d'évolution de X_t lorsque $I_t = i$.
Dans le cadre des modèles de fiabilité dynamique généraux, ces variables continues correspondent à des paramètres physiques

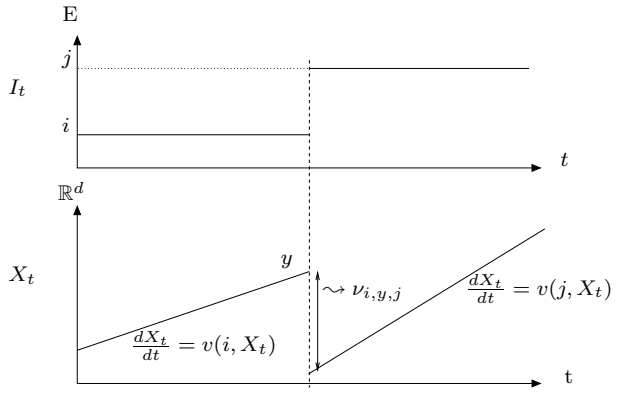

Figure 1 – Evolution d'un processus de fiabilité dynamique avec vitesses constantes

concrets (pression, température, niveau de stockage...). Ici, elles représentent généralement les durées écoulées depuis le dernier saut de I_t, c'est-à-dire depuis le dernier changement d'état, comme par exemple depuis le début d'une réparation. Les vitesses sont constantes et les solutions des équations différentielles sont de la forme $g(i, x, t) = x + c\ t$.

Lors d'un saut de I_t, la variable continue X_t peut également sauter. Lorsque le système passe de l'état i à l'état j, X_t est choisie selon une loi notée $\nu_{i,y,j}$ qui dépend de i, j et de sa valeur y juste avant le saut. Par exemple, lorsque le système tombe en panne, la variable correspondant à la durée écoulée depuis le dernier changement d'état est remise à zéro, la loi est alors une mesure de Dirac en zéro.
La figure 1 représente un exemple d'évolution du processus (I_t, X_t) pour des vitesses constantes.

Le processus (I_t, X_t) est un processus de Markov, c'est-à-dire sans mémoire. Plus précisément, le processus (I_t, X_t) est un processus de Markov déterministe par morceaux (PDMP : Piecewise Deterministic Makov Process, [6] et [7]), car entre deux sauts I_t reste constant et la trajectoire de X_t est déterministe, solution de l'équation différentielle (1).

Exemples

Pour illustrer cette modélisation, nous présentons deux exemples que nous utilisons plus loin pour tester notre algorithme. Ils n'ont pas vocation à décrire une situation industrielle, mais nous permettent de comparer nos résultats à ceux obtenus par simulation de Monte Carlo.

Exemple 1

Dans cet exemple, on considère un matériel ayant trois états différents (marche parfaite, marche dégradée et panne) fonctionnant de la manière suivante :

- Lorsqu'il est en marche parfaite, son taux de défaillance dépend de la durée x passée dans cet état et est égal à $\lambda_1(x)$.
- Le système est soumis à des événements extérieurs qui le font passer en mode dégradé. Ces événements arrivent selon un processus de Poisson de paramètre α constant ($\alpha > 0$), autrement dit, les durées entre les apparitions des événements extérieurs sont des variables aléatoires indépendantes de loi exponentielle de paramètre α.
- Lorsque le matériel est en panne, le taux de réparation dépend de la durée écoulée depuis l'instant de panne : $\mu(t)$.
- Lorsque le système est en mode dégradé, son taux de défaillance en est affecté. Il y a alors plusieurs façons de modéliser cette modification. Naturellement, si l'on considère que le taux de défaillance dépend de l'usure du système, celui-ci doit alors être continu au moment où le système passe de l'état de marche parfaite à l'état dégradé. En effet, un saut représenterait une augmentation instantanée de l'usure. Les deux premières modélisations que nous proposons plus loin correspondent à deux façons de modéliser ce point de vue.
  En revanche, le cas particulier où les taux de défaillance sont deux constantes différentes dans les états de marche parfaite et dégradée, ne rentre pas dans ce cadre-là puisqu'il y a alors une discontinuiré du taux de défaillance. Ceci nous conduit à envisager une troisième modélisation.

On peut décrire ce système en utilisant le modèle de fiabilité dynamique suivant : le système est représenté par (I_t, X_t) où I_t prend ses valeurs dans $\{1, 2, 3\} = \{$Marche parfaite, Marche dégradée, Panne$\}$, et X_t correspond à l'état d'usure du matériel pour les deux états de marche, et à la durée écoulée depuis la dernière panne si le matériel est en réparation. La figure 2 représente le graphe des transitions entre les différents états à l'instant t.

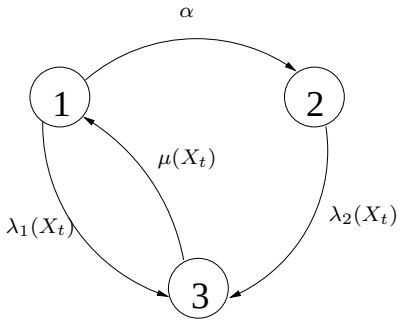

Figure 2 – Taux de transitions entre les états à l'instant t

Les valeurs des paramètres sont les suivantes :

$$
\begin{aligned}
&a(1, x, 2) = \alpha, \quad a(1, x, 3) = \lambda_1(x), \quad a(3, x, 1) = \mu(x),\\
&\mathbf{v}(1, x) = 1, \quad \mathbf{v}(3, x) = 1,\\
&\nu_{1,x,3}(dy) = \delta_0(dy), \quad \nu_{2,x,3}(dy) = \delta_0(dy),\\
&\nu_{3,x,1}(dy) = \delta_0(dy),
\end{aligned}
$$

où δ_0 est la mesure de Dirac en 0. Lorsque le matériel est en marche parfaite, la vitesse d'usure est égale à un, autrement dit l'usure correspond à la durée passée dans cet état.
Nous décrivons ci-dessous les paramètres $a(2, x, 3)$, $\nu_{1,x,2}$ et $\mathbf{v}(2, x)$ en fonction de la modélisation choisie pour la modification du taux de défaillance entre les deux états de marche.
La première modélisation consiste à dire que, en marche dégradée, l'usure du matériel est plus rapide, mais la fonction de taux est identique à celle de l'état de marche parfaite. Dans ce cas, c'est la vitesse d'évolution de la variable continue qui est augmentée en marche dégradée. Ceci se traduit à l'aide des paramètres suivants :

$$a(2, x, 3) = \lambda_1(x),\ \nu_{1,x,2}(dy) = \delta_x(dy),\ \mathbf{v}(2, x) > 1. \qquad (2)$$

La figure 3 illustre cette modélisation, elle montre l'évolution du taux de défaillance lors du passage à l'instant T de l'état de marche parfaite à celui de marche dégradée.

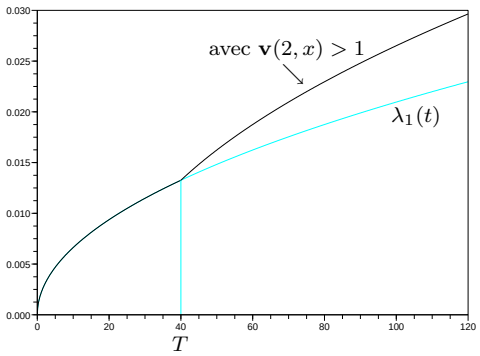

Figure 3 – Evolution du taux dans la modélisation 1

Dans la deuxième modélisation, on a deux fonctions de taux λ_1 et λ_2 qui peuvent avoir des comportements différents (par exemple une loi de Weibull et une loi Gamma). Il ne suffit pas de changer "brusquement" de fonction de taux lors du passage de l'état de marche parfaite à l'état de marche dégradée, ce qui introduirait une discontinuité qui représenterait une augmentation

instantanée de l'usure. On supprime cette discontinuité comme indiqué sur la figure 4 : si T est l'instant de saut ($T = 40$ sur la figure), on détermine le nouvel "âge virtuel" T' du matériel par la relation $\lambda_2(T') = \lambda_1(T)$ (sur la figure $T' = 20$), le taux de défaillance devenant alors la fonction λ_2.
Les paramètres du modèle de fiabilité dynamique sont alors les suivants :

$$a(2, x, 3) = \lambda_2(x),\ \mathbf{v}(2, x) = 1,\ \nu_{2,x,3}(dy) = \delta_{x-x'}(dy), \quad (3)$$

où x' est défini par $\lambda_1(x) = \lambda_2(x')$ si une telle solution existe.

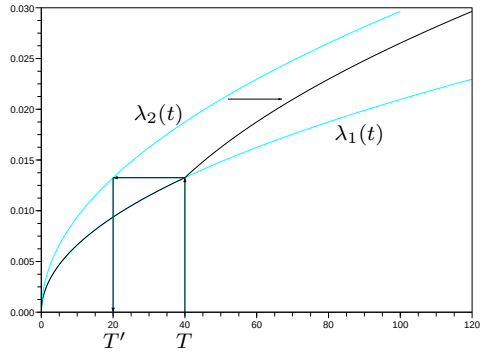

Figure 4 – Evolution du taux dans la modélisation 2

Dans ces deux modélisations il y a continuité du taux de défaillance lors du passage en mode dégradé. Comme nous l'avons signalé, ces modélisations ne permettent pas de traiter le cas où les taux de défaillance λ_1 dans l'état de marche parfaite et λ_2 dans l'état de marche dégradée sont, par exemple, constants mais différents puisque dans ce cas, il y a discontinuité du taux de défaillance lors du passage en mode dégradé.
La troisième modélisation que nous présentons est une généralisation de la première modélisation qui permet d'englober ce cas. Dans cette troisième modélisation, les taux de défaillance sont fonction de l'usure du matériel. Cette usure se fait à la vitesse $\mathbf{v}(1, x) = 1$ dans l'état de marche parfaite et à la vitesse v_2 dans l'état de marche dégradée. Les paramètres du modèle sont alors les suivants :

$$a(2, x, 3) = \lambda_2(x),\ \mathbf{v}(2, x) = v_2(x),\ \nu_{2,x,3}(dy) = \delta_x(dy). \quad (4)$$

Dans cette modélisation, le taux de défaillance comporte un saut lors du passage en mode dégradé comme l'indique la figure 5.

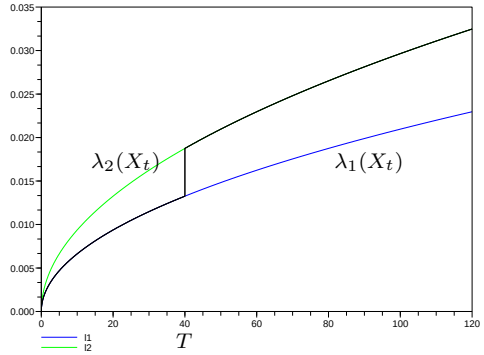

Figure 5 – Evolution du taux dans la modélisation 3

Exemple 2

On considère un système formé de deux composants en parallèle et de deux réparateurs. Les composants sont en interaction pour deux raisons : d'une part il existe une possiblité de défaillance de cause commune, et d'autre part une dépendance fonctionnelle : lorsqu'un des composants est en panne, l'autre est plus sollicité. Les défaillances de cause commune arrivent suivant un processus de Poisson de paramètre λ_{cc}. Lorsque la défaillance de cause commune a lieu, les deux composants tombent simultanément en panne.

Pour la dépendance fonctionnelle, nous utilisons la première modélisation décrite pour l'exemple 1. Le taux de défaillance du composant i est une fonction λ_i de son usure et l'usure d'un composant est plus rapide lorsque l'autre composant est en panne. L'usure de chaque composant se fait à vitesse 1 lorsque les deux composants sont en marche. Lorsque le composant 1 est en marche et le composant 2 est en panne, l'usure du composant 1 se fait à vitesse 2. Lorsque le composant 2 est en marche et le composant 1 est en panne, l'usure du composant 2 se fait à vitesse 3. Les taux de réparation sont notés μ_i et ne sont pas supposés constants.
Pour modéliser cet exemple à l'aide du formalisme de la fiabilité dynamique, on note les états possibles du système :
– 1 : les deux composants fonctionnent,
– 2 : le premier composant fonctionne mais pas le deuxième,
– 3 : le deuxième composant fonctionne mais pas le premier,
– 4 : les deux composants sont en panne.
La variable continue est le vecteur à deux dimensions $X_t = (X_t^1, X_t^2)$ pour lequel chaque composante représente l'état d'usure pour le composant correspondant lorsque celui-ci est en marche et la durée écoulée depuis le début de la réparation lorsqu'il est en panne. La figure 6 représente sous forme de graphe les transitions entre les différents états.

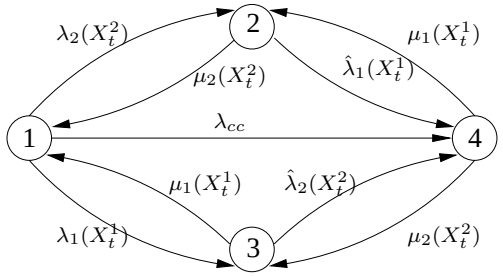

Figure 6 – Transitions entre les différents états

On obtient les paramètres suivants :

$$\mathbf{v}\big(1, (t_1, t_2)\big) = (1, 1)$$
$$a(1, (t_1, t_2), 2) = \lambda_2(t_2), \nu_{1,(t_1,t_2),2}(ds_1, ds_2) = \delta_{t_1}(ds_1)\delta_0(ds_2),$$
$$a(1, (t_1, t_2), 3) = \lambda_1(t_1), \nu_{1,(t_1,t_2),3}(ds_1, ds_2) = \delta_0(ds_1)\delta_{t_2}(ds_2),$$
$$a(1, (t_1, t_2), 4) = \lambda_{cc}, \nu_{1,(t_1,t_2),4}(ds_1, ds_2) = \delta_0(ds_1)\delta_0(ds_2),$$
$$\mathbf{v}\big(2, (t_1, t_2)\big) = (2, 1)$$
$$a(2, (t_1, t_2), 1) = \mu_2(t_2), \nu_{2,(t_1,t_2),1}(ds_1, ds_2) = \delta_{t_1}(ds_1)\delta_0(ds_2),$$
$$a(2, (t_1, t_2), 4) = \hat{\lambda}_1(t_1), \nu_{2,(t_1,t_2),4}(ds_1, ds_2) = \delta_0(ds_1)\delta_{t_2}(ds_2),$$
$$\mathbf{v}\big(3, (t_1, t_2)\big) = (1, 3)$$
$$a(3, (t_1, t_2), 1) = \mu_1(t_1), \nu_{3,(t_1,t_2),1}(ds_1, ds_2) = \delta_0(ds_1)\delta_{t_2}(ds_2),$$
$$a(3, (t_1, t_2), 4) = \hat{\lambda}_2(t_2), \nu_{3,(t_1,t_2),4}(ds_1, ds_2) = \delta_{t_1}(ds_1)\delta_0(ds_2),$$
$$\mathbf{v}\big(4, (t_1, t_2)\big) = (1, 1)$$
$$a(4, (t_1, t_2), 2) = \mu_1(t_1), \nu_{4,(t_1,t_2),2}(ds_1, ds_2) = \delta_0(ds_1)\delta_{t_2}(ds_2),$$
$$a(4, (t_1, t_2), 3) = \mu_2(t_2), \nu_{4,(t_1,t_2),3}(ds_1, ds_2) = \delta_{t_1}(ds_1)\delta_0(ds_2).$$

Algorithme

Notre objectif est de déterminer (ou d'approcher) la loi stationnaire π du processus (I_t, X_t) qui permettra ensuite de calculer des quantités comme la disponibilité ou des fonctions de coût en régime stationnaire (appelées aussi disponibilité et fonctions de coût asymptotiques).
A partir de la modélisation présentée, on peut écrire un système d'équations aux dérivées partielles vérifiées par la loi stationnaire π dont on ne sait pas calculer la solution exacte sauf dans de très rares cas. Notre algorithme repose alors sur la discrétisation de ce système d'équations. Ce système d'équations discrétisées (avec un pas h) est un système linéaire qu'il suffit ensuite de résoudre pour avoir une valeur approchée $\tilde{\pi}^h$ de la quantité recherchée

π. Les valeurs obtenues sont d'autant plus proches des valeurs réelles que le pas de discrétisation h est petit.

Principe

Il existe une interprétation plus physique et plus parlante de cet algorithme : le principe est d'approcher le processus (I_t, X_t) à valeurs dans l'espace non dénombrable $E \times \mathbb{R}^d$, par un processus markovien de saut $(\tilde{I}_t, \tilde{X}_t)$ ayant un grand nombre d'états. La loi stationnaire $\tilde{\pi}^h$ de ce processus de saut est alors solution d'un grand système linéaire. En notant A la matrice génératrice du processus markovien de saut $(\tilde{I}_t, \tilde{X}_t)$, le vecteur $\tilde{\pi}^h$ représentant sa loi stationnaire vérifie le système suivant :

$$A^t \tilde{\pi}^h = 0. \tag{5}$$

Remarque 1 *Il est bien connu ([3]) que sous des conditions d'irréductibilité, ce système possède une infinité de solutions positives toutes proportionnelles entre elles. $\tilde{\pi}^h$ est alors l'unique solution de ce système qui est une probabilité.*

Il y a alors deux étapes dans l'algorithme : d'une part la construction de cette matrice, d'autre part la résolution du système.

Construction de la matrice

Pour simplifier les formules et de ce fait les rendre plus compréhensibles, nous présentons la construction de la matrice en dimension $d = 1$ lorsque les vitesses sont positives et lorsque X_t prend ses valeurs dans $[0, +\infty[$. Le processus (I_t, X_t) est alors à valeurs dans l'espace $E \times [0, +\infty[$.
Pour approcher le processus (I_t, X_t) par un processus $(\tilde{I}_t, \tilde{X}_t)$ à valeurs dans un espace fini, on discrétise $[0, +\infty[$ en choisissant un pas de discrétisation $h > 0$ "petit" et un réel M multiple de h. On pose $K_h = M/h$, $L_k = [kh, (k+1)h[$ pour $k \in \{0, ..., K_h - 1\}$, et $L_{K_h} = [M, +\infty[$. Les états de la variable $\tilde{X}_t$ seront les intervalles L_k.

Les transitions entre les états discrets sont directement liées à celles du processus continu. Lorsque $(\tilde{I}_t, \tilde{X}_t) = (i, L_k)$, décrivons les sauts possibles. Deux cas se présentent :

- $\tilde{I}_t$ change d'état, ce qui correspond au cas où I_t change d'état, la transition se fera alors avec un taux correspondant à la moyenne des taux de saut de (I_t, X_t) sur les cellules correspondantes : pour $j \neq i$

$$A\big((i, L_k), (j, L_l)\big) = \frac{1}{h} \int_{L_k} a(i, y, j) \int_{L_l} \nu_{i,y,j}(dx) dy.$$

- $\tilde{I}_t$ ne change pas d'état : la variable I_t ne saute pas dans un autre état, par contre la variable $\tilde{X}_t$ poursuit sa trajectoire qui, dans notre cas est croissante. Ainsi $\tilde{X}_t$ va passer dans l'intervalle L_{k+1}, sauf pour $k = K_h$ auquel cas, la variable reste dans l'intervalle L_{K_h}. Pour $k < K_h$, cette transition a pour taux :

$$A\big((i, L_k), (i, L_{k+1})\big) = \frac{1}{h} \mathbf{v}(i, h(k+1)).$$

  Pour expliquer cette formule, prenons l'exemple d'une vitesse égale à un, le taux est alors $\frac{1}{h}$, et la loi du temps de saut de (i, L_k) vers (i, L_{k+1}) est exponentielle de moyenne h. Donc sur une durée de moyenne h, $\tilde{X}_t$ avance de h, ce qui correspond bien "en moyenne" à une vitesse 1.

La matrice A étant génératrice, on défini les termes diagonaux pour que la somme de chaque ligne soit nulle :

$$\begin{aligned} & A\big((i, L_k), (i, L_k)\big) \\ &= -\frac{1}{h} \mathbf{v}(i, h(k+1)) - \frac{1}{h} \sum_{j \neq i} \sum_{l=0}^{K_h} \int_{L_k} a(i, y, j) \int_{L_l} \nu_{i,y,j}(dx) dy \\ &= -\frac{1}{h} \mathbf{v}(i, h(k+1)) - \frac{1}{h} \sum_{j \neq i} \int_{L_k} a(i, y, j) dy. \end{aligned}$$

Résolution du système

Lorsque la matrice A est construite, il reste à résoudre le système (5). Compte tenu de la remarque 1, cette résolution consiste en l'inversion d'une matrice D qui correspond à la matrice A^t modifiée pour que la solution du système soit unique et que cette solution soit bien une probabilité. Pour que le vecteur $\tilde{\pi}^h$ soit une probabilité, il faut que la somme de toutes ses coordonnées soit égale à 1. La matrice D est donc par exemple égale à la matrice A^t dans laquelle on a remplacé la dernière ligne par une ligne de 1. Pour obtenir le système complet le second membre de l'équation qui est le vecteur nul, doit également être modifié. Ce second membre devient le vecteur B dont toutes les coordonnées sont nulles mis à part la dernière qui est égale à un. On doit alors résoudre le système

$$D\tilde{\pi}^h = B \tag{6}$$

qui admet, sous des conditions d'irréductibilité, une unique solution : la loi stationnaire du processus $(\tilde{I}_t, \tilde{X}_t)_t$.

Lorsque l'on résout ce système, on obtient le vecteur $\tilde{\pi}^h$ de dimension $Card(E) \times K_h$. Chaque coordonnée $\tilde{\pi}^h(i, L_k)$ approche la valeur de $\int_{L_k} \pi(i, dx)$. On considère alors que sur L_k, pour $k < K_h$, la mesure $\pi(i, dx)$ est approchée par la mesure de densité $\frac{1}{h}\tilde{\pi}(i, L_k)$.
Lorsque l'on programme, on choisit le pas de discrétisation h, mais aussi la valeur M. Le résultat donne ainsi une estimation de π sur l'intervalle $[0, M[$ et une valeur approchée de la queue de cette distribution sur $[M, +\infty[$. Un intervalle $[0, M[$ adéquat doit contenir sinon le support de π, du moins une grande partie de la masse de π.
Pour choisir M, on peut faire tourner l'algorithme pour différentes valeurs de M avec un pas de discrétisation h grand afin de cerner au mieux l'intervalle qui convient. On fixe alors M, puis on fait tourner l'algorithme avec un pas de discrétisation beaucoup plus petit.

Dans beaucoup d'exemples et notamment dans les exemples que nous présentons dans cet article, la matrice A ne contient que peu de coordonnées non nulles : elle est creuse. En utilisant la fonction sparse dans les logiciels Scilab et Matlab, cette inversion se fait sans difficulté : le résultat est direct et assez rapide dans les cas de petites dimensions (nous le verrons dans les exemples). Il peut cependant y avoir des problèmes soit lorsque la matrice n'est pas creuse, soit lorsque le nombre de variables continues est important soit lorsque le pas de discrétisation est très petit (car la dimension de la matrice devient alors très grande). Dans ce cas, on peut utiliser une méthode itérative, comme Gauss Seidel (voir [8] et [5]).

Résultats

La valeur approchée de π nous permet de calculer un certain nombre de valeurs. Par exemple la disponibilité asymptotique D_∞ qui est approchée par :

$$D_\infty = \sum_{i \in \mathcal{M}} \sum_{k=0}^{K_h} \tilde{\pi}^h(i, L_k). \tag{7}$$

où $\mathcal{M}$ est l'ensemble des états de marche.

On peut également évaluer par exemple le nombre de pannes par unité de temps : nombre de passage d'un état de marche à un état de panne. Pour cela on utilise que le nombre moyen de pannes par unité de temps est :

$$\mathbb{E}(N_p) = \int_0^\infty \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{P}} a(i, x, j) \pi(i, dx), \tag{8}$$

où $\mathcal{P}$ est l'ensemble des états de panne. On obtient ansi une valeur approchée du nombre de pannes en prenant

$$\mathbb{E}(N_p) \simeq \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{P}} \sum_{k=0}^{K_h} \frac{1}{h} \int_{L_k} a(i, x, j) dx \times \tilde{\pi}_h(i, L_k). \tag{9}$$

Méthodes par simulation

Nous présentons maintenant différentes méthodes pour calculer par simulation de Monte Carlo, d'une part la disponibilité asymptotique, d'autre part le nombre moyen de pannes par unité de temps. Les deux premières méthodes sont générales contrairement à la troisième qui s'utilise uniquement dans le cas d'existence d'un point de régénération.

Première méthode de Monte Carlo

Comme on s'intéresse à des quantités asymptotiques, la première méthode qui peut venir à l'esprit est de calculer la disponibilité asymptotique comme limite de la disponibilité instantanée. Cela revient à calculer $\mathbb{P}(I_t \in \mathcal{M})$ pour t suffisamment grand pour que cette quantité se soit stabilisée. Or cette quantité étant calculée par méthode de Monte Carlo, il faut un grand nombre de simulations pour que la stabilisation observée soit significative, et ce, sur un temps très long.

Deuxième méthode de Monte Carlo

La deuxième méthode utilise le théorème ergodique qui nous dit que :

$$\frac{1}{t}\int_0^t \mathbf{1}_{\{I_s\in\mathcal{M}\}}ds \quad \overset{t\to+\infty}{\longrightarrow} \quad D_\infty \quad p.s. \tag{10}$$

Pour l'implémenter numériquement, il s'agit de simuler une seule trajectoire sur un temps t très long. On retient le temps passé dans les états de marche, et cette quantité divisé par t donne la valeur approchée de la disponibilité asymptotique. Lors de la simulation de cette trajectoire, on peut calculer également le nombre de fois où le système est tombé en panne par unité de temps.

Monte Carlo Régénératif

Lorsque le processus étudié possède des instants de régénération, on peut utiliser une troisième méthode que nous qualifions de Monte Carlo "régénératif".
Dans l'exemple 1, cet instant peut être l'instant d'atteinte de (1,0), car le processus I_t revient dans l'état 1 en un temps fini presque sûrement (sous des hypothèses classiques/normales) et lors de ce saut, la variable continue est toujours remise à zéro.
Sous des conditions d'irréductibilité, on sait alors ([3]) que l'on peut se restreindre aux valeurs moyennes sur un cycle pour calculer la disponibilité asymptotique :

$$D_\infty \quad = \quad \frac{\mathbb{E}\left(\int_{S_0}^{S_1}\mathbf{1}_{\{I_t\in\mathcal{M}\}}dt\right)}{\mathbb{E}\left(S_1-S_0\right))}\ , \tag{11}$$

où S_0 et S_1 sont les deux premiers instants de régénération.
Cette quantité se calcule en simulant n cycles, en partant du point de régénération, et en arrêtant la trajectoire lors du retour en ce même point. Pour chaque cycle, on observe la durée passée dans les états de marche. La moyenne de ces quantités sur les n simulations permet d'estimer le numérateur de l'équation (11) qui, divisé par la durée moyenne d'un cycle, nous donne la disponibilité asymptotique.
De la même manière, on peut calculer le nombre de fois où le système tombe en panne par unité de temps.

Remarque 2 *Notons (S_n) les instants de régénération successifs. Comme les histoires du processus sur des cycles successifs de régénération sont stochastiquement indépendants, la méthode de Monte Carlo régénératif revient à calculer*

$$\frac{1}{S_n-S_0}\int_{S_0}^{S_n}\mathbf{1}_{\{I_t\in\mathcal{M}\}}dt,$$

pour n grand. On retrouve (10) avec $t = S_n$.

Remarque 3 *Pour une programmation optimale, il faut prendre comme instant initial un instant de régénération.*

L'illustration de cette remarque est très claire dans l'exemple 2, l'état initial naturel du système correspond à un matériel neuf et en état de marche. Or le processus ne reviendra jamais à cette valeur car lorsque la variable I_t saute dans l'état 1, une seule des coordonnées de X_t est remise à zéro. On peut prendre en revanche les instants d'atteinte de $(4,(0,0))$ comme instants de régénération : ils correspondent au passage de l'état 1 à l'état 4 lors d'une défaillance de cause commune. Si $\lambda_{cc} \neq 0$, ce point est atteignable en un temps fini presque sûrement. Lors des simulations, on utilise donc ce point comme valeur initiale pour le processus et le cycle se termine lorsque le processus revient à ce point. La durée d'un cycle est d'autant plus courte en moyenne que λ_{cc} est grand.

Exemples numériques

Nous comparons notre méthode aux deux méthodes de Monte Carlo décrites dans le paragraphe précédant (la deuxième et la régénérative). Nous les comparons aussi à la solution analytique théorique lorsqu'elle existe. Nous utilisons les petits exemples présentés précédemment dans lesquels nous prenons des lois exponentielles, et/ou des lois de Weibull.

Exemple 1 avec lois exponentielles

Nous nous intéressons au cas de l'exemple 1 où tous les taux sont constants : $\lambda_1(t) = \lambda_1$, $\lambda_2(t) = \lambda_2$ et $\mu(t) = \mu$. Dans ce cas particulier, la modélisation en terme de fiabilité dynamique est a priori inutile. Le processus I_t décrivant l'évolution du matériel est un processus de Markov usuel à valeurs dans $\{1,2,3\}$ dont la loi stationnaire m se calcule facilement de manière analytique. On obtient :

$$\begin{aligned} m(1) &= \frac{\lambda_2\mu}{(\lambda_2+\alpha)\mu+(\alpha+\lambda_1)\lambda_2} \\ m(2) &= \frac{\alpha\mu}{(\lambda_2+\alpha)\mu+(\alpha+\lambda_1)\lambda_2} \\ m(3) &= \frac{(\alpha+\lambda_1)\lambda_2}{(\lambda_2+\alpha)\mu+(\alpha+\lambda_1)\lambda_2}. \end{aligned} \tag{12}$$

Cependant, afin de valider les résultats fournis par notre algorithme, nous lui appliquons la modélisation de la fiabilité dynamique précédemment décrite. Nous notons donc π la loi stationnaire du processus de fiabilité dynamique (I_t, X_t). Nous sommes capables de la calculer explicitement en résolvant le système d'équations aux dérivées partielles qu'elle satisfait.

$$\begin{aligned} \pi(1,dt) &= f_1(t)dt = ce^{-(\gamma+\lambda_1)t}dt \\ \pi(2,dt) &= f_2(t)dt = \frac{c}{\gamma+\lambda_1-\lambda_2}\left(e^{-\lambda_2 t}-e^{-(\gamma+\lambda_1)t}\right)dt \\ \pi(3,dt) &= f_3(t)dt = ce^{-\mu t}dt \end{aligned} \tag{13}$$

où c est la constante de normalisation permettant à la mesure π d'être une probabilité. On vérifie immédiatement qu'on a évidemment $m(i) = \pi(i,\mathbb{R}_+)$.
Les valeurs des paramètres que nous utilisons pour les résultats qui suivent sont $\lambda_1 = 1/20, \lambda_2 = 1/10, \gamma = 1/8, \mu = 1/2$.
La figure 7 représente les densités f_1, f_2, f_3 pour les états de marche parfaite, marche dégradée et panne de la loi stationnaire théorique ainsi que les densités obtenues par notre méthode calculées sur $[0,50]$ avec un pas $h = 0,05$. Sur cette figure, nos résultats se superposent parfaitement avec les résultats théoriques. Pour plus de clarté, nous avons représenté sur la figure 8 les erreurs absolues sur f_2 qui sont inférieures à 2×10^{-4}. Les erreurs dans les deux autres états sont du même ordre.

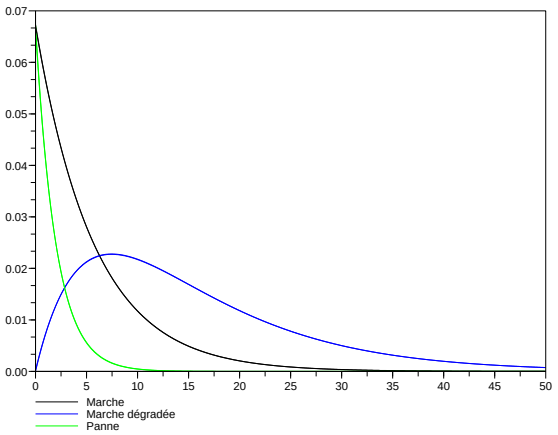

Figure 7 – Solutions superposées théorique et numérique de l'exemple avec des taux constants

Les résultats théoriques indiquent que

$$\pi(1,\mathbb{R}^+) = 0.38462, \qquad \pi(2,\mathbb{R}^+) = 0.48077, \qquad \pi(3,\mathbb{R}^+) = 0.13461. \tag{14}$$

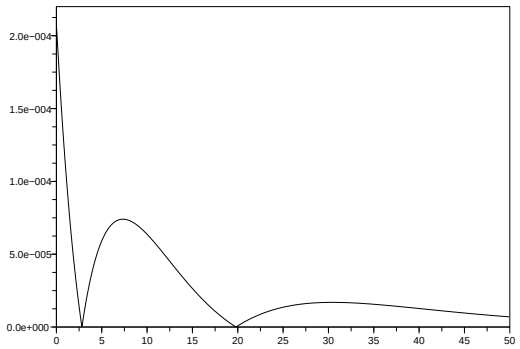

Figure 8 – Erreurs absolues entre notre algorithme et la solution théorique sur l'état de marche dégradée

Notre algorithme qui a donné les résultats instantanément sur cet exemple, fournit les valeurs approchées des probabilités de présence dans chaque état :

$$\tilde{\pi}(1,\mathbb{R}^+) \simeq 0.38461, \qquad \tilde{\pi}(2,\mathbb{R}^+) \simeq 0.48077,$$
$$\tilde{\pi}(3,\mathbb{R}^+) \simeq 0.13462.$$

Celles-ci sont exactes à 10^{-5} près. Pour la masse restante sur $[50,+\infty[$, les valeurs approchées sont

$$\tilde{\pi}^h(1,[50,+\infty[) = 6.39 \times 10^{-5}, \;\; \tilde{\pi}^h(2,[50,+\infty[) = 7.59 \times 10^{-3}$$
$$\tilde{\pi}^h(3,[50,+\infty[) = 2.6 \times 10^{-12}.$$

Ces approximations sont très proches des valeurs théoriques :

$$\pi(1,[50,+\infty[) = 6.09 \times 10^{-5}, \quad \pi(2,[50,+\infty[) = 7.46 \times 10^{-3}$$
$$\pi(3,[50,+\infty[) = 1.87 \times 10^{-12}.$$

Nous avons aussi calculé le nombre moyen de pannes par unité de temps. Les résultats sont donnés dans le tableau 1. Ceux-ci sont comparés aux résultats théoriques et aux résultats par les deux méthodes de Monte Carlo. Nous indiquons aussi la disponibilité asymptotique par les différentes méthodes et les temps de calcul. Ces calculs se font très simplement dans cet exemple car les taux sont constants.

Méthode	Disp.	Nb. pannes	Tps. calcul
Théorique	0.86538	0.06731	-
Algorithme	0.86538	0.06731	0.16 CPU
Monte Carlo	0.86539	0.06723	125 CPU
M.C. rég.	0.86549	0.06727	148 CPU

Table 1 – Comparaison des méthodes (lois exponentielles)

Il est clair que sur ce petit exemple avec des lois exponentielles, notre algorithme est beaucoup plus efficace que les simulations de Monte Carlo. On peut se demander si c'est toujours le cas lorsque l'on modifie les lois : nous considérons donc maintenant des lois de Weibull.

Exemple 1 avec lois de Weibull

Rappelons que le taux de défaillance λ d'une variable suivant la loi de Weibull de paramètres (β,η) a la forme suivante :

$$\frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1}.$$

On choisit la première modélisation de l'exemple 1 consistant à modifier la vitesse d'évolution de l'usure dans l'état de marche dégradée tout en gardant la même fonction de taux que dans l'état de marche parfaite, c'est-à-dire les paramètres décrits en (2). La variable X_t représente donc l'usure du système lorsque celui-ci est en marche et la durée passée en réparation dans l'état de panne.

Les paramètres utilisés pour obtenir les résultats sont les suivants :

- λ_1 est le taux de la loi de Weibull de paramètres $(1.5\,;1000)$, la moyenne des temps de bon fonctionnement du composant est de 965 h.
- μ est le taux de la loi de Weibull de paramètres $(3\,;2)$, la moyenne des temps de réparation est de $1,8$ h.
- l'évolution de la variable physique se fait avec vitesse 1 dans l'état de marche et celui de panne ($\mathbf{v}(1,x) = \mathbf{v}(3,x) = 1$), et avec une vitesse deux fois plus élevée dans l'état de marche dégradée : $\mathbf{v}(2,x) = 2$.

Les densités obtenues par notre algorithme sont présentés dans la figure 9 pour les états de marche et de marche dégradée et dans la figure 10 pour l'état de panne.

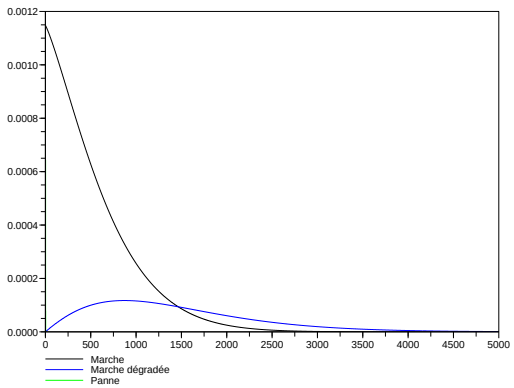

Figure 9 – Densités obtenues avec notre algorithme pour les états 1 et 2

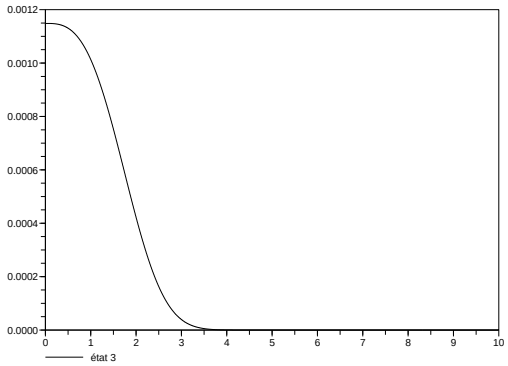

Figure 10 – Densité de l'état de panne obtenue avec notre algorithme

Le tableau 2 permet de comparer les résultats de notre algorithme à ceux obtenus par simulation de Monte Carlo en calculant la disponibilité et le nombre moyen de panne par unité de temps. Nous donnons également les temps de calcul.

Méthode	Disp.	Nb. pannes	Tps. calcul
Algorithme	0.99795	0.00115	5 CPU
Monte Carlo	0.99795	0.00114	112 CPU
M.C. rég.	0.99795	0.00115	105 CPU

Table 2 – Comparaison des méthodes (lois de Weibull)

Pour des résultats équivalents, notre méthode est beaucoup plus rapide. Par ailleurs, nous nous sommes aperçu lors des essais qu'il vaut mieux privilégier un bon choix de M (de manière à inclure quasiment toute la masse de la loi stationnaire) plutôt qu'un pas h très petit.

Exemple 2 avec lois de Weibull

Comparons maintenant notre algorithme aux méthodes de Monte Carlo sur l'exemple 2 présenté précédemment. Les paramètres utilisés sont les suivants :

- λ_1 est le taux de la loi de Weibull de paramètres (2 ;13000). La moyenne des temps de bon fontionnement du composant 1 est donc de 11 521 h.
- λ_2 est le taux de la loi de Weibull de paramètres (1.2 ;8000). La moyenne des temps de bon fontionnement du composant 2 est donc de 7 525 h.
- μ_1 est le taux de la loi de Weibull de paramètres (2.2 ;1). La moyenne des durées de réparation du composant 1 est donc de 0.88 h.
- μ_2 est le taux de la loi de Weibull de paramètres (2.5 ;1.1). La moyenne des durées de réparation du composant 2 est donc de 1 h.
- Les événements extérieurs arrivent selon le processus de Poisson de paramètre $\alpha = \frac{1}{20000}$. Les événements extérieurs arrivent en moyenne toutes les 20 000 h.

Le tableau 3 présente la disponibilité et le nombre moyen de pannes par unité de temps obtenues par les différentes méthodes ainsi que les temps de calcul.

Méthode	Disp.	Nb. pannes	Tps. calcul
Algorithme	0.999966	5.10^{-5}	45 CPU
Monte Carlo	0.999965	5.10^{-5}	508 CPU
M.C. rég.	0.999965	5.10^{-5}	512 CPU

Table 3 – Comparaison des méthodes sur l'exemple 2 (lois de Weibull)

La figure 11 présente les densité de probabilité dans les différents états.

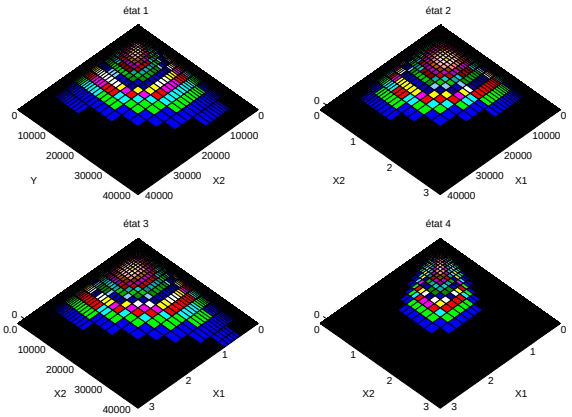

Figure 11 – Solution obtenue avec notre algorithme pour l'exemple 2

Conclusion

Dans cet article, nous avons proposé un algorithme numérique déterministe permettant de calculer la loi stationnaire 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 de saut à valeurs 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. Il constitue une alternative déterministe aux méthodes de simulation de Monte Carlo qui permet d'obtenir des résultats très satisfaisants avec des temps de calcul bien inférieurs. Cependant, en grande dimension, les logiciels de calculs tels que Scilab ou Matlab ne pourront plus résoudre directement le système linéaire, c'est pourquoi nous nous orientons vers des méthodes itératives de résolution.

Références

[1] S. Asmussen, *Applied Probability and Queues*. Wiley, 1992.

[2] O.J. Boxma, H. Kaspi, O. Kella, D. Perry, *On/Off Storage Systems with State Dependent Input, Output and Switching rates*. Probability in the Engineering and Informational Sciences, 2005, 19, p.1-13.

[3] C. Cocozza-Thivent, *Processus Stochastiques et Fiabilité des Systèmes*. Collection Mathématiques & Applications 28, Springer-Verlag, 1997.

[4] C. Cocozza-Thivent, R. Eymard, S. Mercier, *Méthodologie et algorithmes pour la quantification de petits systèmes redondants*. Congrès Lambda-Mu 14, Bourges octobre 2004, p.498-505.

[5] C. Cocozza-Thivent, R. Eymard, *Algorithmes de calculs en fiabilité dynamique*. Congrès Lambda-Mu 15, Lille, Octobre 2006.

[6] O.L.V. Costa, F. Dufour, *Stability of Piecewise-Deterministic Markov Processes*. Society for Industrial and Applied Mathematics, 1999, vol. 37, number 5, p. 1483-1502.

[7] M.H.A. Davis, *Markov Models and Optimization*. Chapman& Hall, 1993.

[8] J.F. Traub, *Iterative methods for the solution of equations*. Prentice-Hall, 1964.