ENCADREMENT DE QUANTITES FIABILISTES POUR UN GROS SYSTEME MARKOVIEN
BOUNDS FOR RELIABILITY QUANTITIES IN LARGE MARKOV MODELS

Sophie MERCIER
Université de Marne-la-Vallée
Laboratoire d'Analyse et de Mathématiques Appliquées (UMR CNRS 8050)
5, bd Descartes, F-77454 Marne la Vallée Cedex 2 - France

Résumé

La modélisation markovienne joue un rôle central dans les divers logiciels utilisés par les industriels pour l'évaluation des indicateurs usuels de fiabilité à horizon fini (fiabilité, disponibilité instantanée, coûts ou production moyenne sur un intervalle, nombre moyen de panne sur un intervalle, etc). Si l'on dispose la plupart du temps de formules exactes pour ces indicateurs, celles-ci font généralement intervenir des exponentielles de matrices. Lorsque la taille des systèmes augmente, la taille des matrices impliquées grossit exponentiellement, ce qui engendre de grosses difficultés pour leur évaluation numérique. Ce problème majeur a donné lieu à de très nombreux travaux. Nous essayons d'apporter notre contribution en proposant un nouvel algorithme de calcul approché de ces indicateurs, avec un contrôle sur la précision des résultats obtenus.

Abstract

Markov models are capital in industrial reliability software for the evaluation of usual indicators with finite horizon (such as reliability, instantaneous availability, mean costs or production on a finite interval, mean number of failures on a finite interval, ...). Most of the time, exact formulas are available for such indicators, which imply matrix exponentiation. Unfortunately, the size of the matrices involved in such formulas increases exponentially with the size of the system, often leading to difficulties for their numerical evaluation. This major problem has lead to numerous publications on the subject. We try to bring our stone by providing some new algorithm for their numerical estimation, with control on the precision of the results.

1. Introduction

Les processsus de Markov de sauts à valeurs dans un espace d'états fini sont très courants pour modéliser l'évolution d'un système industriel. L'intérêt et l'attrait d'une telle modélisation est qu'elle permet de disposer de formules analytiques pour la plupart des indicateurs de performance usuels en fiabilité. Nous nous intéressons ici à ceux dont l'évaluation numérique est a priori la plus susceptible d'engendrer des difficultés, à savoir les indicateurs dits à "horizon fini", par opposition aux indicateurs asymptotiques. Parmi ces indicateurs, on peut citer par exemple la fiabilité, la disponibilité à l'instant t (instantanée ou moyennée sur $[0,t]$), les fonctions de coûts ou de production (instantanées ou moyennées sur $[0,t]$), le nombre moyen de pannes sur $[0,t]$, les durées cumulées moyennes de panne sur $[0,t]$, etc. Pour tous ces indicateurs (et bien d'autres), on dispose, dans le cadre d'une modélisation markovienne, de formules exactes exprimées sous forme matricielle faisant très souvent intervenir des exponentielles de matrices. Lorsque les systèmes sont de petite taille ou du moins de taille raisonnable, il est alors facile de calculer numériquement ces indicateurs. Lorsque les systèmes sont plus gros, les tailles des matrices impliquées grossissent exponentiellement et les calculs numériques deviennent rapidement soit beaucoup trop gourmands en temps de calculs, soit impossibles à réaliser si l'on ne prend garde à la méthode utilisée. Ce problème majeur pour les industriels a été à l'origine de nombreux travaux : on peut par exemple consulter le livre de W. J. Stewart [15], les articles de références de C. B. Moler et C. F. Van Loan [10] et [11] (avec 161 citations), ou encore [13] ou [14], pour n'en citer que quelques-uns. Il ne s'agit pas de faire ici un quelconque panorama des méthodes existantes comme l'uniformisation, les méthodes basées sur les sous-espaces de Krylov, la résolution d'équations différentielles ordinaires, ..., ou l'une quelconque de leurs très nombreuses dérivées. Nous essayons simplement d'apporter notre contribution en proposant de nouveaux algorithmes très simples à implémenter permettant de calculer des bornes numériques pour les indicateurs usuels de la fiabilité (à horizon fini), ces bornes pouvant être ajustées autant que nécessaire. Dans un contexte industriel soumis à des contraintes de sécurité et/ou économiques, ce contrôle de la précision des résultats semble un atout par rapport à de nombreuses autres méthodes où ce contrôle est soit impossible, soit coûteux en temps de calcul. Certaines méthodes nécessitent par ailleurs des produits matriciels difficiles à effectuer lorsque la taille des sytèmes devient trop grande. Les algorithmes proposés ici n'impliquent que des produits matrice par vecteur mais pas de produit matriciel, ce qui permet leur utilisation pour de plus gros systèmes.

Le principe de base pour obtenir ces bornes consiste à "encadrer" le processus de Markov qui décrit l'évolution du système par deux processus semi-markoviens discrets, dont l'échelle de temps est $h\mathbb{N}$ ($h>0$), où h peut être choisi aussi petit qu'on le souhaite. L'intérêt d'un tel encadrement réside sur les trois points suivants :

1. la plupart des indicateurs usuels de fiabilité à horizon fini sont solutions d'équations dites de "renouvellement markovien" (cf paragraphe 2 pour plus de détails),
2. la solution d'une équation de renouvellement markovien (ERM) liée au processus de Markov initial peut être encadrée par les solutions d'ERM liées aux nouveaux processus semi-markoviens discrets, la largeur de l'encadrement tendant vers 0 lorsque h tend vers 0,
3. le calcul de ces bornes obtenues comme solutions d'ERM discrètes est très facile à implémenter.

En résumé, cette méthode permet d'encadrer la plupart des indicateurs fiabilistes à horizon fini, les bornes étant aisément calculables et convergeant vers la quantité recherchée lorsque h tend vers 0.

Notons qu'un principe d'encadrement similaire avait déjà été utilisé dans [6], [7] et [9] en dehors du cadre markovien pour évaluer d'autres quantités fiabilistes.

Ce papier est organisé de la façon suivante : les notations et les bases mathématiques sont exposées dans le paragraphe 2, ainsi que les algorithmes de calcul des bornes de la solution d'une équation de renouvellement markovien associé à un processus de Markov. Un exemple de système est décrit dans le paragraphe 3. Il est utilisé dans les paragraphes 4, 5 et 6 pour tester les encadrements

obtenus pour divers indicateurs fiabilistes. (Le cas où l'indicateur recherché est le semi-groupe de transition du processus de Markov ou une quantité dérivée, comme la disponibilité ou la fiabilité, a déjà été étudié dans [8] et n'est pas repris ici, du fait de la taille réduite de ce papier). Enfin, nous concluons dans le paragraphe 7.

2. Notations et bases mathématiques

On considére un système dont l'évolution dans le temps est décrite par un processus de Markov de sauts $(X_t)_{t\geq 0}$ à valeurs dans un espace d'états fini $E = \{1, ..., m\}$. Pour $i, j \in E$ avec $i \neq j$, on note $a_{i,j}$ le taux de transition constant de l'état i vers l'état j et $b_i = \sum_{j\neq i} a_{i,j}$. La matrice génératrice du processus $(X_t)_{t\geq 0}$ est alors $A = (A(i,j))_{i,j\in E}$ avec

$$A(i,j) = \begin{cases} a_{i,j} \text{ si } i \neq j \\ -b_i \text{ si } i = j \end{cases}$$

On note P la matrice de transition du processus $(X_t)_{t\geq 0}$ avec

$$P_{i,j} = \mathbb{P}_i(X_{T_1} = j) = \begin{cases} \frac{a_{i,j}}{b_i} \text{ si } b_i \neq 0 \\ 0 \text{ si } b_i = 0 \end{cases}$$

pour tous $i, j \in E$ où $\mathbb{P}_i$ désigne la probabilité conditionnelle sachant que $X_0 = i$ et T_1 désigne le premier instant de saut de $(X_t)_{t\geq 0}$. Les autres instants de sauts de $(X_t)_{t\geq 0}$ sont $T_0 = 0 < T_1 < ... < T_n < ...$ avec $\sup_{n\in\mathbb{N}} T_n = +\infty$ presque sûrement pour tout $i \in E$.

On note enfin $(P_t)_{t\geq 0}$ le semi-groupe de transition de $(X_t)_{t\geq 0}$ défini par

$$P_t(i,j) = \mathbb{P}_i(X_t = j)$$

pour tous $i, j \in E, t \geq 0$.

A titre d'exemple d'équation de renouvellement markovien, supposons que le système évolue dans $E = U \cup D$, où U et D représentent respectivement l'ensemble des états de marche et ceux de panne ($U \cap D = \varnothing$). Si le système part de l'état i, la disponibilité du système à l'instant t est alors $D(i,t) = \mathbb{P}_i(X_t \in U)$. En notant $\mathbf{1}_{\{.\}}$ la fonction indicatrice, $D(i,t)$ vérifie l'équation suivante :

$$\begin{aligned} D(i,t) &= \mathbb{P}_i(X_t \in U, t < T_1) + \mathbb{P}_i(X_t \in U, T_1 \leq t) \\ &= \mathbf{1}_{\{i\in U\}} \mathbb{P}_i(t < T_1) \\ &\quad + \sum_{j\in E} \mathbb{P}_i(X_t \in U, T_1 \leq t | X_{T_1} = j) \mathbb{P}_i(X_{T_1} = j) \end{aligned}$$

En utilisant la propriété de Markov à l'instant T_1 et le fait que, sachant que $X_0 = i$, T_1 suit la loi exponentielle de paramètre b_i, on obtient :

$$\begin{aligned} &D(i,t) \\ &= \mathbf{1}_{\{i\in U\}} e^{-b_i t} + \sum_{j\in E} \int_0^t \mathbb{P}_j(X_{t-u} \in U) b_i e^{-b_i u} du \times P_{i,j} \\ &= \mathbf{1}_{\{i\in U\}} e^{-b_i t} + \sum_{j\in E} \int_0^t a_{i,j} e^{-b_i u} D(j, t-u) du \end{aligned}$$

Posons $g(i,t) = 1_{\{i\in U\}} e^{-b_i t}$ et $q(i,j,du) = a_{i,j} e^{-b_i u} du$ (pour tous $i, j \in E$). Cette équation s'écrit alors :

$$D(i,t) = g(i,t) + \sum_{j\in E} \int_0^t q(i,j,du) D(j, t-u)$$

et on dit que $D(i,t)$ vérifie une équation de renouvellement markovien.

Plus généralement, soit $\mathbb{B}_+$ l'ensemble des fonctions $f : E \times \mathbb{R}_+ \to \mathbb{R}_+$ telles que $u \longmapsto f(i,u)$ est bornée sur $[0,t]$ pour tous $t \geq 0$ et $i \in E$. On dit que $f \in \mathbb{B}_+$ vérifie une équation de renouvellement markovien s'il existe $g \in \mathbb{B}_+$ telle que

$$f(i,t) = g(i,t) + \sum_{j\in E} \int_0^t q(i,j,du) f(j, t-u) \ du$$

pour tous $i \in E$ et $t \geq 0$, ce que l'on écrit sous la forme :

$$f = g + dq * f \tag{1}$$

Rappelons qu'une telle équation a une solution f unique (sous les hypothèses de ce papier), cf [2] e.g., que nous notons $f := f_g$ dans la suite.

Pour g fixée, notre but est d'encadrer cette solution f_g par des quantités aisément calculables. Pour cela, comme indiqué dans l'introduction, le principe est de construire deux processus semi-markoviens discrets dont l'échelle de temps est $h\mathbb{N}$ et qui "encadrent" le processus markovien initial. Plus précisément, si U désigne une variable aléatoire (v.a.r.) de loi exponentielle de taux λ, on peut comme dans [6], encadrer U par deux v.a.r. discrètes U^h et U^{h+} de loi géométrique à valeurs respectivement dans $h\mathbb{N}$ et $h\mathbb{N}^*$. Lorsque $kh \leq U < (k+1)h$, il suffit en effet de poser :

$$U^h = kh \text{ et } U^{h+} = (k+1)h.$$

Il est alors facile de vérifier que

$$\mathbb{P}\left(U^h = kh\right) = \mathbb{P}(kh \leq U < (k+1)h) = e^{-\lambda kh}\left(1 - e^{-\lambda h}\right)$$

pour $k \in \mathbb{N}$ et que

$$\mathbb{P}\left(U^{h+} = kh\right) = e^{-\lambda(k-1)h}\left(1 - e^{-\lambda h}\right) \text{ pour } k \in \mathbb{N}^*.$$

En d'autres termes, U^h et U^{h+} sont de loi géométrique de paramètre $1 - e^{-\lambda h}$ à valeurs respectivement dans $h\mathbb{N}$ et $h\mathbb{N}^*$.

Le processus markovien de saut $(X_t)_{t\geq 0}$ est entièrement décrit par les états successifs qu'il visite ($Y_0 = X_0$, $Y_1 = X_{T_1}$, ... , $Y_n = X_{T_n}$, ...) et les durées inter-sauts $U_1 = T_1$, $U_2 = T_2 - T_1$, ..., $U_{n+1} = T_{n+1} - T_n$ qui sont de loi exponentielle. On peut alors construire deux processus semi-markoviens discrets dont l'échelle de temps est $h\mathbb{N}$ de telle sorte que ces nouveaux processus visitent les mêmes états Y_0, Y_1, ..., Y_n,... que $(X_t)_{t\geq 0}$, et les intervalles inter-sauts de ces nouveaux processus sont les v.a.r. discrètes de lois géométriques sur $h\mathbb{N}$ et $h\mathbb{N}^*$ qui encadrent les intervalles « inter-sauts » de $(X_t)_{t\geq 0}$.

Les noyaux semi-markoviens discrets associés ont pour masses respectives au point kh :

$$\begin{aligned} q^h(i,j,kh) &= \mathbb{P}_i(kh < T_1 \leq (k+1)h, X_{T_1} = j) \\ &= P_{i,j} e^{-b_i kh}\left(1 - e^{-b_i h}\right) \qquad (2) \\ q^{h+}(i,j,kh) &= \mathbf{1}_{\{k\geq 1\}} q^h(i,j,(k-1)h) \\ &= \mathbf{1}_{\{k\geq 1\}} P_{i,j} e^{-(k-1)b_i h}\left(1 - e^{-b_i h}\right) \qquad (3) \end{aligned}$$

pour tout $k \in \mathbb{N}$. 4

Une équation de renouvellement markovien associée à $\left(q^h(i,j,kh)\right)_{k\in\mathbb{N}}$ s'écrit :

$$f(i,Nh) = g(i,Nh) + \sum_{j\in E} \sum_{k=0}^{N} q^h(i,j,kh) f(j,(N-k)h). \tag{4}$$

L'unique solution est notée $f = f_g^h$.

De même, $f = f_g^{h+}$ est l'unique solution de

$$f(i,Nh) = g(i,Nh) + \sum_{j\in E}\sum_{k=0}^{N} q^{h+}(i,j,kh)\, f(j,(N-k)h). \tag{5}$$

Dans [9], on a montré que l'unique solution f_g de (1) pouvait être approchée par f_g^h et f_g^{h+} (dans un contexte plus général). Plus précisément, on a le résultat suivant :

Theorème 1 *Soit $(X_t)_{t\geq 0}$ un processus markovien à valeurs dans E fini. Pour toute $g \in \mathbb{B}_+$:*

1. *si $t \longmapsto g(i,t)$ est croissante pour tout $i \in E$, alors, pour tout $h > 0$:*

$$f_g^{h+} \leq f_g \leq f_g^h < +\infty \tag{6}$$

2. *si g est de la forme $g = g_1 - g_2$ avec $t \longmapsto g_j(i,t)$ croissante pour $j = 1,2$ et $i \in E$, alors, pour tout $h > 0$:*

$$f_{g_1}^{h+} - f_{g_2}^h \leq f_g = f_{g_1} - f_{g_2} \leq f_{g_1}^h - f_{g_2}^{h+} < +\infty \tag{7}$$

3. *si $u \longmapsto g(i,u)$ est uniformément continue sur $[0,t]$ pour tout $i \in E$ (où $t \geq 0$), alors :*

$$\lim_{h\to 0^+} f_g^h(i,t) = \lim_{h\to 0^+} f_g^{h+}(i,t) = f_g(i,t) \tag{8}$$

Dans le cas particulier où la matrice génératrice A de $(X_t)_{t\geq 0}$ est triangulaire et $g(i,t) = v(i)\, e^{-b_i t}$ avec v monotone, un encadrement simplifié est donné dans [8] qui n'est pas repris ici du fait de la taille réduite de ce papier.

Les inégalités (6) et (7) fournissent des bornes pour la quantité recherchée f_g. Ces bornes convergent vers f_g grâce à (8). Ces bornes sont facilement calculables et on a montré dans [8] les résultats suivants où, si M représente une matrice de taille $n_1 \times n_2$, $M(\cdot,i)$ représente la $i-$ ième colonne :

Proposition 2 *Soient $g \in \mathbb{B}_+$ et $h > 0$. Pour tout $N \in \mathbb{N}$, on pose : $f^h(\cdot,Nh) = \left(f^h(i,Nh)\right)_{i\in E}$ en colonne, de même pour $g(\cdot,Nh)$ et $f^{h+}(\cdot,Nh)$. Soit I la matrice identité de taille card(E) et*

$$\begin{aligned} D_h &= diag\left(e^{-b_i h}, i = 1..m\right) \\ C_h &= D_h + (I - D_h)\, P \\ B_h &= I - (C_h - D_h) = I - (I - D_h)\, P \end{aligned}$$

où $diag(u_1,...,u_m)$ représente la matrice diagonale ayant pour termes diagonaux u_1, ..., u_m (pour tous $u_1,...,u_m \in \mathbb{R}$).

La matrice B_h est alors inversible et

$$\begin{aligned} f_g^h(\cdot,0) &= B_h^{-1} g(\cdot,0) \\ f_g^{h+}(\cdot,0) &= g(\cdot,0) \end{aligned}$$

De plus, pour tout $N \in \mathbb{N}$:

$$\begin{aligned} &f_g^h(\cdot,(N+1)h) \\ &= B_h^{-1}\left[g(\cdot,(N+1)h) + D_h\left(f_g^h(\cdot,Nh) - g(\cdot,Nh)\right)\right] \\ &f_g^{h+}(\cdot,(N+1)h) \\ &= C_h f_g^{h+}(\cdot,Nh) - D_h g(\cdot,Nh) + g(\cdot,(N+1)h) \end{aligned}$$

Indication de démonstration. D'après $(2-3)$, on a

$$\begin{aligned} q^h(\cdot,\cdot,kh) &= D_h^k (I - D_h) P = D_h^k (C_h - D_h) \\ q^{h+}(\cdot,\cdot,kh) &= \mathbf{1}_{\{k\geq 1\}} D_h^{k-1} (C_h - D_h) \end{aligned}$$

où $q^h(\cdot,\cdot,kh) = \left(q^h(i,j,kh)\right)_{i,j\in E}$, idem pour $q^{h+}(\cdot,\cdot,kh)$. Les résultats découlent alors de (4) et (5).

Le calcul de $f_g^h(\cdot,Nh)$ et de $f_g^{h+}(\cdot,Nh)$ est très facile récursivement à partir de la proposition 2. De plus, l'algorithme induit ne nécessite pas de produit matriciel mais seulement des produits matrice par un vecteur, ce qui est important lorsque E est de grande taille (cf [12] e.g.).

De la même façon, on n'a pas besoin de calculer B_h^{-1} et seules des quantités de la forme $B_h^{-1}v$ doivent être calculées, où v est un vecteur colonne. Ceci est équivalent à résoudre des systèmes linéaires de la forme $B_h w = v$. Pour accélérer la résolution numérique de tels systèmes, on factorise B_h une fois pour toute sous la forme $B_h = LU$ avec L triangulaire inférieure et U triangulaire supérieure (éventuellement à une permutation près, cf [1] e.g. ou la documentation de Matlab). Calculer $B_h^{-1}v$ revient alors à la résolution successive des deux systèmes triangulaires $v = Lz$ et $z = Uw$, ou encore $B_h^{-1}v = U\backslash(L\backslash v)$ dans les notations de Matlab. Le conditionnement de ces systèmes est étudié dans [8] où l'on montre qu'ils sont d'autant mieux conditionnés que $h\left(\max_{i\in E} b_i - \min_{i\in E} b_i\right)$ est petit.

On peut maintenant écrire un premier algorithme :

Algorithme 3 – *Données d'entrée : h, N, A, P*

$$\begin{aligned} b &\longleftarrow - \textit{ diagonale principale de } A \textit{ écrite en colonne} \\ g &\longleftarrow \textit{entrée sous la forme d'une matrice} \\ &\quad \textit{de taille } m \times (N+1) \\ D_h &\longleftarrow diag\left(e^{-hb_i}, i = 1..m\right) \\ X &\longleftarrow (I - D_h) P \\ C_h &\longleftarrow D_h + X \\ B_h &\longleftarrow I - X \\ [L;U] &\longleftarrow \textit{factorisation } LU \textit{ de } B_h \end{aligned}$$

– *Initialisation :*

$$\begin{aligned} f(\cdot,0) &\longleftarrow U\backslash(L\backslash g(\cdot,0)) \\ f^+(\cdot,0) &\longleftarrow g(\cdot,0) \end{aligned}$$

– *Pour $k = 0$ à $N-1$ faire*

$$\begin{aligned} X &\longleftarrow g(\cdot,(k+1)h) - D_h g(\cdot,kh) \\ f(\cdot,(k+1)h) &\longleftarrow U\backslash(L\backslash[D_h f(\cdot,kh) + X]) \\ f^+(\cdot,(k+1)h) &\longleftarrow C_h f^+(\cdot,kh) + X \end{aligned}$$

– *Sorties : $f(i,kh)$ et $f^+(i,kh)$, i.e. les bornes $f_g^h(i,kh)$ et $f_g^{h+}(i,kh)$ pour $f_g(i,kh)$, pour tous $i \in E$ et $0 \leq k \leq N$.*

On n'a pas forcément besoin de connaître les valeurs de tous les $f_g(\cdot,kh)$ avec $0 \leq k \leq N$. Dans ce cas, il n'est pas nécessaire de garder en mémoire tous les $f_g^h(\cdot,kh)$ et tous les $f_g^{h+}(\cdot,kh)$, ce qui consomme de la place mémoire et ralentit l'éxécution du programme. L'algorithme précédent peut alors être simplifié. Nous en donnons une nouvelle version dans le cas particulier où $g(\cdot,(k+1)h) - D_h g(\cdot,kh)$ est indépendant de k (cas fréquent, cf exemples plus loin) lorsque seules les valeurs de $f_g(\cdot,n_0 h)$, $f_g(\cdot,2n_0 h)$, ... et $f_g(\cdot,N_0 n_0 h) = f_g(\cdot,Nh)$ sont demandées ($N = N_0 n_0$). Si on veut seulement $f_g(\cdot,Nh)$, on prendra $n_0 = N$ et $N_0 = 1$.

Algorithme 4 – *Données d'entrée : N_0 et n_0 à la place de N ; le reste est inchangé mis-à-part g pour laquelle il suffit de rentrer $g(\cdot,0)$ et $g(\cdot,h)$.*

– *Initialisation :*

$$\begin{aligned} f(\cdot,0) &\longleftarrow U\backslash(L\backslash g(\cdot,0)) \\ f^{+}(\cdot,0) &\longleftarrow g(\cdot,0) \\ X &\longleftarrow g(\cdot,h) - D_h g(\cdot,0) \end{aligned}$$

– *Pour $k = 0$ à $N_0 - 1$ faire :*

$$\left\{ \begin{array}{l} Y \longleftarrow f(\cdot,k) \\ Z \longleftarrow f^{+}(\cdot,k) \\ \text{Pour } l = 1 \text{ à } n_0 \text{ faire} \\ \left\{ \begin{array}{l} Y \longleftarrow U\backslash(L\backslash[D_h Y + X]) \\ Z \longleftarrow C_h Z + X \end{array} \right. \\ f(\cdot,k+1) \longleftarrow Y \\ f^{+}(\cdot,k+1) \longleftarrow Z \end{array} \right.$$

– *Sorties : $f(i,kh)$ et $f^{+}(i,kh)$, c'est-à-dire les bornes $f_g^h(i,kn_0h)$ et $f_g^{h+}(i,kn_0h)$ pour $f_g(i,kn_0h)$ pour tous $i \in E$ et $0 \leq k \leq N_0$ ($N = N_0 n_0$).*

3. Un système "test"

Les algorithmes du paragraphe précédent sont utilisés dans les paragraphes 4, 5 et 6 pour encadrer différents critères fiabilistes. Le système utilisé pour faire les tests numériques est le suivant :

On considère un système en parallèle comportant n composants identiques et indépendants ($n \geq 2$), de taux de panne λ et de taux de réparation μ ($\lambda > 0$ et $\mu > 0$). L'espace d'états est $E = \{0, ..., n\}$ où l'état i correspond à exactement i composants en panne. Une panne de cause commune arrive selon un processus de Poisson homogène de taux $\alpha > 0$ indépendemment du comportement interne du système. Lorsqu'un tel phénomène arrive, chaque composant est affecté instantanément et indépendemment des autres avec une probabilité $p \in [0;1]$. De la même façon, le système est contrôlé instantanément à des instants qui suivent un processus de Poisson de taux $\beta > 0$, indépendemment du comportement interne du système. Lors de ces contrôles, certains paramètres extérieurs comme le voltage, la pression, la température, ... sont ré-ajustés et on essaye de remettre en marche les composants en panne. Ces composants redémarrent instantanément et indépendemment les uns des autres avec une probabilité de succès de $1 - \gamma$ ($\gamma \in [0;1]$).

La matrice génératrice du système soumis aux pannes de cause commune et aux contrôles est :

$$A = A_1 + A_2$$

avec

$$A_1(i,j) = \left\{ \begin{array}{c} (n-i)\lambda \text{ si } j = i+1 \text{ et } 0 \leq i \leq n-1 \\ i\mu \text{ si } j = i-1 \text{ et } 1 \leq i \leq n \\ -\sum_{j \neq i} A_1(i,j) \text{ si } j = i \text{ et } 0 \leq i \leq n \end{array} \right.$$

(matrice génératrice du système sans pannes de cause commune et sans contrôles)

et, en posant $q = 1 - p$:

$$A_2(i,j) = \left\{ \begin{array}{c} C_{n-i}^{j-i} p^{j-i} q^{n-j} \alpha \text{ si } 0 \leq i \leq j-1 \leq n-1 \\ C_i^{i-j} (1-\gamma)^{i-j} \gamma^j \beta \text{ si } 0 \leq j \leq i-1 \leq n-1 \\ -\sum_{j \neq i} A_2(i,j) \text{ si } j = i \text{ et } 0 \leq i \leq n \end{array} \right.$$

(matrice génératrice correspondant aux pannes de cause commune et aux contrôles).

Par exemple, en prenant $n = 5$ (et $m = 6$), on obtient, en posant $\nu = 1 - \gamma$:

$$A = \left(\begin{array}{ccc} -5\lambda - \alpha\left(1-q^5\right) & 5\lambda + 5p\alpha q^4 & 10p^2\alpha q^3 \\ \mu + \beta\nu & * & 4\lambda + 4p\alpha q^3 \\ \beta\nu^2 & 2\mu + 2\beta\gamma\nu & * \\ \beta\nu^3 & 3\beta\gamma\nu^2 & 3\mu + 3\beta\gamma^2\nu \\ \beta\nu^4 & 4\beta\gamma\nu^3 & 6\beta\gamma^2\nu^2 \\ \beta\nu^5 & 5\beta\gamma\nu^4 & 10\beta\gamma^2\nu^3 \end{array} \right. \cdots$$

$$\cdots \left(\begin{array}{ccc} 10p^3\alpha q^2 & 5p^4\alpha q & p^5\alpha \\ 6p^2\alpha q^2 & 4p^3\alpha q & p^4\alpha \\ 3\lambda + 3p\alpha q^2 & 3p^2\alpha q & p^3\alpha \\ * & 2\lambda + 2p\alpha q & p^2\alpha \\ 4\mu + 6\beta\gamma^3\nu & * & \lambda + p\alpha \\ 10\beta\gamma^3\nu^2 & 5\mu + 5\beta\gamma^4\nu & * \end{array} \right)$$

où les $*$ sont telles que les sommes sur les lignes sont nulles.

Lorsque $p = 0$ et $\gamma = 1$, on obtient le modèle usuel d'un système parallèle avec une matrice génératrice tri-diagonale. Lorsque $0 < p < 1$, $0 < \gamma < 1$, $\alpha > 0$ et $\beta > 0$, on obtient une matrice génératrice "pleine". En prennant $\mu = 0$ et $\beta = 0$, on obtient une matrice génératrice triangulaire supérieure. Enfin, en ajustant les valeurs de α, β, λ et μ, on peut obtenir des coefficients diagonaux b_i très proches ou très disparates. Cet exemple permet donc d'effectuer des tests dans des cas très différents.

Tous les calculs numériques des paragraphes à suivre ont été effectués avec le logiciel Matlab. En particulier, les calculs d'exponentielles de matrices (pas nécessaires par notre méthode mais indispensables pour avoir une référence) sont effectués à l'aide de la fonction EXPM1 de Matlab. Cette fonction est basée sur la méthode 3 de [10] et [11] et utilise une approximation de type Padé avec changement d'échelle et élévation au carré (scaling and squaring), cf le livre de référence de Matlab : pour calculer e^B, la fonction cherche tout d'abord n tel que $\left\| \frac{A}{2^n} \right\|_2 < 1$ (étape récursive qui peut être longue si E est grand) ; une approximation de type Padé est ensuite utilisée pour calculer $e^{\frac{B}{2^n}}$ (approximation de la forme $\left[P\left(\frac{B}{2^n}\right)\right]^{-1} Q\left(\frac{B}{2^n}\right)$ où P et Q sont des polynômes) ; enfin, des élévations au carré successives donnent $e^B = \left(e^{\frac{B}{2^n}}\right)^{2^n}$. Cette méthode est connue pour donner de bons résultats (cf [10] et [11]) mais n'est pas adaptée lorsque E est grand. Elle nécessite en effet de nombreux produits matriciels et se révèle effectivement lente comparée à nos méthodes lorsque E est grand. De ce fait, les tests comparatifs n'ont pu être effectués pour n très grand et nous avons pris $n = 500$.

4. Nombre moyen de visites à j sur $[0,t]$

On s'intéresse tout d'abord à la fonction de renouvellement markovien $\rho_j(i,t) = \rho_{i,j}(t)$ définie par :

$$\rho_j(i,t) = \rho_{i,j}(t) = \sum_{n \geq 0} \mathbb{E}_i\left(\mathbf{1}_{\{T_n \leq t\}} \mathbf{1}_{\{Y_n = j\}}\right)$$

pour $i, j \in E$, $t \geq 0$, qui représente le nombre moyen de visites à j sur $[0,t]$ lorsque le système est au départ dans l'état i.

Typiquement, $\rho_{i,j}(t)$ pour $i, j \in E$ peut représenter le nombre moyen de pannes sur $[0,t]$ ou le nombre moyen de passages dans l'état de marche parfaite sur $[0,t]$. Si les pannes coûtent très cher (par exemple si une indemnité est due à un client à chaque panne du système), il est alors important d'être capable d'évaluer leur nombre moyen sur un intervalle de temps donné (par exemple la durée du contrat avec le client) et de connaître la précision du calcul effectué.

Pour appliquer les méthodes présentées précédemment, rappelons que $\rho_j(i,t) = \rho_{i,j}(t)$ vérifie une équation de renouvellement

markovien, cf [2] ou [3] e.g.. Plus précisément, en séparant le cas $n = 0$ et en utilisant la propriété de Markov à l'instant T_1 (de loi exponentielle de paramètre b_i si le système part de $X_0 = i$) pour les autres termes, on a :

$$\begin{aligned}
&\rho_j(i,t) \\
&= \mathbf{1}_{\{i=j\}} \\
&\quad + \sum_{n\geq 1}\sum_{k\in E} \mathbb{E}_i\left(\mathbf{1}_{\{T_n\leq t\}}\mathbf{1}_{\{Y_n=j\}}|Y_1=k\right)\mathbb{P}_i(Y_1=k) \\
&= \mathbf{1}_{\{i=j\}} \\
&\quad + \sum_{n\geq 1}\sum_{k\in E}\int_0^t \mathbb{E}_k\left(\mathbf{1}_{\left\{T_{n-1}\leq t-u\right\}}\mathbf{1}_{\left\{Y_{n-1}=j\right\}}\right) P_{i,k} b_i e^{-b_i u} du \\
&= \mathbf{1}_{\{i=j\}} + \sum_{n\geq 1}\sum_{k\in E}\int_0^t \rho_j(k,t-u)\, q(i,k,du) \\
&= I_j(i,t) + (dq * \rho_j)(i,t)
\end{aligned}$$

où $I_j(i,t) = \mathbf{1}_{\{i=j\}}$. Ainsi, avec nos notations :

$$\rho_j(i,t) = f_{I_j}(i,t).$$

Comme $t \longmapsto I_j(i,t)$ est croissante (car constante) pour tout $i,j \in E$ et que I_j est uniformément continue sur $E\times[0,t]$ pour tout $t \geq 0$, on sait d'après le théorème 1 que

$$\begin{aligned}
\rho^{h+}(i,j,[0,t]) &= f^{h+}_{I_j}(i,t) \\
&\leq \rho(i,j,[0,t]) \leq \rho^h(i,j,[0,t]) = f^h_{I_j}(i,t)
\end{aligned} \tag{9}$$

et que les deux bornes convergent vers $\rho(i,j,[0,t])$ quand $h \to 0^+$. De plus, comme $I_j(\cdot,(k+1)h) - D_h I_j(\cdot,kh)$ est indépendant de k, on utilise l'algorithme 4 pour calculer les bornes $\rho^h(i,j,[0,t])$ et $\rho^{h+}(i,j,[0,t])$.

Nous calculons aussi $\rho(i,j,[0,t])$ par des méthodes matricielles afin de tester nos bornes : soit $n_{k,j}(i,t)$ le nombre moyen de sauts de k à j sur $[0,t]$ lorsque le système part de l'état i défini par

$$n_{k,j}(i,t) = \mathbb{E}_i\left(\sum_{n\geq 1}\mathbf{1}_{\left\{Y_{n-1}=k\right\}}\mathbf{1}_{\{Y_n=j\}}\mathbf{1}_{\{T_n\leq t\}}\right)$$

pour $i,j,k \in E$ tels que $k \neq j$ et $t \geq 0$. On a alors :

$$\begin{aligned}
\rho_{i,j}(t) &= I(i,j) + \sum_{k\neq j} n_{k,j}(i,t) \\
&= I(i,j) + \sum_{k\neq j}\int_0^t P_s(i,k)\, a_{k,j} ds \\
&= I(i,j) + \sum_{k\neq j} x_t(i,k)\, a_{k,j}
\end{aligned} \tag{10}$$

avec

$$x_t(i,j) = \int_0^t P_s(i,j)\, ds.$$

Pour calculer les $x_t(i,j)$ avec $j \in E$, on peut remarquer que :

$$\begin{aligned}
\sum_{k\in E} x_t(i,k)\, a_{k,j} &= \sum_{k\in E}\int_0^t P_s(i,k)\, a_{k,j} ds \\
&= \int_0^t (P_s A)(i,j)\, ds \\
&= \left[P_s(i,j)\right]_0^t = (P_s - I)(i,j)
\end{aligned}$$

car $\frac{d}{ds}P_s = P_s A$. Ceci peut s'écrire : $x_t(i,\cdot)\, A = (P_s - I)(i,\cdot)$. Ce système est de rang 1 (la matrice A n'est pas inversible) mais on peut remarquer que

$$\sum_{j\in E} x_t(i,j) = \sum_{j\in E}\int_0^t P_s(i,j)\, ds = \int_0^t 1\, ds = t$$

pour tout $i \in E$. On peut alors remplacer la dernière équation du système $x_t(i,\cdot)\, A = (P_s - I)(i,\cdot)$ par cette nouvelle équation. On est ainsi ramené à un système de Cramer qui s'écrit

$$x_t(i,\cdot)\, A_1 = ((P_s - I)(i,1:m-1), t)$$

où A_1 représente la matrice A où la dernière colonne a été remplacée par des 1 et où $((P_s - I)(i,1:m-1), t)$ représente la ligne $(P_s - I)(i,\cdot)$ où le dernier élément a été remplacé par t. On utilise ensuite les exponentielles de matrices programmées dans Matlab (cf paragraphe précédent) pour calculer $P_s = e^{sA}$. Ceci permet d'obtenir les $x_t(i,k)$ avec $k \in E$ puis $\rho_{i,j}(t)$ grâce à (10).

Exemple 5 *On prend*

$$\begin{aligned}
&n = 500; t_{max} = 2.5 * 10^4; h = 400; N_0 = N_1 = 63; \\
&\alpha = 10^{-7}; \beta = 10^{-6}; p = 0.1; \\
&\lambda = 10^{-6}; \mu = 10^{-5}; \gamma = 0.9
\end{aligned}$$

et on trace $\rho_{0,0}(t)$ qui représente le nombre moyen de passages dans l'état de marche parfaite sur $[0,t]$ dans la figure 1. La durée de calcul pour calculer les 63 points est de 135 ucpu par Matlab et de 2.03 ucpu par notre méthode (pour obtenir les deux bornes).

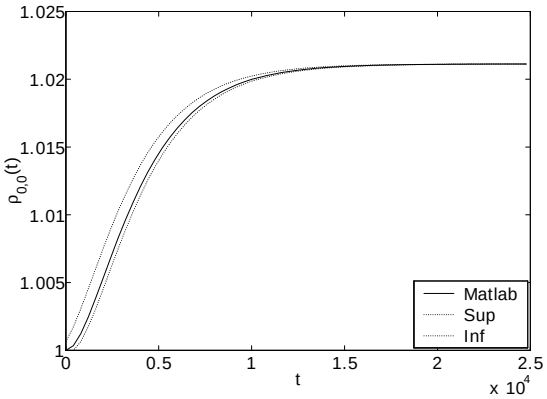

Figure 1

On constate sur cette figure que le système revient très rarement dans l'état de marche parfaite après $t = 2 \times 10^4$. On prend maintenant $\mu = 10^{-4}$, les autres paramètres du système étant inchangés. On prend par ailleurs :

$$t_{max} = 10^5; h = 100; N_0 = N_1 = 101.$$

Les durées de calcul deviennent 277.9 ucpu par Matlab et de 31.75 ucpu par notre méthode (pour obtenir les deux bornes). Les résultats sont donnés dans la figure 2.

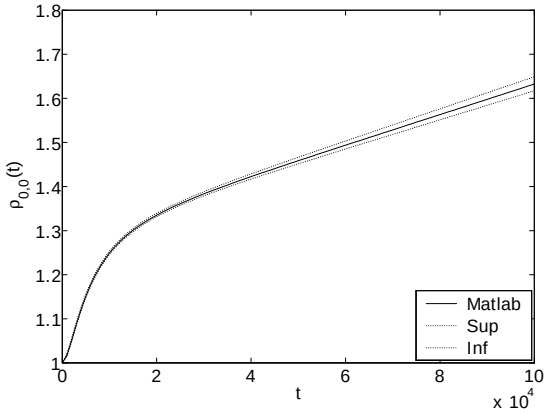

Figure 2

On constate dans cette figure qu'avec un taux de réparation plus grand, le système revient régulièrement (en moyenne) dans l'état de marche parfaite après $t = 2 \times 10^4$.

Dans les deux cas, les bornes obtenues sont cohérentes avec les résultats matriciels donnés par Matlab.

5. Nombre moyen de sauts de k à j

La connaissance des $\rho_{i,j}(t)$ n'est pas toujours suffisante lorsque l'on cherche par exemple le nombre moyen de pannes $N_P(0,t)$ sur $[0,t]$. En effet, considérons le système-test du paragraphe 3. Si le système fonctionne dès que au moins deux composants fonctionnent, le nombre moyen de pannes sur $[0,t]$ est alors

$$\begin{aligned} N_P(0,t) &= \sum_{k=0}^{n-2} \left(n_{k,n-1}(0,t) + n_{k,n}(0,t) \right) \\ &= \rho_{0,n-1}(t) + \rho_{0,n}(t) - n_{n-1,n}(0,t) - n_{n,n-1}(0,t) \end{aligned}$$

où on rappelle que $n_{k,j}(i,t)$ représente le nombre moyen de sauts de k à j sur $[0,t]$ (cf paragraphe précédent). La connaissance des $n_{k,j}(i,t)$ est donc indispensable et ne fait pas double emploi avec les $\rho_{i,j}(t)$. Notons que même si l'on peut calculer les $\rho_{i,j}(t)$ à partir des $n_{k,j}(i,t)$ (car $\rho_{i,j}(t) = I(i,j) + \sum_{k \neq j} n_{k,j}(i,t)$), il est plus rapide d'utiliser la méthode décrite au paragraphe précédent plutôt que de calculer tous les $n_{k,j}(i,t)$ et de les sommer (du moins par notre méthode puisque par Matlab, on a effectivement calculé tous les $n_{k,j}(i,t)$ pour obtenir $\rho_{i,j}(t)$).

Pour encadrer les $n_{k,j}(i,t)$, on peut remarquer qu'ils vérifient eux-aussi une équation de renouvellement markovien : en utilisant à nouveau la propriété de Markov à l'instant T_1, on a :

$$\begin{aligned} & n_{k,j}(i,t) \\ &= \mathbb{P}_i(Y_0 = k, Y_1 = j, T_1 \leq t) \\ &\quad + \sum_{n \geq 2} \mathbb{P}_i(Y_{n-1} = k, Y_n = j, T_n \leq t) \\ &= \mathbf{1}_{\{i=k\}} p_{k,j} \left(1 - e^{-b_k t}\right) \\ &\quad + \sum_{l \in E} \sum_{n \geq 2} \int_{[0,t]} q(i,l,du) \\ &\qquad \times \mathbb{P}_l(Y_{n-2} = k, Y_{n-1} = j, T_{n-1} \leq t - u) \\ &= g_{k,j}(i,t) + (dq * n_{k,j})(i,t) \end{aligned}$$

où $g_{k,j}(i,t) = \mathbf{1}_{\{i=k\}} p_{k,j} \left(1 - e^{-b_k t}\right)$.

On en déduit :

$$n_{k,j} = f_{g_{k,j}}$$

où $t \longmapsto g_{k,j}(i,t)$ est croissante pour tout $i \in E$ de sorte que $f^{h+}_{g_{k,j}} \leq f_{g_{k,j}} \leq f^{h}_{g_{k,j}}$. De plus, les deux bornes convergent vers $f_{g_{k,j}}$.

Là encore, comme

$$g_{k,j}(i,(l+1)h) - (D_h g_{k,j})(i,lh) = \mathbf{1}_{\{i=k\}} p_{k,j} \left(1 - e^{-b_k h}\right)$$

est indépendant de l, on utilise l'algorithme 4 pour calculer les bornes $f^{h}_{g_{k,j}}$ et $f^{h+}_{g_{k,j}}$.

Ces bornes sont testées en calculant les $n_{k,j}(i,t) = x_t(i,k)\, a_{k,j}$ à l'aide de la méthode matricielle décrite dans le paragraphe précédent.

Exemple 6 *On prend*

$$\begin{aligned} & n = 500; t_{max} = 9 \times 10^4; h = 300; N_0 = 51; \\ & \alpha = 10^{-7}; \beta = 10^{-6}; p = 0.1; \\ & \lambda = 10^{-6}; \mu = 10^{-5}; \gamma = 0.9 \end{aligned}$$

et on s'intéresse à $n_{0:10,11:500}(0,t) = \sum_{i=0}^{10} \sum_{j=11}^{500} n_{i,j}(0,t)$. Imaginons par exemple que la production du système soit optimale lorsque le système est dans un état élement de $\{0;1;...;10\}$ et strictement inférieure sinon. $n_{0:10,11:500}(0,t)$ représente alors le nombre moyen de pertes de production optimale sur $[0,t]$. S'il faut payer une indemnité à chaque perte de la production optimale, ce nombre est alors important à connaître. $n_{0:10,11:500}(0,t)$ est tracé dans la figure 3. La durée de calcul pour les 51 points est de 108.8 ucpu par Matlab et de 9.6 ucpu par notre méthode.

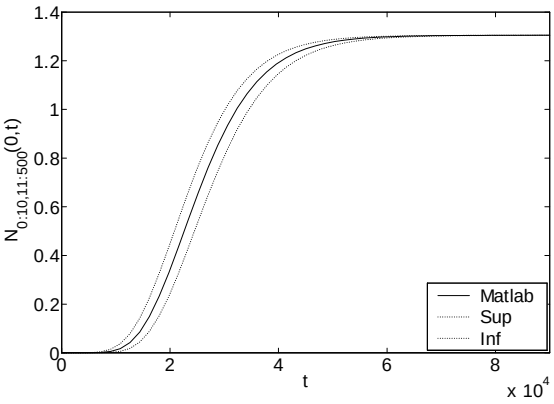

Figure 3

On cherche maintenant le nombre moyen de défaillances internes du système (pas par cause commune) sur $[0,t]$: $n_{déf}(t)$. On prend $\alpha = 0$, tous les autres paramètres du système étant inchangés. On a alors $n_{déf}(t) = \sum_{i=0}^{n} n_{i,i+1}(0,t)$ et les résultats sont donnés dans la figure 4 pour $h = 300; N_0 = 101$. La durée de calcul est de 31.8 ucpu par notre méthode pour les 101 points et de 29.4 ucpu par Matlab pour 11 points.

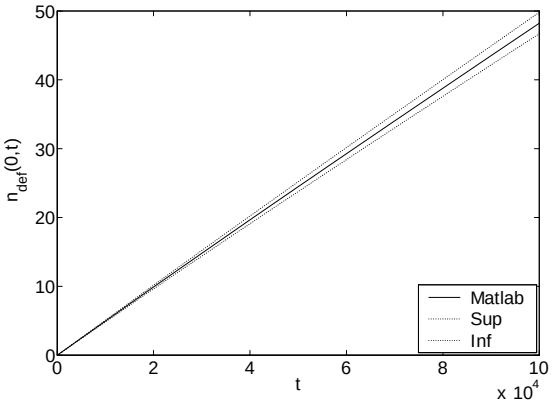

Figure 4

Dans les deux cas, les bornes sont cohérentes avec les résultats de Matlab.

6. Durée moyenne de séjour cumulée

L'indemnité due à un client du fait de pannes peut dépendre de la durée passée en panne. Il peut alors être nécessaire d'évaluer la durée moyenne cumulée passée dans un état de panne sur une durée fixée (la durée du contrat par exemple). Plus généralement, pour $i \in E$, $A \subset E$ et $t \geq 0$, on note $C_A(i,t)$la durée moyenne de séjour cumulée dans A sur $[0,t]$ lorsque le système part de i définie par :

$$C_A(i,t) = \mathbb{E}_i\left(\int_{[0,t]} \mathbf{1}_{\{X_u \in A\}} du\right)$$
$$= \sum_{j\in A} \int_{[0,t]} P_u(i,j)\, du = \sum_{j\in A} x_t(i,j)$$

On peut calculer $C_A(i,t)$ à l'aide de méthodes matricielles et de Matlab à partir des $x_t(i,j)$ comme dans les deux paragraphes précédents. De plus, $C_A(i,t)$ vérifie comme précédemment une équation de renouvellement markovien et on peut montrer que :

$$C_A(i,t) = g_A(i,t) + \sum_{j\in E} \int_0^t C_A(j,t-v)\, q(i,j,dv)$$

avec

$$g_A(i,t) = \mathbf{1}_{\{i\in A\}} \int_{[0,t]} e^{-b_i u} du$$
$$= \begin{cases} \mathbf{1}_{\{i\in A\}} \frac{1}{b_i}\left(1 - e^{-b_i u}\right) \text{ si } b_i \neq 0 \\ \mathbf{1}_{\{i\in A\}} \times t \text{ si } b_i = 0 \end{cases}$$

pour tous $i \in E$, $t \geq 0$. On a donc $C_A = f_{g_A}$où $t \longmapsto g_A(i,t)$ est croissante pour tout $i \in E$ et où g_A est uniformément continue. On en déduit $C_A^{h+} \leq C_A \leq C_A^h$ et la convergence des deux bornes quand $h \to 0^+$.

Là encore, comme

$$g_A(i,(l+1)h) - (D_h g_A)(i,lh)$$
$$= \begin{cases} \mathbf{1}_{\{i\in A\}} \frac{1}{b_i}\left(1 - e^{-b_i h}\right) \text{ si } b_i \neq 0 \\ \mathbf{1}_{\{i\in A\}} \times t \text{ si } b_i = 0 \end{cases}$$

est indépendant de l, on utilise l'algorithme 4 pour calculer les bornes C_A^h et C_A^{h+}.

Exemple 7 *On prend*

$$n = 500; t_{max} = 6 \times 10^4; h = 50; N_0 = 31;$$
$$\alpha = 10^{-7}; \beta = 10^{-6}; p = 0.1;$$
$$\lambda = 10^{-6}; \mu = 10^{-5}; \gamma = 0.9$$

et on trace $C_{\{0:5\}}(0,t) = C_{\{0,1,2,3,4,5\}}(0,t)$, qui représente la durée moyenne cumulée passée dans les états 0 à 5 sur $[0,t]$, dans la figure 5 (durée Matlab = 54 ucpu, durée pour les bornes = 3.9 ucpu). On trace ensuite $C_{\{200:500\}}(0,t)$ dans les figures 6 et 7 (sans les valeurs données par Matlab dans la figure 7), puis $C_{\{150:500\}}(0,t)$ dans la figure 8.

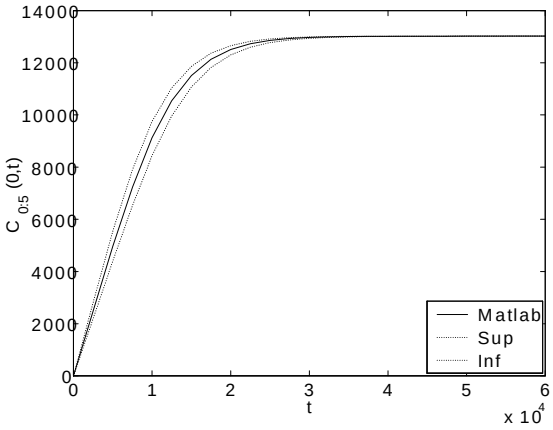

Figure 5

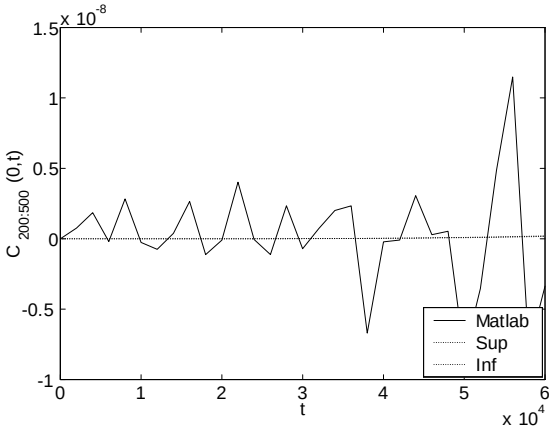

Figure 6

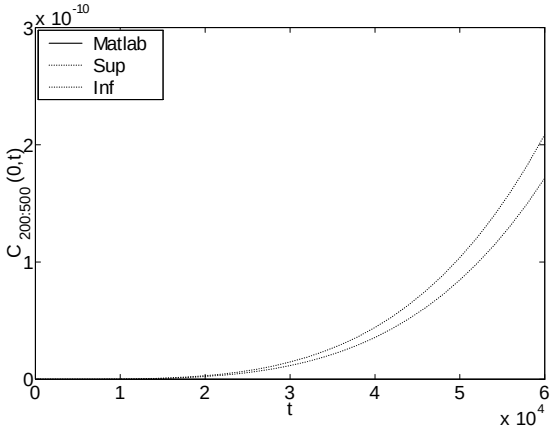

Figure 7

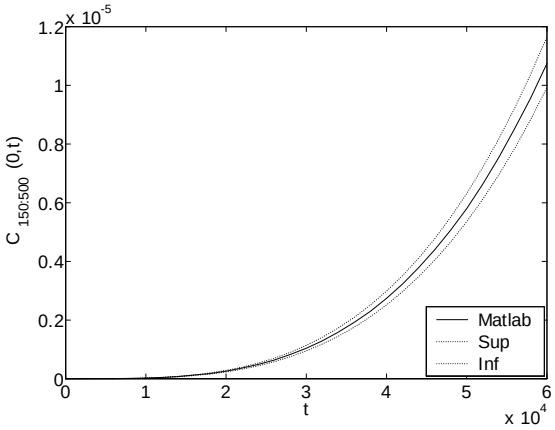

Figure 8

Enfin, on a tracé $C_{\{0:500\}}(0,t) =$ durée cumulée passée dans E sur $[0,t]$, c'est-à-dire $C_{\{0:500\}}(0,t) = t$ dans la figure 9, afin de vérifier les résultats.

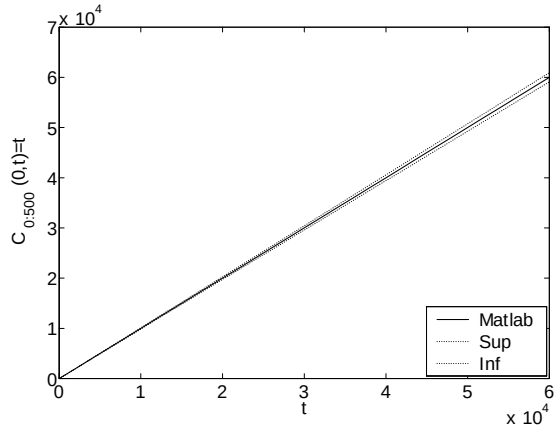

Figure 9

On constate sur ces différentes figures que les bornes sont cohérentes avec les résultats donnés par Matlab lorsque les valeurs ne sont pas trop faibles (cf figures 5, 8 et 9). En revanche, pour des résultats très faibles (ici $<3\times10^{-10}$), les résultats donnés par les bornes semblent être plus stables que ceux donnés par Matlab (cf figures 6 et 7). Notons que l'on a constaté le même phénomène pour les autres critères étudiés précédemment même si les courbes n'ont pas été données ici par manque de place.

7. Conclusion

Nous avons proposé, dans ce papier, des algorithmes de calculs de bornes pour des quantités fiabilistes qui sont solution d'équations de renouvellement markovien, et ce pour un système markovien. La plupart des quantités usuelles de la fiabilité sur un horizon fini vérifient de telles équations : on s'est ici intéressé au nombre moyen de passage dans un état donné, au nombre moyen de sauts d'un état à un autre état, à la durée moyenne passée dans un sous-espace de l'espace d'états, le tout sur un horizon fixé $[0,t]$. Typiquement, ce type de quantité est nécessaire pour un industriel pour estimer par exemple le nombre moyen de pannes sur une durée de contrat, le nombre moyen de pertes de production optimale, la disponibilité cumulée, des fonctions de coût et/ou de production cumulées, ... D'autres critères, quant à eux directement liés au semi-groupe de transition du processus de Markov associé (comme la disponibilité instantanée ou la fiabilité) peuvent aussi être encadrés par ces méthodes, comme nous l'avons montré dans [8]. La plupart des quantités usuelles de la fiabilité sur un horizon fini peuvent ainsi être encadrées. L'intérêt principal de ces encadrements est de permettre un contrôle sur la précision des résultats obtenus, ce qui, dans un contexte industriel, n'est sans doute pas à négliger. De plus, les algorithmes proposés sont très simples à implémenter et ne nécessitent aucun produit matriciel, ce qui peut permettre de les utiliser pour de gros systèmes. Leurs performances ont été testées sur un certain nombre d'exemples et semblent très prometteuses. Plus précisément, leur efficience est manifeste pour une échelle de temps pas trop grande, les temps de calculs pouvant sans doute devenir un peu lourds pour de grandes valeurs de t (?). Pour de nombreuses quantités fiabilistes, on dispose en fait d'un développement asymptotique lorsque t est grand (cf [9] pour quelques exemples et [2] pour des résultats théoriques). Une possibilité serait alors de calculer ces quantités par nos méthodes lorsque t est plus petit qu'un certain t_0 puis d'utiliser un développement asymptotique pour $t \geq t_0$. Ceci nécessiterait bien-sûr une étude préalable pour le choix de t_0. Un tel changement de formule en un certain point ("switch-over point method") a déjà été préconisé à diverses reprises pour le calcul des fonctions de renouvellement usuelles (non markovien), cf [4] et [5] e.g.. Il semble naturel d'y songer pour le calcul de quantités liées à des fonctions de renouvellement markovien comme celles envisagées ici.

8. Références

[1] P.G. Ciarlet, Introduction to Numerical Linear Algebra and Optimisation, Cambridge University Press, 1989.

[2] E. Cinlar, Introduction to stochastic processes, Englewood Cliffs, N.J. : Prentice-Hall, 1975.

[3] C. Cocozza-Thivent, *Processus stochastiques et fiabilité des systèmes*, Mathématiques et Applications n°28, Springer, 1997.

[4] From, S.G. (2001) Some new approximations for the renewal function, Communications in Statistics – Simulation and Computation, 30 (1), 113–128.

[5] Garg, A., Kalagnanam, J.R. (1998) Approximations for the renewal function, IEEE Transactions on Reliability, 47 (1).

[6] S. Mercier, Encadrement de variables aléatoires et application à la fiabilité, Lambda-Mu 14, Bourges, 12-14 octobre 2004.

[7] S. Mercier, Discrete random bounds for general random variables and applications to reliability, European Journal of Operational Research, available online 15 February 2006, doi :10.1016/j.ejor.2005.09.043.

[8] S. Mercier (submitted) Bounds and approximations for continuous-time markovian transition probabilities.

[9] S. Mercier (submitted) Numerical bounds for semi-markovian quantities and applications to reliability.

[10] C. B. Moler and C. F. Van Loan, Nineteen Dubious Ways to Compute the Exponential of a Matrix, SIAM Review 20 (1979) 801–836.

[11] C. B. Moler and C. F. Van Loan, Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. SIAM Review 45 (2003) 3–49.

[12] A. Rauzy, An experimental study on iterative methods to compute transient solutions of large Markov models, Reliability Engineering & System Safety (2004) Vol. 86, Issue 1, 105–115.

[13] A. Reibman, R. M. Smith and K. S. Trivedi, Markov and Markov Reward Models : A Survey of Numerical Approaches, European Journal of Operations Research (1989) Vol. 40, 257–267.

[14] R. B. Sidje, W. J. Stewart, A numerical study of large sparse matrix exponentials arising in Markov chains, Computational Statistics & Data Analysis (1999) Volume 29 Issue 3.

[15] W. J. Stewart, Introduction to the numerical solution of Markov chains, [B] Princeton, NJ : Princeton Univ. Press, 1994.