# Rapport de projet OS13
# Analyse de politique de maintenance

TRAN Quoc Nhat Han & Adrien WARTELLE

13 janvier 2019

## Sommaire



**Résumé**

A partir de données liées au fonctionnement d'un système, nous allons déterminer un modèle approprié, lié à l'état de celui-ci : durée de vie ou niveau de défaillance. Nous allons ensuite, grâce au modèle, optimiser une politique de maintenance : basée sur l'âge ou conditionnelle.

# 1 Maintenance basée sur l'âge

## 1.1 Rappel

Considérons un système non maintenu. En l'observant, nous obtenons une liste des dates de panne, grâce auxquelles nous pouvons construire une politique de remplacement systématique basée sur l'âge : *Nous remettons à neuf le système lorsqu'il tombe en panne ou après une durée $t_0$ si il a survécu jusque là.*

Le but est de minimiser le coût moyen cumulé.

$$\mathbb{E}(C) = \frac{\mathbb{E}(C(S))}{\mathbb{E}(S)} \tag{1}$$

Où $S$ est la variable aléatoire représentant la date de remplacement et $C(S)$ est le coût de maintenance cumulé à l'instant $S$ (sachant que $C(S)$ vaut $c_c(= 1200)$ lors d'une maintenance corrective et $c_p(= 800)$ lors d'une maintenance préventive).

### 1.2 Modéliser la durée de vie du système

L'importation de données de `FailureTimes_5.csv` (l'annexe 3.1) montre des dates de pannes d'ordres grandement variées (300 à 27000) (l'annexe 3.2).

Mettre à l'exponentiel des valeurs extrêmes résultera en des valeurs nulles (si négatives) ou Inf (si positives) pour l'ordinateur, ce qui est indésirable. Alors nous devons forcément les réduire en les divisant par un scalaire `scale`, prenons par example 1000. (Figure 1)

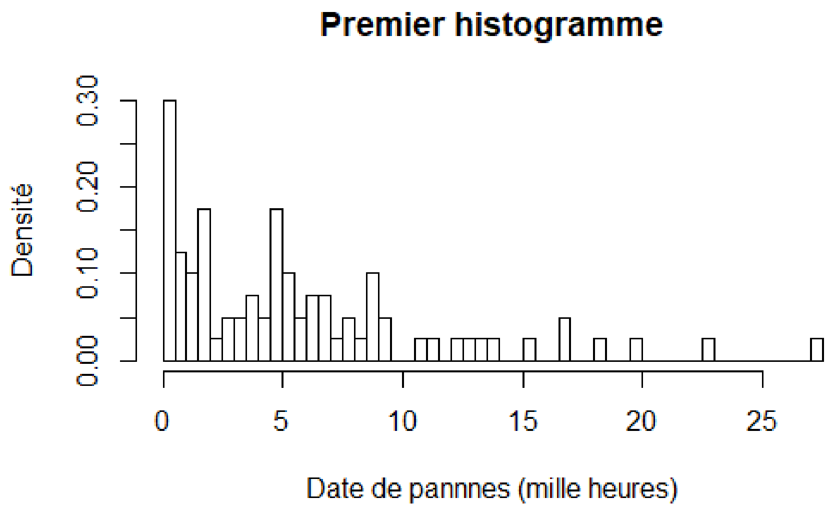



FIGURE 1 – Le premier histogramme de distribution de pannes

Nous pouvons remarquer que les valeurs sont positives (étant données que ce sont des temps) et que la distribution semble posséder deux parties importantes. En effet, les pannes se concentrent autour de 2 sommets, l'un à $[0; 0,5]$ et l'autre à $[4,5; 5]$. Ceci nous fait penser naturellement à un mixage de deux lois. Le premier sommet est suivi d'une pente forte, et la distribution locale de l'autre sommet a une forme de pic. Nous avons donc pensé estimer un mixage de loi *Exponentielle* et *Gamma* afin de modéliser les données. En effet la première partie correspondrait à une loi exponentielle tandis que la seconde à une loi gamma.

La fonction de densité du mixage avec le paramètre $\theta = (p_1, p_2, \lambda, \alpha, \beta)$ est donnée par :

$$\begin{aligned} f_\theta(x) &= p_1 f_1(x) + p_2 f_2(x) \\ &= p_1 \lambda e^{-\lambda x} + p_2 \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \end{aligned} \tag{2}$$

Où $f_1, f_2$ désignent réspectivement $exp(\lambda)$ et $\Gamma(\alpha, \beta)$ ; $p_1, p_2 > 0 : p_1 + p_2 = 1$.

Nous allons utiliser l'algorithme EM, la méthode la plus efficace pour estimer l'estimateur du maximum de vraisemblance (MLE) de mixage fini.

Soit $X$ la variable aléatoire de durée de vie du système. Soient $(x_1, ..., x_N)$ les observations.

Soit la matrice de probabilité d'appartenance $(\zeta_{ki})$ : $\zeta_{ki}$ vaut la probabilité que $x_i$ suive la loi $f_k$.

$$\zeta_{ki} = \frac{p_k f_k(x_i)}{p_1 f_1(x_i) + p_2 f_2(x_i)} \forall k = \overline{1,2}\ \forall i = \overline{1,N} \tag{3}$$

La fonction de vraisemblance :

$$\ln \Lambda = \sum_{i=1}^{N} \ln f_\theta(x_i) = \sum_{i=1}^{N} \ln\left(p_1 f_1(x_i) + p_2 f_2(x_i)\right) \tag{4}$$

Nous cherchons à maximiser $\ln \Lambda$ en la dérivant selon $\lambda, \alpha, \beta$.

Pour $\lambda$ :

$$\begin{aligned} \frac{\partial}{\partial \lambda} \ln \Lambda &= \sum_{i=1}^{N} \frac{p_1 e^{-\lambda x_i} - p_1 \lambda x_i e^{-\lambda x_i}}{p_1 f_1(x_i) + p_2 f_2(x_i)} \\ &= \sum_{i=1}^{N} \frac{p_1 f_1(x_i)}{p_1 f_1(x_i) + p_2 f_2(x_i)} \left(\frac{1}{\lambda} - x_i\right) \\ &= \frac{1}{\lambda} \sum_{i=1}^{N} \zeta_{1i} - \sum_{i=1}^{N} \zeta_{1i} x_i = 0 \\ \Leftrightarrow \lambda &= \frac{\sum\limits_{i=1}^{N} \zeta_{1i}}{\sum\limits_{i=1}^{N} \zeta_{1i} x_i} \end{aligned} \tag{5}$$

Pour $\beta$ :

$$
\begin{aligned}
\frac{\partial}{\partial \beta} \ln \Lambda &= \sum_{i=1}^{N} \frac{p_2 x_i^{\alpha-1}}{\Gamma(\alpha)} \frac{\alpha \beta^{\alpha-1} e^{-\beta x_i} - \beta^{\alpha} x_i e^{-\beta x_i}}{p_1 f_1(x_i) + p_2 f_2(x_i)} \\
&= \sum_{i=1}^{N} \frac{p_2 f_2(x_i)}{p_1 f_1(x_i) + p_2 f_2(x_i)} \left( \frac{\alpha}{\beta} - x_i \right) \\
&= \frac{\alpha}{\beta} \sum_{i=1}^{N} \zeta_{2i} - \sum_{i=1}^{N} \zeta_{2i} x_i = 0 \\
\Leftrightarrow \beta &= \alpha \frac{\sum\limits_{i=1}^{N} \zeta_{2i}}{\sum\limits_{i=1}^{N} \zeta_{2i} x_i} \qquad (6)
\end{aligned}
$$

Pour $\alpha$ :

$$
\begin{aligned}
\frac{\partial}{\partial \alpha} \ln \Lambda &= \sum_{i=1}^{N} \frac{p_2 e^{-\beta x_i}}{p_1 f_1(x) + p_2 f_2(x)} \left( \frac{\beta (\ln \beta + \ln x_i) (\beta x_i)^{\alpha-1}}{\Gamma(\alpha)} - \beta^{\alpha} x^{\alpha-1} \frac{\Psi(\alpha)}{\Gamma(\alpha)} \right) \\
&= \sum_{i=1}^{N} \frac{p_2 f_2(x_i)}{p_1 f_1(x) + p_2 f_2(x)} (\ln \beta + \ln x_i - \Psi(\alpha)) \\
&= \left( \sum_{i=1}^{N} \zeta_{2i} \right) \ln \beta + \sum_{i=1}^{N} \zeta_{2i} \ln x_i - \Psi(\alpha) \left( \sum_{i=1}^{N} \zeta_{2i} \right) = 0 \\
\Leftrightarrow 0 &= \ln \alpha + \ln \frac{\sum\limits_{i=1}^{N} \zeta_{2i}}{\sum\limits_{i=1}^{N} \zeta_{2i} x_i} + \frac{\sum\limits_{i=1}^{N} \zeta_{2i} \ln x_i}{\sum\limits_{i=1}^{N} \zeta_{2i}} - \Psi(\alpha) \quad (\ \beta \text{ substitué par } (6)) \\
\Leftrightarrow 0 &= \ln \alpha - \Psi(\alpha) - c
\end{aligned}
$$

Où $c = \ln \left( \frac{\sum_{i=1}^{N} \zeta_{2i} x_i}{\sum_{i=1}^{N} \zeta_{2i}} \right) - \frac{\sum_{i=1}^{N} \zeta_{2i} \ln(x_i)}{\sum_{i=1}^{N} \zeta_{2i}}$ ; $\Psi$ est la fonction digamma.

Selon la méthode de Newton-Rashphon, nous pouvons résoudre $\alpha$ numériquement avec ce formul itératif :

$$
\alpha_{r+1} = \alpha_r - \frac{\ln \alpha_r + \Psi(\alpha_r) - c}{\frac{1}{\alpha_r} - \Psi'(\alpha_r)}
$$

[1] propose un autre formule convergeant plus vite :

$$
\frac{1}{\alpha_{r+1}} = \frac{1}{\alpha_r} + \frac{\ln(\alpha_r) - \Psi(\alpha_r) - c}{\alpha_r^2 \left( \frac{1}{\alpha_r} - \Psi'(\alpha_r) \right)} \qquad (7)
$$

Avec $\Psi'$ la fonction trigamma. L'itération part avec $\alpha_0 = \frac{0.5}{c}$.
Au final, pour $p_k$ :

$$
p_k = \frac{\sum\limits_{i=1}^{N} \zeta_{ki}}{N} \forall k = \overline{1, 2} \qquad (8)
$$

Etant donné (3), (5), (6), (7) et (8), nous utilisons l'algorithme EM :

1. **Initialisation :** Choisir $\theta_0$.
2. **Etape E :** Evaluer $(\zeta_{ki})$ sachant $\theta_c$ en utilisant (3).
3. **Etape M :** Calculer $\theta_{c+1}$ à l'aide des équations (5), (6), (7) et (8).
   *Note :* Pour $\alpha$, l'itération se termine quand $|\alpha_{r+1} - \alpha_r| < \varepsilon_\alpha$ où $\varepsilon_\alpha$ est un réel positif fixé à l'initialisation.
4. **Evaluation :** Si $\|\theta_{c+1} - \theta_c\| < \varepsilon_\theta$ ($\varepsilon_\theta$ est un réel positif fixé à l'initialisation), l'algorithme s'arrête et $\theta = \theta_{c+1}$.
   Sinon, reviens à l'étape E avec $c = c + 1$.

Avant de lancer l'algorithme, nous avons essayé d'obtenir un ensemble de paramètres initiaux $\theta_0$ qui soient cohérents avec la distribution des données. Nous avons choisi :

$$(p_{1_0}, p_{2_0}, \lambda_0, \alpha_0, \beta_0) = (0.5; 0.5; 1; 10; 2)$$

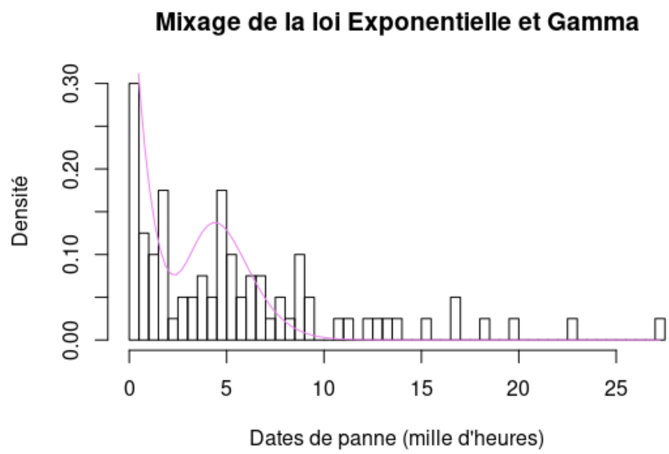



FIGURE 2 – Mixage de la loi Exponentielle et Gamma : **paramètres initiaux**

Après l'utilisation de l'algorithme EM, nous avons obtenu le résultat :

$$(p_1, p_2, \lambda, \alpha, \beta) = (0.2194518; 0.7805482; 1.56738; 1.665659; 0.2332427)$$

Nous avons tracé la fonction de densité $f_\theta$ trouvé (figure 3) et réalisé un test de Kolmogorov-Smirnov qui donne $p - value = 0,9663111$ signifiant $96,63\%$ de nous tromper si nous rejetons ce modèle. Nous l'acceptons alors, quoiqu'il ne génère pas 2 sommets comme la remarque initiale. Le code est trouvable à l'annexe 3.3.

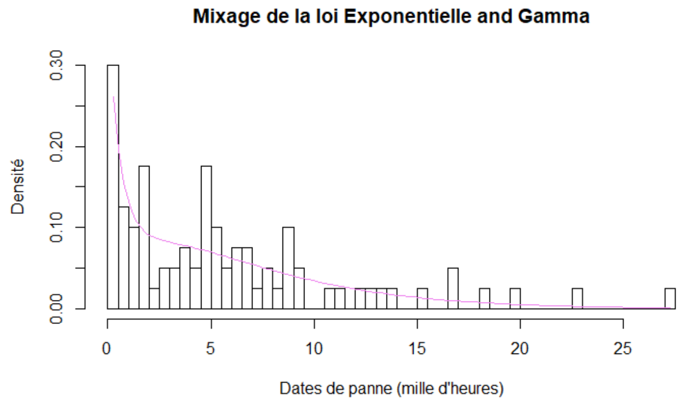



FIGURE 3 – Mixage de la loi Exponentielle et Gamma :**paramètres finaux**

### 1.3 La politique de maintenance basée sur l'âge

Avec la fonction $f_\theta$ trouvée, nous allons pouvoir déterminer une politique optimale (selon le modèle) basée sur l'âge.

Nous avons par définition : $S = \min(X, t_0)$.

Autrement dit, $S = X\mathbb{I}_{\{X<t_0\}} + t_0\mathbb{I}_{\{X\geqslant t_0\}}$.

Cela se traduit avec le coût : $C(S) = c_c\mathbb{I}_{\{X<t_0\}} + c_p\mathbb{I}_{\{X\geqslant t_0\}}$, avec $c_c, c_p$ les coûts de maintenances correctives et préventives respectivement.

Le coût moyen :

$$\begin{aligned}
\mathbb{E}(C(S)) &= c_c\mathbb{E}\left(\mathbb{I}_{\{X<t_0\}}\right) + c_p\mathbb{E}\left(\mathbb{I}_{\{X\geqslant t_0\}}\right) \\
&= c_c P(X < t_0) + c_p P(X \geqslant t_0) \\
&= c_c F_\theta(t_0) + c_p(1 - F_\theta(t_0)) \\
&= (c_c - c_p) F_\theta(t_0) + c_p \qquad (9)
\end{aligned}$$

La durée moyenne :

$$\begin{aligned}
\mathbb{E}(S) &= \mathbb{E}\left(X\mathbb{I}_{\{X<t_0\}}\right) + t_0\mathbb{E}\left(\mathbb{I}_{\{X\geqslant t_0\}}\right) \\
&= \int_0^{t_0} x f_\theta(x)\,dx + t_0 P(X \geqslant t_0) \\
&= xF_\theta(x)\big|_0^{t_0} - \int_0^{t_0} F_\theta(x)\,dx + t_0(1 - F_\theta(t_0)) \\
&= t_0 - \int_0^{t_0} F_\theta(x)\,dx \qquad (10)
\end{aligned}$$

De (9) et (10), nous détaillons le coût moyen sur une durée de temps (1) :

$$\mathbb{E}(C) = \frac{\mathbb{E}(C(S))}{\mathbb{E}(S)} = \frac{(c_c - c_p) F_\theta(t_0) + c_p}{t_0 - \int_0^{t_0} F_\theta(x)\,dx} \qquad (11)$$

L'annexe (3.4) montre comment chercher l'optimum numériquement. La valeur minimum est $t_0 = 27,29639$ (mille heures), correspondant à un coût moyen par durée de temps de $210,6402$ (euros / mille heures). Nous constatons que $t_0^{min}$ est très proche du maximum de durée de vie, indiquant que l'optimisation de $t_0$ est inutile car le système ne viellit que très peu.[1].

1. La loi de durée de vie trouvée est très proche d'une loi exponentielle car c'est celle-ci qui pré-pondère et cette loi implique un système toujours neuf (d'age 0), et donc sans besoin de maintenance basée sur l'âge. On peut facilement montrer que l'optimum de $t_0^*$ dans le cas exponentiel est infini.

## 2 Maintenance basée sur dégradation

### 2.1 Rappel

En observant de multiples systèmes identiques, nous effectuons des mesures de dégradation sur des intervalles de temps réguliers tout au long de leurs durée de vie.

La valeur limite de dégradation est $L = 20$. C'est-à-dire que lorsque le niveau de dégradation dépasse $L$, le système tombe en panne et nous ne pourrons plus le mesurer.

On souhaite mettre en place une politique de maintenance conditionnelle, basée sur un seuil $M$ inférieur à $L$ et l'intervalle de temps $\Delta T$ entre les inspections répétée. Appelons $X_t$ le niveau de dégradation à l'instant $t$ (instant d'une inspection) :

— Si $X_t < M$, nous laissons le système tel quel.
— Si $M \leq X_t < L$, un remplacement préventif est réalisé au coût $c_p$. Et puis $X_t$ est remis à 0.
— Si $X_t \geq L$, un remplacement correctif est réalisé au coût $c_c$. Et puis $X_t$ est remis à 0.

Le but est minimiser le coût moyen sur une durée de temps (1) en choisissant bien le seuil $M$ et l'intervalle d'inspection $\Delta T$.

### 2.2 Modéliser la dégradation du système

Avec les données de `DegradLevel_2.csv`, nous traçons leurs processus de dégradation (figure (4)). (Les annexes (3.5) et (3.6))

Comme les temps d'inspection sont de l'ordre du millier, il faudrait les diviser par un scalaire (par example, $scale = 1000$) afin d'assurer la précision des calculs numériques.

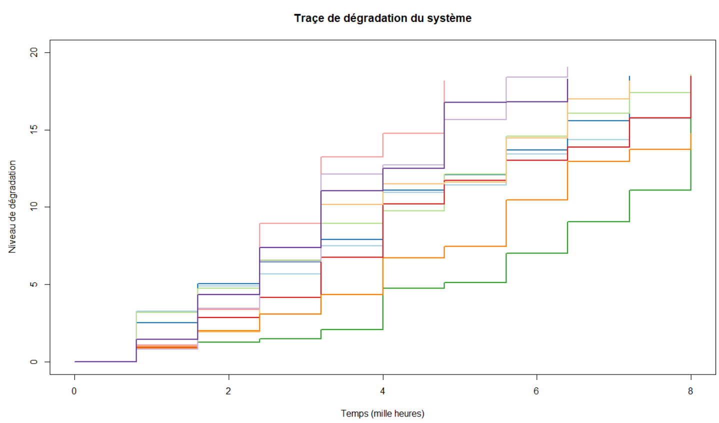



FIGURE 4 – Traçe de dégradation du système

Nous voyons que les accroissements sont positifs, suggérant un modèle de processus Gamma.

Soit $X(t)$ la variable aléatoire de dégradation du système. Supposons que $X(t) - X(s) \sim \Gamma(a(t-s), b) \, \forall t > s > 0$. Nous allons estimer les paramètres $a, b$

en modélisant la distribution des incréments entre deux moments successifs.

Fixons $t - s = \delta = 0.8$, car les mesures données sont effectués au bout de chaque intervalle de 0.8.

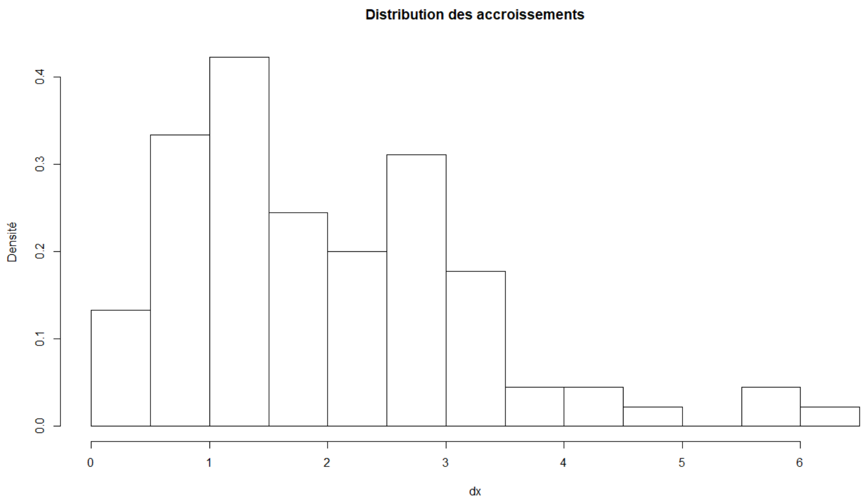



FIGURE 5 – Histogramme des incréments de l'intervalle $\delta = 0.8$

A l'aide du librairie MASS : $(a; b) = \left(\frac{\alpha}{\delta}; \beta\right) = (2.843101; 1.140354)$ où $(\alpha, \beta)$ sont les paramètres estimés par MASS. Le code est mis à l'annexe (3.7).

Un test rapide de Kolmogorov-Smirnov nous donne $p - value = 0.8651934$, indiquant le modèle est largement acceptable.

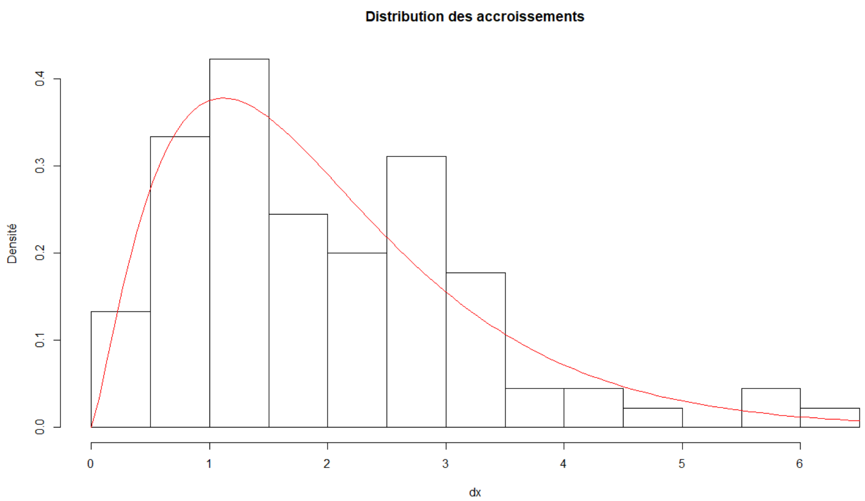



FIGURE 6 – Histogramme et la courbe de densité estimé des incréments de l'intervalle $\delta = 0.8$

D'autant plus, si nous refaisons les calculs ci-dessus avec $\delta = 1.6, 2.4, 3.2, etc.$, nous voyons les valeurs de $a$ et $b$ ne varient pas trop. Alors, nous choisissons le couple $(a, b)$ avec $p$ le plus grand. ($\delta$ trop grand réduira nombreux de données, résultant une perte importante de précision)

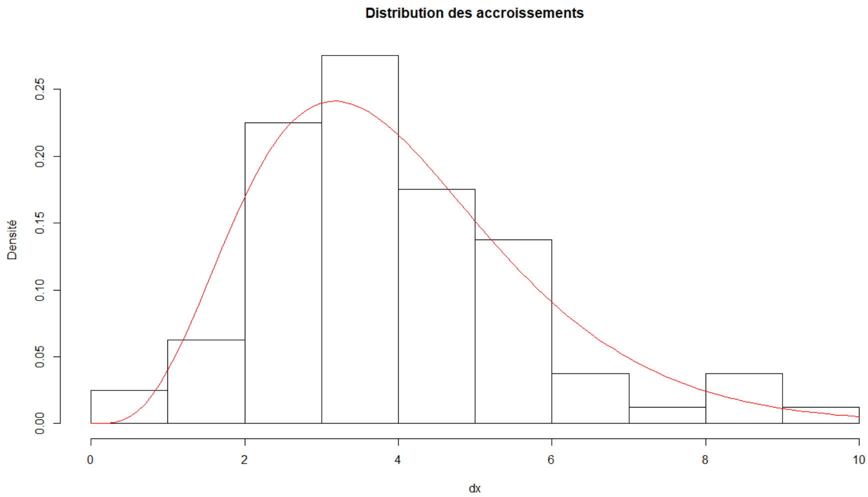



FIGURE 7 – Histogramme et la courbe de densité estimé des incréments de l'intervalle $\delta = 1.6$

| $\delta$ | $a$ | $b$ | $p$ |
|---|---|---|---|
| 0.8 | 2.8431009 | 1.1403540 | 0.8651934 |
| 1.6 | 3.0207719 | 1.2091646 | 0.9919939 |
| 2.4 | 3.0187497 | 1.1907719 | 0.1279252 |
| 3.2 | 2.997763 | 1.185547 | 0.300610 |
| 4.0 | 3.5036580 | 1.4061063 | 0.6307951 |

TABLE 1 – Calcul $(a, b)$ avec différents $\delta$

Au final, nous choisissons $X(t) - X(s) \sim \Gamma(a(t-s), b)\ \forall t > s > 0$, $\delta = 1.6$ et $(a, b) = (3.0207719; 1.2091646)$.

## 2.3 La politique de maintenance basée sur dégradation

Cette politique, outre que $c_c = 1200$ et $c_p = 800$, introduit ainsi le coût d'inspection répétive $c_i = 10$.

Soit $S$ la variable aléatoire représentant la date de remplacement ; $C(S)$ est le coût de maintenance cumulé à l'instant $S$, et $N(S)$ le nombre d'inspections depuis la dernière remplacement jusqu'à $S$.

$$C(S) = c_i N(S) + c_p \mathbb{I}_{\{L > X_{N(S)} \geq M\}} + c_c \mathbb{I}_{\{X_{N(S)} \geq L\}} \tag{12}$$

Posons $F_t = \Gamma(at, b)$ et les instants d'inspections $t_1, t_2, ..., t_{N(S)}$ avec $t_{j+1} - t_j = \Delta T$. Puisque $t_0 = 0$, nous aurons $t_k = k\Delta T$ et $X_{t_k} \sim F_{k\Delta T}$.

Comme le calcul analytique de $C(S)$ et $N(S)$ devient grossier (l'annexe (3.8)), nous passons à la méthode Monté Carlo, qui nous permet d'estimer la vraie valeur en répétant un nombre suffisant de simulations.

— L'algorithme (1) calcule la durée de vie d'une simulation.
— L'algorithme (2) calcule le coût d'une simulation.
— L'algorithme (3) simule de multiples processus et retourne une valeur approximative de $\mathbb{E}(C)$.
— L'algorithme (4) trouve l'optimum sur une zone de recherche.

**Algorithme 1 :** Mesurer la durée de vie d'un processus

**Entrées :** $p$ le processus
$\Delta T$ l'intervalle d'inspection
$M$ le seuil de maintenance préventive
**Output :** $S$ la durée de vie

*len* $\leftarrow$ longeur de $p$;
**si** $p_{len} < M$ **alors**
    `// La simulation n'est pas suffisante longue`
    **retourner** *0*
**sinon**
    $n_s \leftarrow$ le dernier index de $p$ tel que $p_{n_s} < M$;
    **retourner** $n_s + 1$
**fin**

**Algorithme 2 :** Mesurer le coût de maintenance d'un processus

**Entrées :** $p$ le processus
$M$ le seuil
**Output :** $C(S)$ le coût de maintenance

*cout* $\leftarrow 0$;
*len* $\leftarrow$ longeur de $p$;
**si** $p_{len} \geq M$ **alors**
    $n_s \leftarrow$ le dernier index de $p$ tel que $p_{n_s} < M$;
    `//` $n_s$ `correspond le nombre d'inspection`
    $cout \leftarrow cout + c_i * n_s$;
    `// A l'inspection suivante`
    **si** $p_{n_s+1} \geq L$ **alors**
        `// Le système est tombé en panne`
        $cout \leftarrow cout + c_c$;
    **sinon**
        `// Le système fonctionne encore`
        $cout \leftarrow cout + c_p$;
    **fin**
**fin**
`// Le coût est zéro si le niveau de dégradation ne dépasse pas encore` $M$
**retourner** *cout*

**Algorithme 3 :** Simuler le processus et calculer $\widehat{\mathbb{E}(C)}$

**Entrées :** $\Delta T$ l'intervalle d'inspection
$M$ le seuil
$nSim$ le nombre de simulation
$nStep$ le nombre d'inspection maximale

**Output :** $\widehat{\mathbb{E}(C)}$ la valeur approximative

Générer la matrice de l'incréments $simStep$ de taille $nSim \times nStep$ aux valeurs aléatoires selon la loi $\Gamma(a\Delta T, b)$;
Calculer la matrice de processus $simProc$ de taille $nSim \times nStep$ en faisant la somme cumulative de $simStep$ horizontallement;
$simTime \leftarrow$ les durées de vie de processus en appliquant l'algorithme (1);
$simCost \leftarrow$ les coûts de processus en appliquant l'algorithme (2);
$meanSimCost \leftarrow$ la moyenne de tous les coûts non nuls de $simCost$;
$meanSimTime \leftarrow$ la moyenne de durée de vie non nuls de $simTime$;
$meanCostTime \leftarrow \frac{meanSimCost}{meanSimTime}$;
**si** *meanSimCost est* `NaN` **ou** *meanSimTime est* `NaN` **alors**
    $meanCostTime \leftarrow MAX\_DOUBLE$;
**fin**
**retourner** $meanCostTime$

**Algorithme 4 :** Optimiser la politique de maintenance basée sur dégradation

**Entrées :** $[L_{\Delta T}, U_{\Delta T}]$ l'intervalle de recherche de $\Delta T$
$[L_M, U_M]$ l'intervalle de recherche de $M$
$\tau$ la tolérance
$d_{\Delta T}$ le nombre de points à évaluer pour $\Delta T$
$d_M$ le nombre de points à évaluer pour $M$
$e_{\Delta T}$ le ratio de réduction sur l'intervalle de $\Delta T$
$e_M$ le ratio de réduction sur l'intervalle de $M$

**Output :** $\Delta T^*$ l'intervalle d'inspection optimal
$M^*$ le seuil optimal
$o^*$ la valeur objective optimale

`// Suivre le principe de diachotomie`
$\Delta T^* \leftarrow 0$;
$M^* \leftarrow 0$;
$o^* \leftarrow 0$;
**tant que** $\left(\frac{L_{\Delta T} - U_{\Delta T}}{d_{\Delta T}}\right)^2 + \left(\frac{L_M - U_M}{d_M}\right)^2 \geqslant \tau^2$ **faire**
  `// Evaluer`
  $I_{\Delta T} \leftarrow$ l'ensemble de $d_{\Delta T}$ points égaux-distances de $[L_{\Delta T}, U_{\Delta T}]$;
  $I_M \leftarrow$ l'ensemble de $d_M$ points égaux-distances de $[L_M, U_M]$;
  $Z \leftarrow$ résultats de l'application de l'algorithme (3) pour chaque point sur la grille $I_{\Delta T} \times I_M$;
  `// Chercher le minimum`
  $(i^*, j^*) \leftarrow$ l'indice du minimum de $Z$;
  $\Delta T^* \leftarrow$ l'élément $i^*$-ième de $I_{\Delta T}$;
  $M^* \leftarrow$ l'élément $j^*$-ième de $I_M$;
  $o^* \leftarrow Z_{i^*, j^*}$;
  `// Ajuster la nouvelle zone de recherche autour du minimum actuel`
  $l_{\Delta T} \leftarrow L_{\Delta T} - U_{\Delta T}$;
  $l_M \leftarrow L_M - U_M$;
  `// La nouvelle zone ne doit pas déborder celle précédente`
  $L_{\Delta T} \leftarrow \min\left(\Delta T^* + \frac{l_{\Delta T}}{e_{\Delta T}}, L_{\Delta T}\right)$;
  $U_{\Delta T} \leftarrow \max\left(\Delta T^* - \frac{l_{\Delta T}}{e_{\Delta T}}, U_{\Delta T}\right)$;
  $L_M \leftarrow \min\left(M^* + \frac{l_M}{e_M}, L_M\right)$;
  $U_M \leftarrow \max\left(M^* - \frac{l_M}{e_M}, U_M\right)$;
**fin**
**retourner** $(\Delta T^*, M^*, o^*)$

Puisque l'algorithme (3) utilise du calcul matriciel pour accélérer le traitement, $nStep$ doit être fixé judicieusement pour ne pas avoir "trop" de simulations où $X_{t_{nStep}} < M$ (ce qui va forcer les valeurs de $C(S)$ et $S$ à zéro selon les algorithmes (1) et (2)). Soit $\alpha$ la probabilité de cet événement.

$$\begin{aligned} P\left(X_{t_{nStep}} < M\right) &= \alpha \\ \Leftrightarrow F_{nStep*\Delta T}(M) &= \alpha \\ \Leftrightarrow \Gamma\left(a * nStep * \Delta T, b\right)(M) &= \alpha \end{aligned}$$

Nous testons quelques valeurs de $F$. Comme $F$ est croissant, nous l'évaluons directement à $M = 20$.

| $nStep * \Delta T$ | $\Gamma\left(a * nStep * \Delta T, b\right)(20)$ |
|---|---|
| 8 | 0.5284405 |
| 10 | 0.1319799 |
| 12 | 0.01300913 |
| 13 | 0.002925565 |
| 14 | 0.000533971 |

Table 2 – Estimation de $nStep * \Delta T$

La fonction distribution décroît selon $nStep*\Delta T$. Et pourtant, $\alpha \leq 1\%$ nous suffit. D'autant plus, la traitement ralentit remarquablement lorsque $nStep$ est grand. C'est pourquoi, basé sur la table (2), nous emploirons

$$nStep = \left\lfloor \frac{13}{\Delta T} \right\rfloor$$

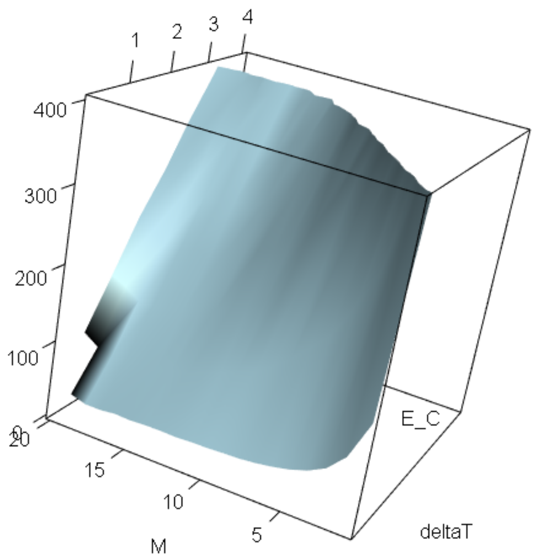
Le code implémenté se situe à l'annexe 3.9.

Bien que le résultat n'est pas stable à cause de l'approche Monté Carlo, nous pouvons arriver à une valeur approximative après plusieurs itérations de l'algorithme (4) sur l'intervalle $[0.001, 4] \times [0, 20]$ :

$$\begin{aligned} (\Delta T^*, M^*, o^*) = (&0.001007627 \text{ (mille heures)}, \\ &18.58027, \\ &10.10541 \text{ (euros/mille heures)}) \end{aligned}$$

Ce résultat nous semble logique car :
- $\Delta T^*$ faible (grâce à coût $c_i$ qui est faible) nous permet d'observer mieux et juste à temps le système.
- $M^*$ proche de $L$ va "remplacer" la plupart des coûts $c_c$ par $c_p$ pour économiser.

FIGURE 8 – Représentation graphique de $E(C)$



# 3 Annexe

## 3.1 L'importation de données de pannes

```
pannes = read.csv(
    file = "FailureTimes_5.csv",
    header = TRUE,
    sep = ",",
    dec = ".",
    colClasses = c("NULL", NA)
)
scale = 1000
data = pannes$Heures / scale
N = length(data)
```

## 3.2 Le premier histogramme de distribution de pannes

```
hist(
    data,
    breaks = 40,
    probability = TRUE,
    xlab = "Date de pannnes (mille heures)",
    ylab = "Densité",
    main = "Premier histogramme"
)
```

## 3.3 Estimer le mixage de la loi Exponentielle et Gamma

```
# Fitting mixture of Exp and Gamma
# Algorithm EM
# Initialisation
k = 2 # number of components
p = c(0.5, 0.5)
lambda = 1
alpha = 10
beta = 2
f = list(
    '1' = function(x) {
        dexp(x, rate = lambda)
    },
    '2' = function(x) {
        dgamma(x, shape = alpha, rate = beta)
    }
)
# Illustration initial
f_theta = function(x) {
  p[[1]] * f[[1]](x) + p[[2]] * f[[2]](x)
}
h_theta = hist(
  data,
  breaks = 40,
  probability = TRUE,
  main = "Mixage de la loi Exponentielle et Gamma",
  xlab = "Dates de panne (mille d'heures)",
  ylab = "Densité"
)
curve(
  f_theta(x),
  add = TRUE,
  col = "violet",
  from = min(h_theta$mids),
  to = max(h_theta$mids)
)
epsilon = list(
    alpha = 1e-4,
    theta = 1e-4
```

```r
)
zeta = matrix(
    0,
    nrow = k,
    ncol = N
)
# Norm
normVec = function(x) sqrt(sum(x^2))
# New value
p_new = p
alpha_new = alpha
beta_new = beta
lambda_new = lambda
repeat {
    ## E Step
    # Calculate each proba
    for (l in 1:k) {
        zeta[l,] = p[[l]] * f[[l]](data)
    }
    # Normalize proba
    zeta = t(t(zeta) / rowSums(t(zeta)))
    ## M step
    # Lambda
    lambda_new = sum(zeta[1,]) / sum(zeta[1,] * data)
    # Alpha
    c = log(sum(zeta[2,] * data) / sum(zeta[2,])) - sum(zeta[2,] * log(data)) / sum(zeta[2,])
    alpha_new = 0.5 / c
    alpha_temp = 0
    repeat {
        alpha_temp = 1 / (1 / alpha_new + (log(alpha_new) - digamma(alpha_new) - c) / (alpha_new^2 * (1 / alpha_new - trigamma(alpha_new))))
        if (abs(alpha_temp - alpha_new) < epsilon$alpha) {
            break
        } else {
            alpha_new = alpha_temp
        }
    }
    alpha_new = alpha_temp
    # Beta
    beta_new = alpha_new * sum(zeta[2,]) / sum(zeta[2,] * data)
    # P
    for (l in 1:k) {
        p_new[[l]] = mean(zeta[l,])
    }
    ## Evaluation
    if (normVec(c(alpha, beta, lambda, p[[1]], p[[2]]) - c(alpha_new, beta_new, lambda_new, p_new[[1]], p_new[[2]])) < epsilon$theta) {
        break
    } else {
        alpha = alpha_new
        beta = beta_new
        lambda = lambda_new
        p = p_new
    }
}
# Final value update
alpha = alpha_new
beta = beta_new
lambda = lambda_new
p = p_new
# Illustration final
f_theta = function(x) {
  p[[1]] * f[[1]](x) + p[[2]] * f[[2]](x)
}
h_theta = hist(
  data,
  breaks = 40,
  probability = TRUE,
  main = "Mixage de la loi Exponentielle et Gamma",
  xlab = "Dates de panne (mille d'heures)",
  ylab = "Densité"
)
curve(
```

```
    f_theta(x),
    add = TRUE,
    col = "violet",
    from = min(h_theta$mids),
    to = max(h_theta$mids)
)
# Kolmogorov-Smirnov test
F_theta = function(x) {
    p[[1]] * pexp(x, rate = lambda) + p[[2]] * pgamma(x, shape = alpha, rate
        = beta)
}
test = ks.test(data, F_theta, exact = TRUE)
```

### 3.4 Optimiser le coût moyen sur une durée de temps

```
# Finding optimal t_0
# Given F_theta
c_c = 1200
c_p = 800
E_C_S = function(x) {
    (c_c - c_p) * F_theta(x) + c_p
}
E_S = function(x) {
    x - integrate(F_theta,0,x)$value
}
E_C = function(x) {
    E_C_S(x) / E_S(x)
}
o = optimize(
    E_C,
    c(min(data),max(data)),
    tol = 1e-5
)
d = seq(
        min(data),
        max(data),
        0.01
    )
plot(
    d,
    lapply(
        d,
        E_C
    ),
    main = "Coût moyenne sur une durée de temps",
    xlab = "t_0",
    ylab = "",
    type = "l"
)
```

### 3.5 Importer les valeurs de dégradation

```
# Import degradation
table = read.csv(
    file = "DegradLevel_2.csv",
    header = TRUE,
    sep = ",",
    dec = "."
)
scale = 1000
time = c(0, table$Temps / scale)
nbProcess = length(table) - 2
process = matrix(
    ,
    nrow = nbProcess,
    ncol = 1 + length(table[[3]])
)
for (i in 1:nbProcess) {
```

```
    process[i,] = c(0, table[[i + 2]]) # degrad = 0 at t = 0
}
```

## 3.6 Premiers traçes de dégradation

```
# Plot process curves
L = 20
# Colormap
library(RColorBrewer)
color = brewer.pal(nbProcess, "Paired")
# First process
plot(
    NULL,
    type = "n",
    main = "Traçe de dégradation du système",
    xlim = c(min(time), max(time)),
    xlab = "Temps (mille heures)",
    ylim = c(0, L),
    ylab = "Niveau de dégradation"
)
for (i in 1:nbProcess) {
    lines(
        x = time,
        y = process[i,],
        type = "s",
        col = color[[i]],
        lwd = 2
    )
}
```

## 3.7 Estimation de paramètres de dégradations

```
## Estimate parameters for process
lag = 2
delta = 0.8 * lag
d = t(diff(t(process), lag))
# Concatenate into one vector
increments = vector(
    mode = "numeric",
    length = length(d)
)
n = dim(d)[[1]]
l = dim(d)[[2]]
for (i in 1:n) {
    increments[((i - 1) * l + 1):(i * l)] = d[i,]
}
# Filter out NA values
increments = increments[!is.na(increments)]
# Histogram
hiso_degrad = hist(
    increments,
    breaks = 10,
    probability = TRUE,
    main = "Distribution des accroissements",
    xlab = "dx",
    ylab = "Densité"
)
# Estimate gamma distribution
library(MASS)
estim = fitdistr(
    increments,
    dgamma,
    list(
        shape = 1,
        rate = 1
    )
)
# Draw estimated density
```

```
a = estim$estimate[[1]] / delta
b = estim$estimate[[2]]
curve(
    dgamma(x, a * delta ,b),
    add = TRUE,
    col = "red"
)
# Kolmogorov-Smirnov test
test = ks.test(increments, "pgamma", a * delta, b)
c(delta, a, b, test$p.value) # Print values
# PDF
f = function(x, t) {
    dgamma(x, shape = a * t, rate = b)
}
# CDF
F = function(x, t) {
    pgamma(x, shape = a * t, rate = b)
}
# Generator
rG = function(n, t) {
    rgamma(n, shape = a *t, rate = b)
}
```

## 3.8 Calcul analytique de maintenance conditionnelle

Rappelons que $X_{t_k} \sim F_{k\Delta t}$ et (12) :

$$C(S) = c_i N(S) + c_p \mathbb{I}_{\{L > X_{N(S)} \geqslant M\}} + c_c \mathbb{I}_{\{X_{N(S)} \geqslant L\}}$$

D'où

$$\begin{aligned}\mathbb{E}(C(S)) = c_i \mathbb{E}(N(S)) + c_p \mathbb{E}\left(\mathbb{I}_{\left\{L > X_{t_{N(S)}} \geqslant M\right\}}\right) \\ + c_c \mathbb{E}\left(\mathbb{I}_{\left\{X_{t_{N(S)}} \geqslant L\right\}}\right)\end{aligned}$$

Détaillons le nombre moyen d'inspection $\mathbb{E}(N(S))$ :

$$\mathbb{E}(N(S)) = \sum_{k=1}^{\infty} kP(N(S) = k) = P(N(S) = 1) + \sum_{k=2}^{\infty} kP(N(S) = k)$$

Or

$$\begin{aligned}
P(N(S) = 1) &= P(X_{t_1} \geqslant M) \\
&= 1 - F_{\Delta T}(M) \\
P(N(S) = k) &= P\left(X_{t_k} \geqslant M \wedge X_{t_{k-1}} < M\right) \\
&= P\left(\Delta X \geqslant M - X_{t_{k-1}} \wedge X_{t_{k-1}} < M\right) \\
&= \int_0^M P(\Delta X \geqslant M - \xi) P\left(X_{t_{k-1}} = \xi\right) d\xi \\
&= \int_0^M (1 - F_{\Delta T}(M - \xi)) f_{(k-1)\Delta T}(\xi) d\xi \\
&= \int_0^M f_{(k-1)\Delta T}(\xi) d\xi - \int_0^M F_{\Delta T}(M - \xi) f_{(k-1)\Delta T}(\xi) d\xi \\
&= F_{(k-1)\Delta T}(M) - \int_0^M F_{\Delta T}(M - \xi) f_{(k-1)\Delta T}(\xi) d\xi
\end{aligned}$$

Par conséquence,

$$\mathbb{E}\left(N\left(S\right)\right) = 1 - F_{\Delta T}\left(M\right)$$
$$+ \sum_{k=2}^{\infty} k \left( F_{(k-1)\Delta T}\left(M\right) - \int_0^M F_{\Delta T}\left(M - \xi\right) f_{(k-1)\Delta T}\left(\xi\right) d\xi \right) \qquad (13)$$

En suite, nous exprimons la probabilité de maintenance préventive :

$$\mathbb{E}\left(\mathbb{I}_{\left\{L > X_{t_{N(S)}} \geqslant M\right\}}\right) = \sum_{k=1}^{\infty} P\left(L > X_{t_k} \geqslant M | X_{t_{k-1}} < M\right)$$
$$k = 1 : \; P\left(L > X_{t_1} \geqslant M\right) = F_{\Delta T}\left(L\right) - F_{\Delta T}\left(M\right)$$
$$\forall k \geqslant 2 :$$
$$P\left(L > X_{t_k} \geqslant M | X_{t_{k-1}} < M\right)$$
$$= \frac{P\left(L > X_{t_k} \geqslant M \wedge X_{t_{k-1}} < M\right)}{P\left(X_{t_{k-1}} < M\right)}$$
$$= \frac{P\left(X_{t_{k-1}} < M \wedge \Delta X \geqslant M - X_{t_{k-1}} \wedge \Delta X < L - X_{t_{k-1}}\right)}{P\left(X_{t_{k-1}} < M\right)}$$
$$= \frac{\int_0^M P\left(\Delta X \geqslant M - \xi \wedge \Delta X < L - \xi\right) P\left(X_{t_{k-1}} = \xi\right) d\xi}{F_{(k-1)\Delta T}\left(M\right)}$$
$$= \frac{\int_0^M \left(F_{\Delta T}\left(L - \xi\right) - F_{\Delta T}\left(M - \xi\right)\right) f_{(k-1)\Delta T}\left(\xi\right) d\xi}{F_{(k-1)\Delta T}\left(M\right)}$$

Ceci induit que :

$$\mathbb{E}\left(\mathbb{I}_{\left\{L > X_{t_{N(S)}} \geqslant M\right\}}\right) = F_{\Delta T}\left(L\right) - F_{\Delta T}\left(M\right)$$
$$+ \sum_{k=2}^{\infty} \frac{\int_0^M \left(F_{\Delta T}\left(L - \xi\right) - F_{\Delta T}\left(M - \xi\right)\right) f_{(k-1)\Delta T}\left(\xi\right) d\xi}{F_{(k-1)\Delta T}\left(M\right)} \qquad (14)$$

Puis, de même manière, la probabilité de maintenance corrective sera :

$$\mathbb{E}\left(\mathbb{I}_{\left\{X_{t_{N(S)}} \geqslant L\right\}}\right) = \sum_{k=1}^{\infty} P\left(X_{t_k} \geqslant L | X_{t_{k-1}} < M\right)$$
$$k = 1 : \; P\left(X_{t_1} \geqslant L\right) = F_{\Delta T}\left(L\right)$$
$$\forall k \geqslant 2 :$$
$$P\left(X_{t_k} \geqslant L | X_{t_{k-1}} < M\right)$$
$$= \frac{P\left(X_{t_k} \geqslant L \wedge X_{t_{k-1}} < M\right)}{P\left(X_{t_{k-1}} < M\right)}$$
$$= \frac{P\left(\Delta X \geqslant L - X_{t_{k-1}} \wedge X_{t_{k-1}} < M\right)}{P\left(X_{t_{k-1}} < M\right)}$$
$$= \frac{\int_0^M P\left(\Delta X \geqslant L - \xi\right) P\left(X_{t_{k-1}} = \xi\right) d\xi}{P\left(X_{t_{k-1}} < M\right)}$$
$$= \frac{\int_0^M \left(1 - F_{\Delta T}\left(L - \xi\right)\right) f_{(k-1)\Delta T}\left(\xi\right) d\xi}{F_{(k-1)\Delta T}\left(M\right)}$$

D'où :

$$\mathbb{E}\left(\mathbb{I}_{\left\{X_{t_{N(S)}}\geqslant L\right\}}\right) = F_{\Delta T}\left(L\right) + \sum_{k=2}^{\infty} \frac{\int_0^M \left(1 - F_{\Delta T}\left(L-\xi\right)\right) f_{(k-1)\Delta T}\left(\xi\right) d\xi}{F_{(k-1)\Delta T}\left(M\right)} \tag{15}$$

Attention, parce que nous considérons $S$ l'instant de la dernière inspection :

$$S = t_{N(S)} = N\left(S\right)\Delta T \Rightarrow \mathbb{E}\left(S\right) = \mathbb{E}\left(N\left(S\right)\right)\Delta T$$

Alors :

$$\begin{aligned}
\mathbb{E}\left(C\right) &= \frac{\mathbb{E}\left(C\left(S\right)\right)}{\mathbb{E}\left(S\right)} \\
&= \frac{c_i\mathbb{E}\left(N\left(S\right)\right) + c_p\mathbb{E}\left(\mathbb{I}_{\left\{L>X_{t_{N(S)}}\geqslant M\right\}}\right) + c_c\mathbb{E}\left(\mathbb{I}_{\left\{X_{t_{N(S)}}\geqslant L\right\}}\right)}{\mathbb{E}\left(N\left(S\right)\right)\Delta T} \\
&= \frac{c_i}{\Delta T} + \frac{1}{\Delta T}\frac{c_p\mathbb{E}\left(\mathbb{I}_{\left\{L>X_{t_{N(S)}}\geqslant M\right\}}\right) + c_c\mathbb{E}\left(\mathbb{I}_{\left\{X_{t_{N(S)}}\geqslant L\right\}}\right)}{\mathbb{E}\left(N\left(S\right)\right)}
\end{aligned}$$

Substiter par (13), (14) et (15) nous donnera le formule exacte de coût moyen par une durée de temps.

## 3.9 Optimisation de maintenance basée sur dégradations

```
# Monte carlo to optimize degradation
set.seed(2019)
L = 20
cost = c(10, 800, 1200) # Inspection, Preventive, Corrective
C_S = function(
    p,
    M
) {
    c = 0
    p = p[!is.na(p)]
    len = length(p)
    if (p[[len]] >= M) {
        n_s = length(p[p < M]) # Last index before >= M
        c = c + cost[[1]] * n_s # Inspection
        if (p[[n_s + 1]] >= L) {
            c = c + cost[[3]] # Corrective
        } else {
            c = c + cost[[2]] # Preventive
        }
    }
    return(c)
}
S = function(
    p,
    deltaT,
    M
) {
    p = p[!is.na(p)]
    len = length(p)
    if (p[[len]] < M) return(0)
    else return(length(p[p < M]) + 1)
}
E_C = function(
```

```
    deltaT,
    M,
    nSim = 200,
    nStep = floor(13 / deltaT),
    silence = TRUE
) {
    simStep = matrix(
        rG(nSim * nStep, deltaT),
        nrow = nSim,
        ncol = nStep
    )
    simProc = t(apply(
        t(simStep),
        2,
        cumsum
    ))
    # Count cost
    simCost = apply(
        t(simProc),
        2,
        C_S,
        M
    )
    # Count time
    simTime = apply(
        t(simProc),
        2,
        S,
        deltaT,
        M
    )
    meanSimCost = mean(simCost[simCost > 0])
    meanSimTime = mean(simTime[simTime > 0])
    meanCostTime = 0
    if (is.nan(meanSimCost) || is.nan(meanSimTime)) meanCostTime = .Machine$
         double.xmax
    else meanCostTime = meanSimCost / meanSimTime
    # For debug purpose
    if (!silence)
        cat(
            "dT =", deltaT,
            "M =", M,
            "mCost =", meanSimCost,
            "mTime =", meanSimTime,
            "mCostTime =", meanCostTime,
            "\n"
        )
    return(meanCostTime)
}
# Optimize
optim_degrad = function(
    lower_deltaT,
    upper_deltaT,
    lower_M,
    upper_M,
    tol = 1e-2,
    div_deltaT = 5,
    div_M = 5,
    cut_deltaT = 2,
    cut_M = 2
){
    deltaT = 0
    M = 0
    obj = 0
    while ((((upper_deltaT - lower_deltaT) / div_deltaT)^2 + ((upper_M -
         lower_M) / div_M)^2) >= tol^2) {
        interval_deltaT = seq(lower_deltaT, upper_deltaT, length.out = div_
             deltaT)
        interval_M = seq(lower_M, upper_M, length.out = div_M)
        cat("deltaT =", interval_deltaT,
            "\nM =", interval_M,
            "\n")
        Z = outer(
            interval_deltaT,
```

```r
            interval_M,
            Vectorize(E_C)
        )
        # First minimum position
        posMin = arrayInd(which.min(Z), dim(Z))
        deltaT = interval_deltaT[[posMin[[1]]]]
        M = interval_M[[posMin[[2]]]]
        obj = Z[posMin]
        cat("deltaT = ", deltaT,
            "\nM = ", M,
            "\nobj = ", obj,
            "\n")
        # Adjust zone of optimize
        l_deltaT = upper_deltaT - lower_deltaT
        l_M = upper_M - lower_M
        upper_deltaT = min(deltaT + l_deltaT / cut_deltaT, upper_deltaT)
        lower_deltaT = max(deltaT - l_deltaT / cut_deltaT, lower_deltaT)
        upper_M = min(M + l_M / cut_M, upper_M)
        lower_M = max(M - l_M / cut_M, lower_M)
    }
    return(list(
        deltaT = deltaT,
        M = M,
        obj = obj
    ))
}
val = optim_degrad(0.001,4,0,20)
# Illustration
interval_deltaT = seq(0.1, 4.1, 0.5)
interval_M = seq(1, 20, 1)
Z = outer(
    interval_deltaT,
    interval_M,
    Vectorize(E_C)
)
library(rgl)
persp3d(
    interval_deltaT,
    interval_M,
    Z,
    xlab = "deltaT",
    ylab = "M",
    zlab = "E_C",
    zlim = c(0, 400),
    col = "lightblue"
)
rgl.snapshot(filename = "img/E_C_degrad.png")
```

# Références


[1] Minka, Thomas P. (2002). "Estimating a Gamma distribution" https://tminka.github.io/papers/minka-gamma.pdf