Systèmes embarqués temps réel (GIF-3004)
Département de génie électrique et de génie informatique
Hiver 2019

UNIVERSITÉ LAVAL

EXAMEN FINAL

Instructions : – Une feuille aide-mémoire recto verso manuscrite est permise ;
– Durée de l'examen : 1 h 50.

Pondération : Cet examen compte pour 25% de la note finale.

Aide-mémoire

Combinatoire

- Factorielle de n : $n! = 1 \times 2 \times 3 \times \cdots \times n \qquad (0! = 1)$
- Choisir k parmi n, selon un certain ordre (arrangement) : $A_n^k = \frac{n!}{(n-k)!}$
- Choisir k parmi n, sans ordre (combinaison) : $\binom{n}{k} = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$

Loi exponentielle

- Densité de probabilité : $p(t) = \lambda \exp(-\lambda t)$, pour $t \geq 0$
- Loi de répartition : $P(X \leq t) = 1 - \exp(-\lambda t)$, pour $t \geq 0$
- Espérance : $\mathbb{E}(t) = \frac{1}{\lambda}$
- Variance : $\mathrm{Var}(t) = \frac{1}{\lambda^2}$

Théorie des files d'attente

Modèle	$M/M/1$	$M/M/m$	$M/M/\infty$	$M/M/1/K$
Utilisation (ρ)	$\frac{\lambda}{\mu}$	$\frac{\lambda}{m\mu}$	$\frac{\lambda}{\mu}$	$\frac{\lambda}{\mu}$
Probabilité d'aucune tâche (π_0)	$1 - \frac{\lambda}{\mu} = 1 - \rho$	$\left[1 + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!} \frac{1}{1-\rho}\right]^{-1}$	$\exp(-\rho)$	$\left[1 + \sum_{n=1}^{K} (\lambda/\mu)^n\right]^{-1} = \frac{1-\rho}{1-\rho^{K+1}}$
Probabilité de n tâches (π_n)	$(1-\rho)\rho^n$	$\pi_0 \frac{(m\rho)^n}{n!}$ pour $1 \leq n \leq m$ $\pi_0 \frac{m^m}{m!}\rho^n$ pour $n > m$	$\exp(-\rho)\frac{\rho^n}{n!}$	$\frac{1-\rho}{1-\rho^{K+1}}\rho^n$ pour $0 \leq n \leq K$ 0 pour $n > K$
Taille moyenne de la file ($\mathbb{E}(X)$)	$\frac{\rho}{1-\rho}$	$\sum_{n=0}^{\infty} n\pi_n = m\rho + \frac{(m\rho)^m}{m!}\frac{\rho}{(1-\rho)^2}\pi_0$	ρ	$\frac{\rho}{1-\rho^{K+1}}\left[\frac{1-\rho^K}{1-\rho} - K\rho^K\right]$
Temps système moyen ($\mathbb{E}(S)$)	$\frac{1/\mu}{1-\rho}$	$\frac{1}{\mu} + \frac{1}{\mu}\frac{(m\rho)^m}{m!}\frac{\pi_0}{m(1-\rho)^2}$	$\frac{1}{\mu}$	$\frac{\mathbb{E}(X)}{\lambda(1-\pi_K)}$

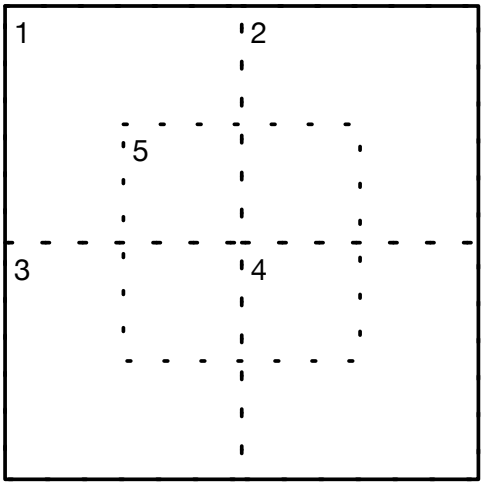
Fiabilité de systèmes complexes

- **Exactement** k composants fonctionnels sur n : $R_s = \binom{n}{k}[R]^k[1-R]^{n-k}$
- **Au moins** k composants fonctionnels sur n : $R_s = \sum_{i=k}^{n}\binom{n}{i}[R]^i[1-R]^{n-i}$

Question 1 (16 points sur 100)

Projetez-vous dans quelques années d'ici, où vous avez la tâche de concevoir un système embarqué effectuant du traitement d'image à l'aide de réseaux de neurones artificiels. Le système que vous concevez comporte un processeur standard, mais également un coprocesseur neuronal qui est utilisé pour faire du traitement d'image (un réseau de neurones utilisé en inférence seulement).

Supposons que l'on veut paralléliser le traitement en divisant chaque image en K régions. Cependant, pour que le traitement corresponde à ce qu'on veut faire avec l'image complète, il faut utiliser M sous-images, avec certaines superposées à une région et d'autres comprenant des recoupements entre régions ($M \geq K$). Par exemple, si on veut séparer une image en $K = 4$ régions, il faut utiliser $M = 5$ sous-images, une pour chaque région et une sous-image supplémentaire centrée sur l'image, comme illustré ci-bas.



Supposez que le coprocesseur neuronal peut traiter jusqu'à quatre sous-images en parallèle. De plus, faites l'hypothèse que le temps de traitement est directement et uniquement proportionnel au nombre de pixels des sous-images. Supposez également que les régions et les sous-images sont toutes de même taille, soit la taille de l'image d'origine divisée par le nombre de régions. Par exemple, lorsqu'une image comprend N pixels, chaque sous-images comportera N/K pixels.

(10) (a) Donnez une équation permettant de déterminer l'accélération (*speed-up*) parallèle possible selon cette modélisation pour le cas général.

Solution: Le temps de traitement séquentiel (t_{seq}) est directement proportionnel au nombre de pixels N multiplié par une constante que l'on nomme Q. Quant au temps de traitement parallèle (t_{par}), il correspond au temps pour traiter une région de N/K pixels multiplié par le nombre de lots nécessaires pour traiter les M sous-images en parallèle sur les quatre cœurs du coprocesseur ($\lceil M/4 \rceil$). L'équation d'accélération est donc la suivante :

$$S = \frac{t_{\text{seq}}}{t_{\text{par}}} = \frac{N\,Q}{\frac{N\,Q}{K}\left\lceil \frac{M}{4} \right\rceil} = \frac{K}{\left\lceil \frac{M}{4} \right\rceil}.$$

(6) (b) Calculez également l'accélération avec les configurations suivantes et désignez la configuration ayant le temps de traitement le plus bas :

(i) $K = 2$ régions avec $M = 3$ sous-images ;

(ii) $K = 4$ régions avec $M = 5$ sous-images ;

(iii) $K = 6$ régions avec $M = 8$ sous-images ;

(iv) $K = 9$ régions avec $M = 13$ sous-images.

Solution: L'accélération pour les configurations proposée est :

(i) $K = 2, M = 3 :\ S = \frac{2}{\lceil 3/4 \rceil} = \frac{2}{1} = 2$;

(ii) $K = 4, M = 5 :\ S = \frac{4}{\lceil 5/4 \rceil} = \frac{4}{2} = 2$;

(iii) $K = 6, M = 8 :\ S = \frac{6}{\lceil 8/4 \rceil} = \frac{6}{2} = 3$;

(iv) $K = 9, M = 13 :\ S = \frac{9}{\lceil 13/4 \rceil} = \frac{9}{4} = 2{,}25$.

La configuration (iii), avec 6 régions et 8 sous-images, offre la meilleure accélération. Elle est donc la configuration avec le temps de traitement parallèle le plus bas, comme le temps séquentiel sur toute l'image est le même pour chaque configuration.

Question 2 (24 points sur 100)

Toujours avec le système embarqué avec coprocesseur neuronal présenté à la question précédente. Supposons maintenant que le système reçoit en entrée des séquences vidéos avec un taux d'image fixe (*framerate*), mais qu'un prétraitement est effectué sur le processeur pour déterminer les images de la séquence nécessitant de plus amples analyses sur le coprocesseur neuronal.

Supposons qu'une modélisation de type théorie des files d'attente est effectuée en établissant que la distribution d'arrivée des images à traiter et la distribution de service (traitement) par le coprocesseur neuronal suivent chacune un processus de Poisson. De plus, le coprocesseur neuronal comporte également quatre cœurs permettant un traitement parallèle, chaque coprocesseur pouvant traiter une image distincte à la fois, et qu'il y a suffisamment d'espace mémoire pour stocker toutes les images en attente de traitement.

(6) (a) Supposons un taux d'arrivée de 5 images à traiter par seconde par le coprocesseur neuronal, calculez le temps de traitement (service) moyen par image pour maintenir le taux d'utilisation du coprocesseur à moins de 50 % (plus de la moitié du temps sans traiter d'image).

Solution: L'énoncé de la question suggère une modélisation de type $M/M/m$, avec $m = 4$.

Selon l'énoncé, $\lambda = 5$, $m = 4$ et la cible est d'avoir $\rho < 0{,}5$. Sachant que $\rho = \frac{\lambda}{m\,\mu}$, on peut calculer que $\mu < \frac{\lambda}{m\,\rho} = \frac{5}{4 \times 0{,}5} = 2{,}5$, donc le taux de service doit être au minimum de 2,5 images par seconde par cœur, soit un temps moyen de traitement de 400 ms ou moins.

(6) (b) Supposons maintenant que le taux d'arrivée est de 2 images par seconde et que le temps de traitement (service) moyen est de 400 ms par image, calculez la probabilité instantanée qu'il y ait une image ou plus en attente de traitement dans le système.

Solution: Sachant qu'il y a $m = 4$ cœurs de traitement, on doit calculer la probabilité $\pi_{>4}$ d'avoir plus de 4 tâches en traitement. Cette probabilité se calcule comme étant la probabilité de ne pas avoir $n = 0,1,\ldots,4$ tâches en traitement, soit $\pi_{>4} = 1 - \sum_{i=0,\ldots,4} \pi_i$.

$$
\begin{aligned}
\rho &= \frac{\lambda}{m\,\mu} = \frac{2}{4 \times \frac{1}{0,4}} = 0,2,\\
\pi_0 &= \left[1 + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!}\frac{1}{1-\rho}\right]^{-1}\\
&= \left[1 + \frac{(4 \times 0,2)^1}{1!} + \frac{(4 \times 0,2)^2}{2!} + \frac{(4 \times 0,2)^3}{3!} + \frac{(4 \times 0,2)^4}{4!}\frac{1}{1-0,2}\right]^{-1}\\
&= \left[1 + \frac{0,8}{1} + \frac{0,64}{2} + \frac{0,512}{6} + \frac{0,4096}{24}\frac{1}{0,8}\right]^{-1}\\
&= [1 + 0,8 + 0,32 + 0,085333 + 0,0213333]^{-1} = 0,449,\\
\pi_1 &= \pi_0 \frac{(m\,\rho)^1}{1!} = 0,449\frac{(4 \times 0,2)^1}{1!} = 0,449 \times 0,8 = 0,3592,\\
\pi_2 &= \pi_0 \frac{(m\,\rho)^2}{2!} = 0,449\frac{(4 \times 0,2)^2}{2!} = 0,449 \times 0,32 = 0,14368,\\
\pi_3 &= \pi_0 \frac{(m\,\rho)^3}{3!} = 0,449\frac{(4 \times 0,2)^3}{3!} = 0,449 \times 0,0853333 = 0,038315,\\
\pi_4 &= \pi_0 \frac{(m\,\rho)^4}{4!} = 0,449\frac{(4 \times 0,2)^4}{4!} = 0,449 \times 0,0170667 = 0,0076629,\\
\pi_{>4} &= 1 - (\pi_0 + \pi_1 + \pi_2 + \pi_3 + \pi_4)\\
&= 1 - (0,449 + 0.3592 + 0,14368 + 0,038315 + 0,0076629)\\
&= 1 - 0,99978579 = 0,00021421.
\end{aligned}
$$

Donc, il y a environ 0,021 % de chance qu'il y ait une image ou plus en attente de traitement.

(6) (c) Toujours avec un modèle avec un taux d'arrivée de 2 images à traiter par seconde et un temps de traitement moyen de 400 ms par image, calculez le temps système moyen nécessaire pour traiter chaque image.

Solution:

$$\begin{aligned}\mathbb{E}(S) &= \frac{1}{\mu} + \frac{1}{\mu}\frac{(m\,\rho)^m}{m!}\frac{\pi_0}{m(1-\rho)^2} \\ &= \frac{1}{2{,}5} + \frac{1}{2{,}5}\frac{(4\times 0{,}2)^4}{4!}\frac{0{,}449}{4\,(1-0{,}2)^2} = 0{,}4 + 0{,}4\frac{0{,}4096}{24}\frac{0{,}449}{2{,}56} \\ &= 0{,}4 + 0{,}4 \times 0{,}017067 \times 0{,}17539 = 0{,}4 + 0{,}000119735 = 0{,}40119735 \approx 0{,}4012.\end{aligned}$$

Une image est donc traitée en $\approx 401{,}2$ ms, en moyenne.

(6) (d) Supposons maintenant que le temps de traitement (service) est déterministe, où le traitement d'une image par un cœur du coprocesseur prend exactement 200 ms, calculez le taux d'arrivée maximal que le système peut supporter.

Solution: Le temps de traitement d'une image est constant, de sorte que le temps moyen de service est précisément égal à ce temps, soit $\mathbb{E}(Z) = Z_i = \frac{1}{\mu} = 0{,}2$ s. De plus, en régime permanent, le taux d'arrivée (λ) ne peut pas excéder le taux de départ ($m\,\mu$), donc $\lambda < m\,\mu = \frac{4}{0{,}2} = 20$. Donc, le taux d'arrivée doit être inférieur à 20 pour que le système fonctionne convenablement.

Question 3 (20 points sur 100)

Supposons maintenant que l'on utilise une version particulière du coprocesseur neuronal fonctionnant à faible voltage pour des économies d'énergie considérables, mais avec des erreurs intermittentes dans les calculs survenant à des moments aléatoires.

(8) (a) Supposons que l'on soit capable de détecter si une erreur est survenue lors d'une opération et que le taux d'erreur de l'opération est de 10^{-4}, soit une erreur pour 10 000 exécutions de l'opération. Sachant que le temps pour effectuer une opération est de 200 ms et que le temps pour refaire une nouvelle exécution d'une opération fautive est de 500 ms. Calculez le temps d'exécution moyen résultant, incluant les reprises de faute, qui ont le même taux de d'erreur. Vous pouvez supposer que les cas avec plus de deux fautes pour une opération sont très rares, donc négligeables pour nos calculs. Indiquez également si ce temps d'exécution moyen reflète bien l'impact que peuvent avoir les erreurs intermittentes sur le fonctionnement d'un système embarqué temps réel.

Solution: On va calculer le temps moyen t_m comme étant le temps pour aucune faute ($t_e = 0{,}2\,\text{s}$), plus le temps de reprise ($t_f = 0{,}5\,\text{s}$) amorti pour une faute et deux fautes selon les probabilités de fautes $P_f = 10^{-4}$.

$$\begin{aligned} t_m \approx t_e + P_f\,(t_f + P_f\,t_f) = & \underbrace{t_e}_{\text{aucune faute}} + \underbrace{P_f\,t_f}_{\text{1 faute}} + \underbrace{P_f^2\,t_f}_{\text{2 fautes}} \\ & \approx 0{,}2 + 0{,}5 \times 10^{-4} + 0{,}5 \times 10^{-8} = 0{,}200050005. \end{aligned}$$

Ce temps moyen ($0{,}200050005\,\text{s}$) est pratiquement égal au temps sans faute ($0{,}2\,\text{s}$). Cependant, dans un système embarqué temps réel, si des contraintes sont posées sur les temps d'opération, les temps totaux d'exécution correspondant à une faute ($0{,}2 + 0{,}5 = 0{,}7\,\text{s}$) ou à une double faute ($0{,}2 + 0{,}5 + 0{,}5 = 1{,}2\,\text{s}$) sont considérables et pourraient engendrer un non-respect des contraintes d'exécution temps réel.

(12) (b) Supposons maintenant que les erreurs ne sont pas directement détectables, qu'il faille effectuer la même opération plusieurs fois pour détecter et corriger les fautes, en utilisant la réponse majoritaire comme résultat. Supposons pour l'opération qui nous intéresse, le taux d'erreur est toujours de 10^{-4} par exécution d'opération. Estimez le nombre d'exécutions d'une même opération nécessaire pour assurer une exécution fiable du système, qui correspond dans notre cas à un taux d'erreur global inférieur à 10^{-9}.

Solution: Pour ce cas, comme les erreurs ne sont pas détectables, on doit faire $2K + 1$ répétitions pour être tolérant à K fautes. De plus, la probabilité d'avoir k exécutions en fautes parmi n répétitions d'une opération se calcule selon

$$F_s = \sum_{i=k}^{n} \binom{n}{i} F^i [1 - F]^{n-i},$$

où F est le taux d'erreur, aussi connu comme la probabilité de défaillance.

Pour être tolérant à $K = 1$ faute, il faut $n = (2K + 1) = 3$ répétitions et donc $k = (K + 1) = 2$ répétitions sans faute. La probabilité d'avoir plus d'une répétition en faute est donc :

$$\begin{aligned} F_s &= \sum_{i=2}^{3} \binom{3}{i} F^i [1 - F]^{3-i} = \binom{3}{2} (10^{-4})^2 [1 - 10^{-4}]^1 + \binom{3}{3} (10^{-4})^3 [1 - 10^{-4}]^0 \\ &= 3 \times 10^{-8} \times [1 - 10^{-4}] + 10^{-12} \\ &\approx 3 \times 10^{-8}. \end{aligned}$$

Ce qui est un taux d'erreur supérieur à 10^{-9}, donc pas acceptable dans le cas présent.

Pour être tolérant à $K = 2$ fautes, il faut effectuer $n = (2K + 1) = 5$ répétitions donc $k = (K + 1) = 3$ répétitions sans faute. La probabilité d'avoir plus de deux répétitions

en faute est :

$$\begin{aligned} F_s &= \sum_{i=3}^{5} \binom{5}{i} F^i [1-F]^{5-i} \\ &= \binom{5}{3}(10^{-4})^3[1-10^{-4}]^2 + \binom{5}{4}(10^{-4})^4[1-10^{-4}]^1 + \binom{5}{5}(10^{-4})^5[1-10^{-4}]^0 \\ &= 10 \times 10^{-12} \times [1-10^{-4}]^2 + 5 \times 10^{-16} \times [1-10^{-4}] + 10^{-20} \\ &\approx 5 \times 10^{-12}. \end{aligned}$$

Ce taux est inférieur à 10^{-9}, donc acceptable pour notre problème.
Il faudrait donc faire 5 répétitions pour avoir un taux d'erreur global acceptable selon les spécifications du problème.

Question 4 (40 points sur 100)

Répondez aussi brièvement et clairement que possible aux questions suivantes.

(4) (a) Expliquez de quelle façon dans Unix sont conçus les programmes selon la règle de la compositionnalité, en insistant en particulier sur la façon dont ceux-ci communiquent entre eux.

Solution: Les programmes dans Unix sont généralement conçus pour être simples et effectuer une tâche particulière, avec une communication faite à l'aide de flots textuels simples. Les programmes sont souvent écrits comme des filtres recevant un flot textuel en entrée, effectuant un traitement sur celui-ci et fournissant un flot textuel en sortie. Des applications complexes sont ainsi composées en connectant plusieurs programmes simples entre eux.

(4) (b) *Premature optimization is the root of all evil* (en français : l'optimisation prématurée est la racine de tous les maux) est un dicton célèbre en informatique, attribué à Donald Knuth. Expliquez ce qu'on entend généralement par celui-ci.

Solution: On entend par ce dicton que l'on doit d'abord développer une version fonctionnelle d'un logiciel avant de tenter d'en optimiser le fonctionnement. En effet, il vaut mieux un logiciel fonctionnel et à 90 % des performances optimales qu'un logiciel pleinement optimisé mais non fonctionnel. Ceci est cohérent avec les approches de développement itératives / incrémentales de génie logiciel, où une des premières étapes consiste à développer un prototype logiciel implantant les fonctionnalités de base désirées, les fonctionnalités plus avancées et des optimisations étant ajoutées lors d'itérations de développement subséquentes.

(4) (c) Pour la gestion des interruptions dans le noyau Linux, expliquez comment les opérations doivent se partager entre la moitié supérieure (*top-half*) et la moitié inférieure (*bottom-half*).

Solution: La moitié supérieure d'une interruption consistent aux traitements courts et immédiats nécessaires à la gestion de l'interruption. Ces traitements doivent être faits rapidement et ne sont pas préemptibles. La moitié inférieure porte sur les traitements plus élaborés et complexes, qui sont souvent implémentés sous la forme d'un KThread ou un tasklet, et qui peuvent être préemptés dans leur exécution.

(4) (d) Expliquez pourquoi la librairie C standard n'est pas disponible lorsqu'on développe du code s'exécutant dans le noyau.

Solution: La librairie C utilise des fonctionnalités du noyau, de sorte qu'on n'a pas la possibilité d'en faire usage dans le noyau directement. De plus, plusieurs routines de la librairie C sont trop grosses et trop inefficaces pour être utilisées dans le noyau.

(4) (e) Pour votre laboratoire 4, portant sur l'implémentation d'une pilote de périphérique, indiquez pourquoi le pilote est implémenté comme un dispositif de caractères, plutôt qu'un dispositif de blocs.

Solution: Le laboratoire 4 consiste en un pilote pour un clavier numérique. Les touches sont pressées séquentiellement, générant un flot de caractères séquentiel. Ceci correspond directement au modèle défini pour les dispositifs de caractères. Les dispositifs de blocs s'appliquent plutôt aux dispositifs permettant des accès aléatoires aux données, ce qui n'est pas adapté à notre cas.

(4) (f) Expliquez pourquoi les réseaux routés, comme le réseau sans-fil de l'université, ne sont pas adaptés à la communication entre dispositifs d'un système temps réels.

Solution: Avec des réseaux routés, les délais de communication peuvent comporter des délais, mais surtout une variance importante des temps de communication. De plus, ces variations dépendent souvent de facteurs externes au système temps réel, par exemple du trafic provenant d'autres utilisateurs ou machines opérants sur le réseau routé. Ceci rend le respect des contraintes d'exécution temps réel très difficiles à respecter avec certitude.

(4) (g) IEEE 1394 (FireWire) et CANbus sont deux standards de bus de communication temps réels. Expliquez les principaux éléments qui les distinguent.

Solution: IEEE 1394 est une standard pour des communications à haut débit point-à-point, mais permettant plusieurs dispositifs d'utiliser le bus. Il est conçu pour permettre, entre autres, la transmission audio et vidéo à débit régulier, avec réservation de bande passante.
CANbus est un bus de communication à faible débit pour communication entre micro-contrôleurs, ne nécessitant pas d'ordinateurs pour le contrôle des activités. C'est un standard utilisé répondu dans le domaine de l'automobile et d'autres domaines, permettant de synchroniser et transmettre des statuts de dispositifs à faible coût.

(4) (h) Pour votre laboratoire 5 portant sur la transmission audio sans fil entre deux unités embarquées, indiquez si une synchronisation des horloges entre les unités est nécessaire.

Solution: Seul un étiquettage séquentiel des paquets de données audio est nécessaire pour assurer l'ordre des paquets soit respecté lorsque le son sera joué. En ce sens, le fait que les unités aient des horloges synchronisées n'est pas nécessaire.

(4) (i) Expliquez pourquoi le choix du niveau de granularité des tâches correspondant à une décomposition parallèle d'une application se décline généralement comme un compromis, une granularité très élevée ou très faible n'étant généralement pas souhaitable.

Solution: Dans une application parallèle, une granularité très faible peut rendre la parallélisation difficile ou inefficace, avec pas assez de tâches exécutables en parallèle. Avec une granularité élevée, des surcoûts de traitement et synchronisation peuvent devenir importants relativement aux coûts de l'exécution parallèle. Il faut donc trouver la bonne granularité d'une application, pour avoir le compromis nous offrant les meilleures performances.

(4) (j) La courbe en forme de baignoire permet de décrire l'évolution du taux de panne d'une grande variété de composants. Expliquez pourquoi utilise-t'on souvent une loi exponentielle pour modéliser la distribution de défaillance de composants dont le taux de panne suit la forme d'une courbe en baignoire.

Solution: La loi exponentielle permet de décrire assez fidèlement les deux premières périodes de la courbe en forme de baignoire, soit la période des pannes hâtive et la période stable. Pour la troisième période, correspondant aux pannes d'usures, on suppose qu'elles surviennent bien au-delà de la durée de vie utile du composant, de sorte qu'on n'a pas à les modéliser dans notre modèle basé sur une loi exponentielle.