Rapport d'étude de cas : Sûreté de fonctionnement

Mickaël EL ALAM

Guillaume Cullerier

Mohcine EL HARRAS

1 Problème 1

1. On peut estimer le MTTF en faisant la moyenne des temps avant défaillance : *MTTF_PT=failure_dates.mean()* (cf code Python)

   On obtient $MTTF_{PT} = 33333h$, et $\lambda_{PT} = 3 * 10^{-5}\text{h}^{-1}$

2. Comme nous avons un taux de défaillance constant, on a la relation $MTTF = \frac{1}{\lambda}$. Pour SDV, qui a un état dégradé, on a $MTTF = \frac{1}{\lambda_d + \lambda_f}$.

$\lambda_{PT} = 3 * 10^{-5}\text{h}^{-1}$	$MTTF_{PT} = 33333h$
$\lambda_{AI} = 1 * 10^{-7}\text{h}^{-1}$	$MTTF_{AI} = 10^7 h$
$\lambda_{CPU} = 6 * 10^{-7}\text{h}^{-1}$	$MTTF_{CPU} = 1.7 * 10^6 h$
$\lambda_{DO} = 1 * 10^{-7}\text{h}^{-1}$	$MTTF_{DO} = 10^7 h$
$\lambda_{DCV} = 1 * 10^{-6}\text{h}^{-1}$	$MTTF_{DCV} = 10^6 h$
$\lambda_{dSDV} = 2.1 * 10^{-6}\text{h}^{-1}$ $\lambda_{fSDV} = 1.9 * 10^{-5}\text{h}^{-1}$	$MTTF_{SDV} = 4.7 * 10^4 h$

   La probabilité qu'un composant soit fonctionnel à sa MTTF est $R(MTTF) = e^{-\lambda * MTTF} = e^{-1}$ ce qui vaut environ 0.37 quel que soit le composant (car le taux de défaillance est constant).

3. Tracés des fonctions de fiabilité

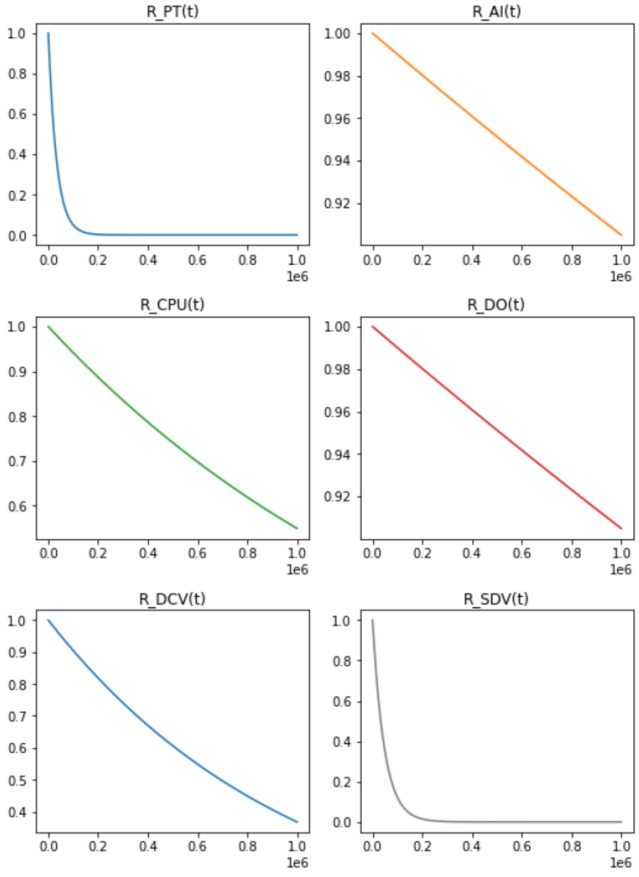

Figure 1 - Tracés de la fiabilité pour les différents composants

L'horizon pour que la moitié des composants aient une fiabilité supérieure à 0.9 est égal à :

$$[0\ ,2.10^5 h]$$

Dans cet horizon, on ne peut pas conclure que le système aura une fiabilité supérieure à 0.9. Pour 2 raisons

L'horizon pour que tous les composants aient une fiabilité supérieure à 0.9 est égal à :

$$[0\ ,3504h]$$

Pareillement, on ne peut pas conclure que le système aura une fiabilité supérieure à 0.9 dans cet horizon

4. Calcul de la fiabilité

   Il faudrait mener une analyse structurée, construire un diagramme de Markov sans scénario de revenir de l'état de panne, construire la matrice de transition $\mathbb{A}$ associée et ensuite déduire la fiabilité.

2 Problème 2

1. RBD du système

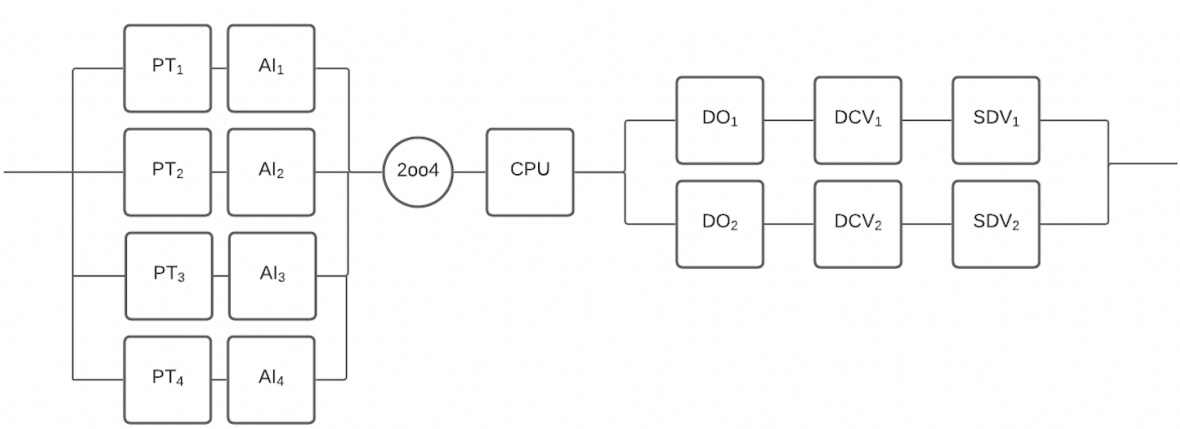

Figure 2 - RBD du système complet

2. La coupe minimale est $\{CPU\}$. Les sets de coupes sont $\{CPU\}, \{DO_1, DO_2\}, \{DO_1, DCV_2\}, \{DO_1, SDV_2\}, \{DCV_1, DCV_2\}, \{DCV_1, DO_2\}, \{DCV_1, SDV_2\}$, $\{SDV_1, SDV_2\}, \{SDV_1, DO_2\}, \{SDV_1, DCV_2\}, \{PT_1, AI_2, PT_3\}$ et toute combinaison de PT et AI de taille 3 avec les trois indices différents.

3. On peut calculer la disponibilité en décomposant le système en 3 parties

$$A_s(t) = A_1(t)A_2(t)A_3(t)$$

La première, on modélise le 2oo4 par le RBD suivant :

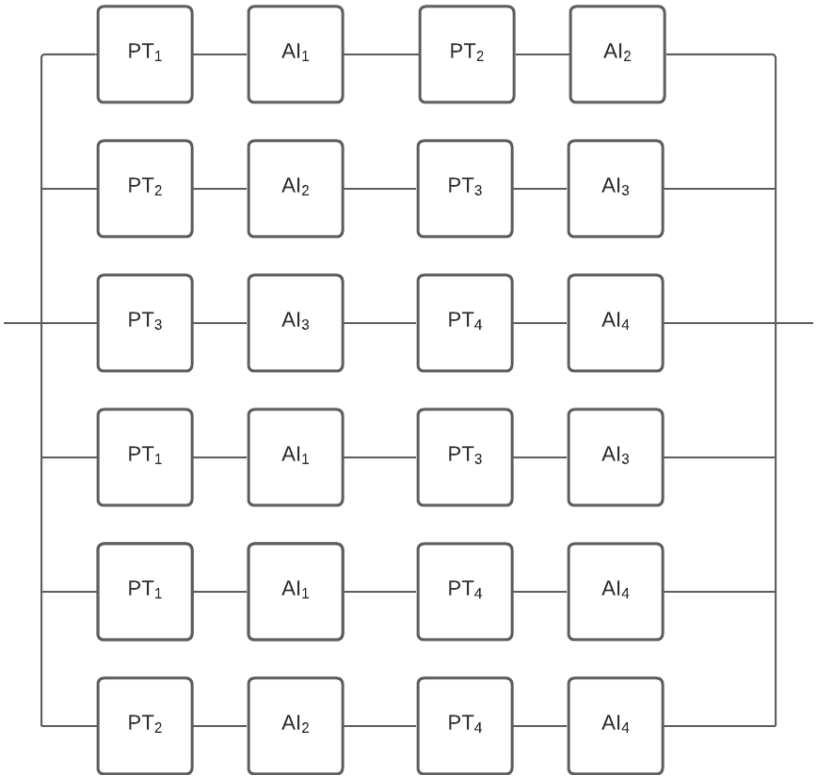

Figure 3 - 2oo4 PT et AI

Donc on retrouve la fonction de disponibilité :

Pour alléger les notations, on note $A_x(t) = A\_x$

On calcule d'abord la disponibilité pour chaque ligne dans le diagramme ci-dessus :

$$A_{ligne=1} = A_{PT1}A_{AI1}A_{PT2}A_{AI2}$$

$$A_{ligne=2} = A_{PT2}A_{AI2}A_{PT3}A_{AI3}$$

$$A_{ligne=3} = A_{PT3}A_{AI3}A_{PT4}A_{AI4}$$

$$A_{ligne=4} = A_{PT1}A_{AI1}A_{PT3}A_{AI3}$$

$$A_{ligne=5} = A_{PT1}A_{AI1}A_{PT4}A_{AI4}$$

$$A_{ligne=6} = A_{PT2}A_{AI2}A_{PT4}A_{AI4}$$

Donc on déduit que :

$$A_1(t) = 1 - \prod_{l=1}^{6}(1 - A_{ligne=l})$$

Ensuite, on a $A_2(t) = A_{CPU}(t)$

Et enfin

$$A_3(t) = 1 - (1 - A_{DO1}A_{DCV1}A_{SDV1})(1 - A_{DO2}A_{DCV2}A_{SDV2})$$

Donc

$$A_S(t) = \left[1 - \prod_{l=1}^{6}(1 - A_{ligne=l})\right] [A_{CPU}(t)] [1 - (1 - A_{DO1}A_{DCV1}A_{SDV1})(1 - A_{DO2}A_{DCV2}A_{SDV2})]$$

3 Problème 3

1. Il y a 4 états possibles : marche, arrêt, un AI défaillant, deux AI défaillants. En effet, si un des AI tombe en panne, le système continue à fonctionner, la maintenance va démarrer immédiatement soit si CPU est défaillant, soit si 3 AI sont défaillants, ce qui implique un arrêt de la production instantané, et un remplacement du SCM.

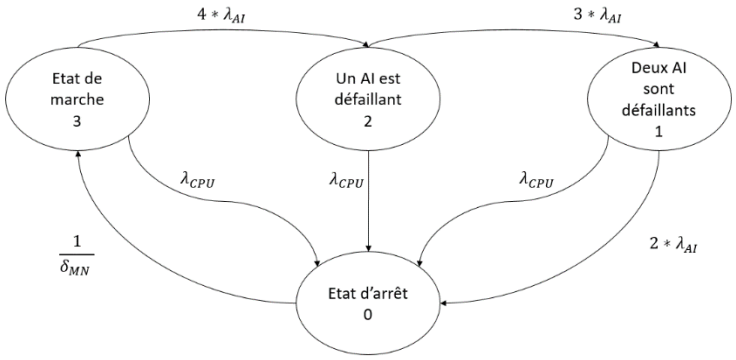

Figure 4 - Représentation de la chaine de Markov pour le système considéré

2. On pose le vecteur suivant :

$$P(t) = [P_3(t) \quad P_2(t) \quad P_1(t) \quad P_0(t)]$$

On a la matrice de transition

$$\mathbb{A} = \begin{pmatrix} -\lambda_{CPU} - 4\lambda_{AI} & 4\lambda_{AI} & 0 & \lambda_{CPU} \\ 0 & -3\lambda_{AI} - \lambda_{CPU} & 3\lambda_{AI} & \lambda_{CPU} \\ 0 & 0 & -\lambda_{CPU} - 2\lambda_{AI} & \lambda_{CPU} + 2 * \lambda_{AI} \\ \frac{1}{\delta_{MN}} & 0 & 0 & -\frac{1}{\delta_{MN}} \end{pmatrix}$$

On sait que $\dot{P}(t) = P(t)\,\mathbb{A} = [P_3(t) \quad P_2(t) \quad P_1(t) \quad P_0(t)]\,\mathbb{A}$

3. Probabilités en fonction du temps t

On considère que le système est en marche à l'instant t=0. En d'autres termes,

$$P(0) = [P_3(0) \quad P_2(0) \quad P_1(0) \quad P_0(0)] = [1 \quad 0 \quad 0 \quad 0]$$

La fonction odeint sur python permet de résoudre un système d'équations différentielles. Le code utilisé est dans le Jupyter Notebook. Cependant, on sait que $\sum_1^4 P_i(t) = 1$, c'est pourquoi on normalise le vecteur de solution afin qu'il ne comporte que des valeurs entre 0 et 1.

Cette solution trouvée ne correspond semble incorrecte, puisque quand l'horizon tend vers l'infini, les probabilités ne tendent pas vers les valeurs trouvées analytiquement.

Comme vu en cours on peut résoudre numériquement en posant :$P(t) = P(0)e^{\mathbb{A}t}$

Pour avoir la solution à un instant t, il suffit d'exécuter le code correspondant à la Q3 dans le notebook et de exécuter la commande Pt_df.iloc[t] et on récupère un vecteur avec les probabilités d'être dans chaque état à l'instant t.

4. Régime permanent

Pour trouver les probabilités en régime permanent, il suffit de résoudre le système suivant :

$$\dot{P}(t) = P(t)\,\mathbb{A} = 0$$

On choisit 4-1=3 équations (avec le plus de 0 possible pour simplifier) et on rajoute une dernière condition qui est de dire que : $\sum_1^4 P_i(t) = 1$, cela revient à considérer le nouveau système suivant :

$$[P_3(t) \quad P_2(t) \quad P_1(t) \quad P_0(t)] \begin{pmatrix} -\lambda_{CPU} - 4\lambda_{AI} & 4\lambda_{AI} & 0 & 1 \\ 0 & -3\lambda_{AI} - \lambda_{CPU} & 3\lambda_{AI} & 1 \\ 0 & 0 & -\lambda_{CPU} - 2\lambda_{AI} & 1 \\ \frac{1}{\delta_{MN}} & 0 & 0 & 1 \end{pmatrix} = (0\ 0\ 0\ 1)$$

On trouve que Les probabilités en régime permanent sont égales à :

$$P(\infty) = [P_3(\infty) \quad P_2(\infty) \quad P_1(\infty) \quad P_0(\infty)] = [0.62 \quad 0.28 \quad 0.10 \quad 10^{-6}]$$

5. La disponibilité du système

On définit la disponibilité comme étant

$$A_s(t) = P(X_s(t) = 1)$$

On considère que dans les états 1 et 2 le système reste fonctionnel donc $X_s(t) = 1$

Ainsi,

$$A_s(t) = P(X_s(t) = 1) = P\left((S = 3) \bigcup (S = 2) \bigcup (S = 1)\right) = P_3(t) + P_2(t) + P_1(t)$$

On calcule numériquement la disponibilité (cf code python) en sommant les 3 vecteurs de probabilité, ainsi, on obtient la disponibilité pour tout t dans l'horizon choisi.

Ensuite, pour le calcul de la proportion de temps moyen pendant laquelle le système est fonctionnel. On calcule d'abord le temps moyen de fonctionnement sur [0, T] :

$$T_m = \int_0^T A_s(u)\,du = \int_0^T [P_3(u) + P_2(u) + P_1(u)]\,du$$

On trouve que le temps moyen de fonctionnement sur un horizon de 1000 heures est de

$$T_m = 999.99\ h$$

Donc la proportion de temps moyen pendant laquelle le système est fonctionnel est :

$$p = \frac{T_m}{T} = \frac{999{,}99}{1000} = 0.99$$

4 Problème 4

1. Voici le diagramme présentant les états et transitions possibles :

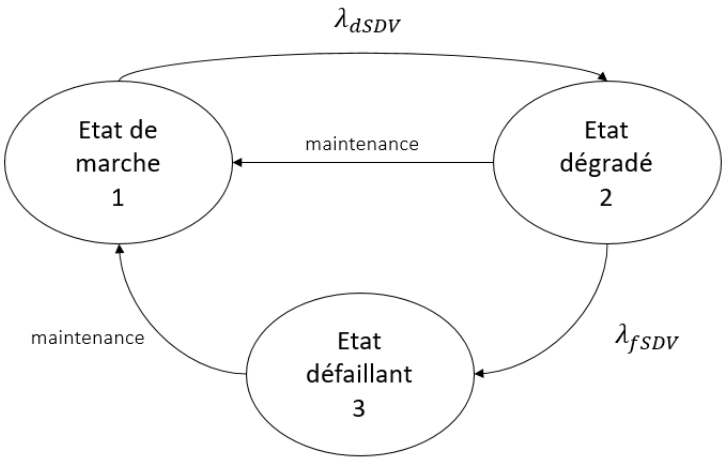

Figure 5 - Diagramme de Markov pour le composant SDV

2. On ne peut pas utiliser de processus de Markov classique car le temps entre un état dégradé ou défaillant et la maintenance ne dépend pas de quand survient la panne, il est prédéterminé et fixe.

3. Flow Chart

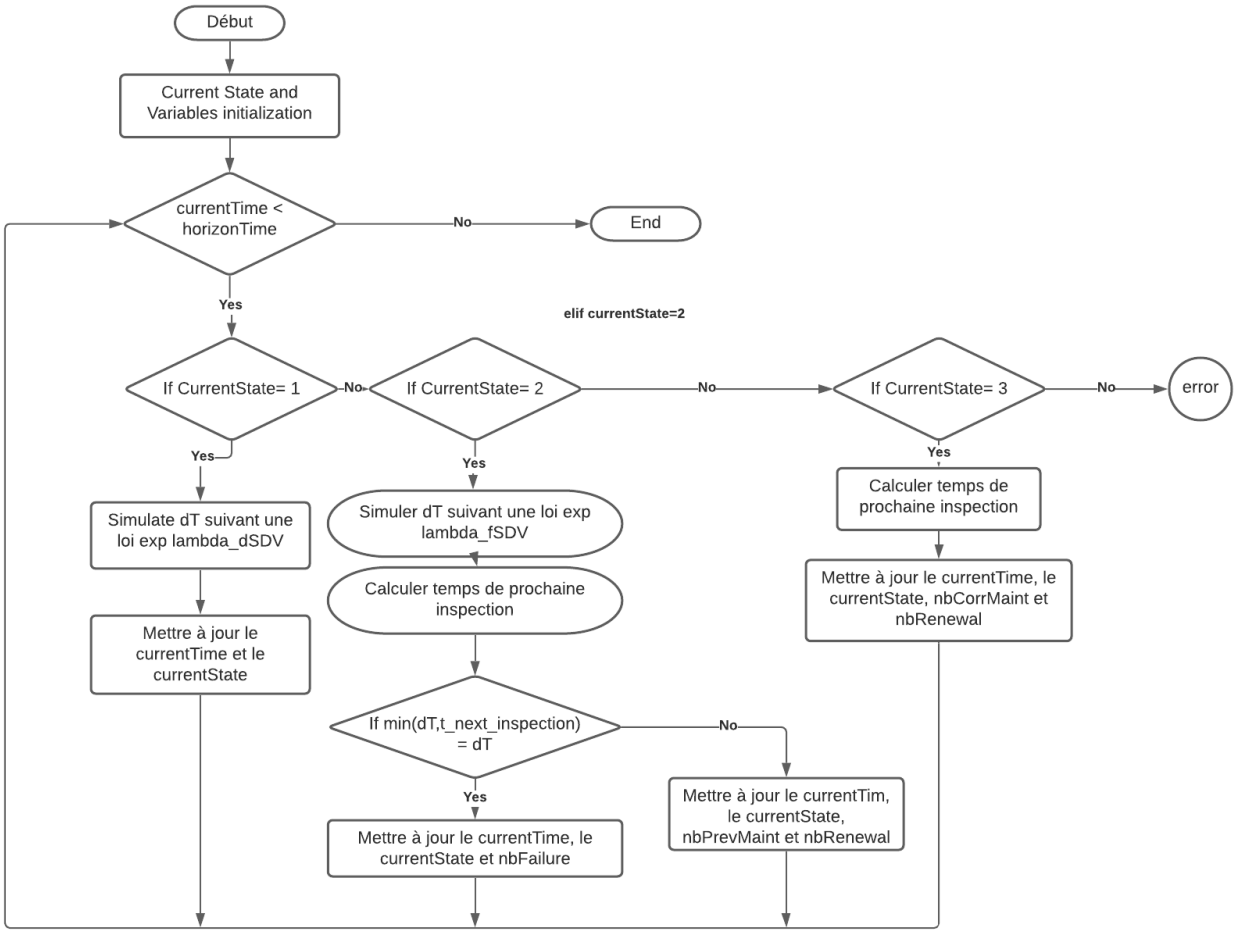

Figure 6 - Flow Chart du pseudo code pour le système

4. Bonus

On obtient à l'aide du script que la probabilité d'être dans l'état 1 à t=100 ans est de 0.98147, que la probabilité d'être dans l'état 2 est de 0.01709 et la probabilité d'être dans l'état 3 est de 0.00144. On peut donc en conclure que la stratégie de maintenance prédictive mis en place permet d'éviter drastiquement d'être dans un état défaillant.