Projet de fiabilité

Étude de systèmes complexes

MASTER 2 : Mathématique appliquée, parcours modélisation statistique

Fiabilité

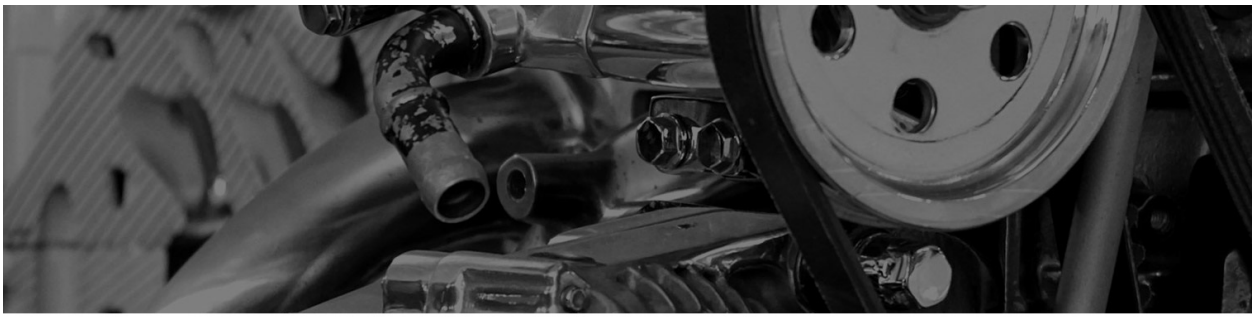
par

Younes Chahma et Camille Giampiccolo

Encadré par

Professeur Célestin C. Kokonendji

24 novembre 2020



Contents

Introduction

À l'heure actuelle, la technologie est tellement développée, notamment dans les entreprises et les usines, qu'il est assez difficile de prévoir les risques. Ces risques peuvent survenir dans énormément de domaines comme par exemple le nucléaire, qui touche l'intégralité de la population. Les risques et les défaillances peuvent vite avoir un impact financier sur une industrie. Comment étudier les risques et aider les entreprises à les prévoir et maintenir leur économie ? Des méthodes mathématiques peuvent aider ces entreprises en modélisant des situations. Dans les parties suivantes, nous étudierons des situations à risque et nous ferons des simulations.

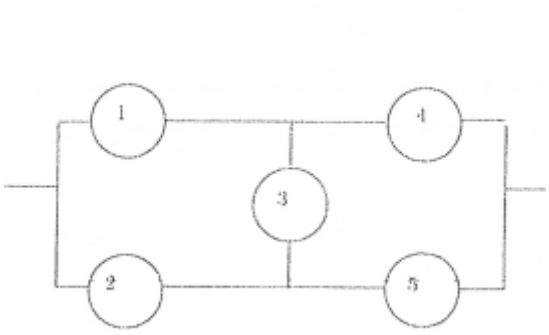
Étude d'un système complexe

Nous allons dans cette première partie de projet, étudier un système composé de deux blocs en série. Nous étudierons le système dans son intégralité, mais aussi par blocs en les isolant, pour faire des comparaisons. Le but est d'estimer le temps de survie du système, de faire des trajectoires selon différents paramètres choisis. Pour cela, nous suivrons l'énoncé du TP1.

Étude de l'état du système

Fonction de structure du bloc 1

Le premier bloc en série du système est représenté par le schéma ci-dessous :



Sa fonction de structure est la suivante :

$\phi(x) = x_3[1-(1-x_1)(1-x_2)][1-(1-x_4)(1-x_5)] + (1-x_3)[1-(1-x_1x_4)(1-x_2x_5)]$

Nous allons l'implémenter de la manière suivante, elle nous renverra l'état du bloc 1 :

```
phi_1 <- function(x){
  rep <- ((x[3]*(1-(1-x[1])*(1-x[2]))*(1-(1-x[4])*(1-x[5])))+(1-x[3])
          *(1-(1-x[1]*x[4])*(1-x[2]*x[5])))
  return(rep)
}
```

Exemple

Dans le cas où les composants 1 et 2 ne fonctionnent pas, mais les autres composants fonctionnent, on obtient le résultat suivant :

```
phi_1(c(0,0,1,1,1))
```

0

La fonction nous renvoie 0, donc le bloc 1 ne fonctionne pas. Ce qui est normal car si les deux premiers composants qui sont en parallèles ne fonctionnent pas, on ne pourra pas atteindre la sortie du circuit.

En revanche, si seulement les premier et quatrième composants fonctionnent, on obtient le résultat suivant :

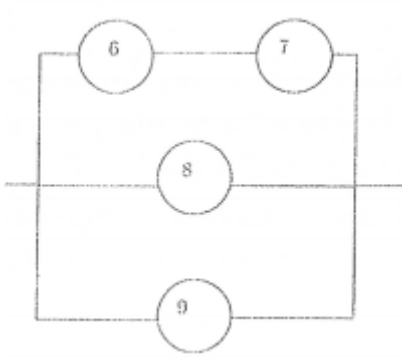
```
phi_1(c(1,0,0,1,0))
```

1

La fonction nous renvoie 1, donc le bloc 1 fonctionne. Ce qui est normal car on peut atteindre la sortie du circuit en passant par les composants 1 et 4.

Fonction de structure du block 2

Le deuxième block en série du système est représenté par le schéma ci-dessous :



Sa fonction de structure est la suivante :

$\phi(x) = [1 - (1 - x_8)(1 - x_9)(1 - x_6 x_7)]$

Nous allons l'implémenter de la manière suivante, elle nous renverra l'état du bloc 2 :

```
phi_2 <- function(x){
  rep <- (1-(1-x[1]*x[2])*(1-x[3])*(1-x[4]))
  return(rep)
}
```

Exemple

Dans le cas où seulement le composant 6 fonctionne, on obtient le résultat suivant :

```
phi_2(c(1,0,0,0))
```

0

Le bloc 2 ne fonctionne pas, ce qui est normal car si on passe par le composant 6, le seul moyen d'atteindre la sortie est en passant par le composant 7 qui dans ce cas de figure, ne fonctionne pas.

En revanche si seulement le composant 8 fonctionne, on obtient :

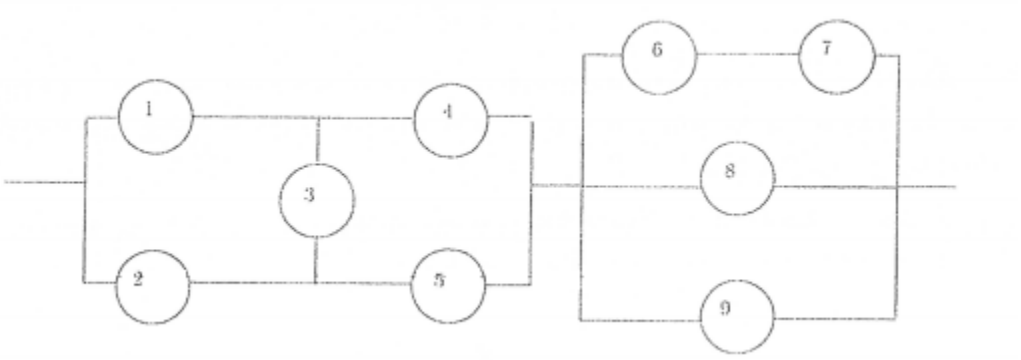
```
phi_2(c(0,0,1,0))
```

1

Le block 2 fonctionne, ce qui est normal car si on passe par le composant 8, on atteint la sortie du circuit.

Fonction de structure du système

Le système, dans son intégralité, est représenté par le schéma ci-dessous :



Sa fonction de structure est la suivante :

$$\phi(x) = [1-(1-x_8)(1-x_9)(1-x_6x_7)]* \\ \{x_3[1-(1-x_1)(1-x_2)][1-(1-x_4)(1-x_5)]+(1-x_3)[1-(1-x_1x_4)(1-x_2x_5)]\}$$

Nous allons l'implémenter de la manière suivante, elle nous renverra l'état du bloc système :

```
phi <- function(x){
  rep <- ((x[3]*(1-(1-x[1])*(1-x[2]))*(1-(1-x[4])*(1-x[5])))+(1-x[3])*
            (1-(1-x[1]*x[4])*(1-x[2]*x[5])))*(1-(1-x[6]*x[7])*(1-x[8])*(1-x[9]))
  return(rep)
}
```

Exemple

Dans le cas où tous les composants fonctionnent, sauf le 2, 3 et 4, on obtient le résultat suivant :

```
phi(c(1,0,0,0,1,1,1,1,1))
```

0

Le système ne fonctionne pas. Ce qui est normal car on entre par le composant 1, mais ensuite, tous les composants qui suivent ne fonctionnent pas, donc impossible d'atteindre le bloc 2 et donc, impossible d'atteindre la sortie.

En revanche, si seulement les composants 3 et 4 ne fonctionnent pas, mais tous les autres composants fonctionnent, on obtient le résultat suivant :

```
phi(c(1,1,0,0,1,1,1,1,1))
```

1

Le système fonctionne, car on entre par le composant 2 qui fonctionne, ensuite le 4 et on atteint le block 2 où tous les composants fonctionnent pour ensuite sortir.

Étude de la durée de vie du système

Fonction de survie

Pour étudier la durée de vie du système, nous implémentons la fonction de survie de durée de vie T du système par le code suivant :

```
survie_sys <- function(loi, lambda, beta, t){
  R1 <- 1-loi(t, shape = beta[1], scale = lambda[1])
  R2 <- 1-loi(t, shape = beta[2], scale = lambda[2])
  R3 <- 1-loi(t, shape = beta[3], scale = lambda[3])
  R4 <- 1-loi(t, shape = beta[4], scale = lambda[4])
  R5 <- 1-loi(t, shape = beta[5], scale = lambda[5])
  R6 <- 1-loi(t, shape = beta[6], scale = lambda[6])
  R7 <- 1-loi(t, shape = beta[7], scale = lambda[7])
  R8 <- 1-loi(t, shape = beta[8], scale = lambda[8])
  R9 <- 1-loi(t, shape = beta[9], scale = lambda[9])
  rep <- ((R3[t]*(1-(1-R1[t])*(1-R2[t]))*(1-(1-R4[t])*(1-R5[t])))+(1-R3[t])*
            (1-(1-R1[t]*R4[t])*(1-R2[t]*R5[t])))*(1-(1-R6[t]*R7[t])*
                (1-R8[t])*(1-R9[t]))
  return(rep)
}
```

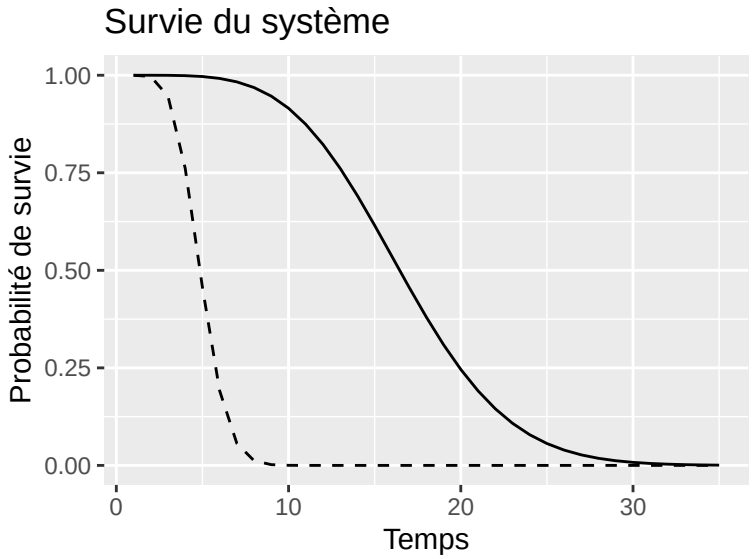
Quelques explication : La foncton prend en argument une fonction de survie pour chaque composant, des paramètres λ_i et βi pour chaque composants $i \in 1, ..., 9$ et une durée t. Elle renvoie la probabilité de survie du système à chaque instant dans t.

Simulations

Pour une première étude, on va regarder comment varie la fonction de survie en fonction de différentes valeurs pour λ.

- on fixe $\beta = 5$, pour tous les composants, pour chaque simulations
- $\lambda = 20$, pour tous les composants, pour la première simulation
- $\lambda = 5$, pour tous les composants, pour la deuxième simulation
- on utilise la fonction de répartition de la loi de weibull

On observe les courbes pour chaque simulations dans le graphique suivant :



La courbe en trait plein représente la cas où $\lambda = 20$, pour tous les composants.

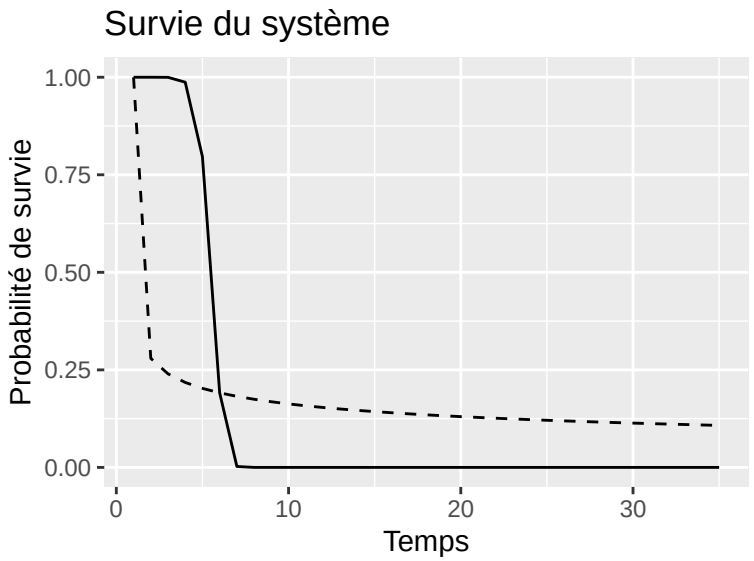
La courbe en pointillés représente la cas où $\lambda = 5$, pour tous les composants.

On observe que si le paramètre $\lambda = 5$, la probabilité de survie diminue plus rapidement que lorsque $\lambda = 20$ (à $\beta = 2$ fixé).

Pour une deuxième étude, on va regarder comment varie la fonction de survie en fonction de différentes valeurs pour β.

- on fixe $\lambda = 5$, pour tous les composants, pour chaque simulations
- $\beta = 0.1$, pour tous les composants, pour la troisième simulation
- $\beta = 5$, pour tous les composants, pour la quatrième simulation
- on utilise la fonction de répartition de la loi de weibull

On observe les courbes pour chaque simulations dans le graphique suivant :



La courbe en trait plein représente la cas où $\beta = 5$, pour tous les composants.

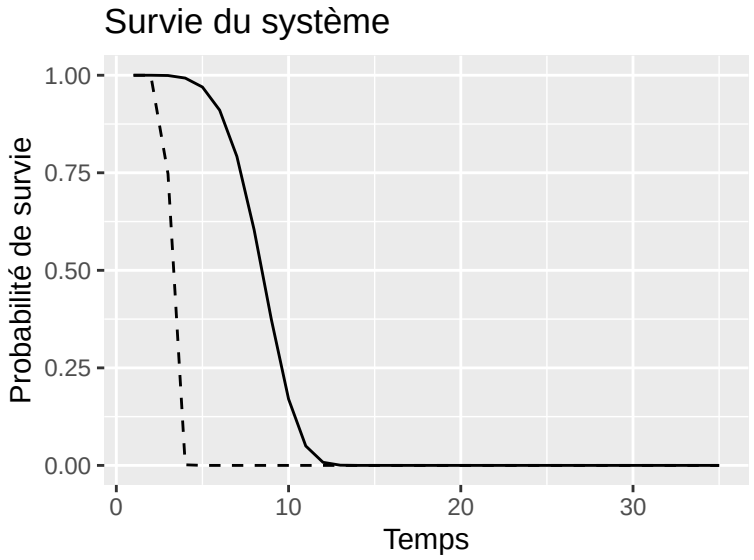
La courbe en pointillés représente la cas où $\beta = 0.1$, pour tous les composants.

On observe que si le paramètre $\beta = 5$, la probabilité de survie diminue jusqu'a 0 assez rapidement mais si $\beta = 0.1$, la probabilité de survie diminue encore plus rapidement mais ensuite, stagne autour de 0.1 (à $\lambda = 5$ fixé).

Pour une troisième étude, on va regarder comment varie la fonction de survie en fonction de différentes valeurs pour λ avec λ qui varie aussi en fonction des composants.

- on fixe $\beta = 5$, pour tous les composants, pour chaque simulations
- $\lambda_1 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = \lambda_9 = 10$, et $\lambda_2 = \lambda_3 = \lambda_4 = 2$, pour la cinquième simulation.
- $\lambda_1 = \lambda_4 = \lambda_8 = 10$, et $\lambda_2 = \lambda_3 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_9 = 2$, pour la sixième simulation
- on utilise la fonction de répartition de la loi de weibull

On observe les courbes pour chaque simulations dans le graphique suivant :



La courbe en trait plein représente la cas où $\lambda_1 = \lambda_4 = \lambda_8 = 10$, et $\lambda_2 = \lambda_3 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_9 = 2$.

La courbe en pointillés représente la cas où $\lambda_1 = \lambda_5 = \lambda_6 = \lambda_7 = \lambda_8 = \lambda_9 = 10$, et $\lambda_2 = \lambda_3 = \lambda_4 = 2$.

La probabilité de survie diminue plus lentement pour la sixième simulation que pour la cinquième simulation. Or, nous avons vu précédemment que plus λ est élevé et plus la probabilité de survie diminue lentement dans le temps.

- Dans la sixième simulation, nous avons fixé $\lambda = 10$ pour un ensemble de composants importants au bon fonctionnement du système.
- Dans la cinquième simulation, nous avons fixé $\lambda = 2$, moins élevé que pour la sixième simulation, pour un ensemble de composants importants au bon fonctionnement du système.

Il est donc normal d'observer que cette différence de diminution de probabilité de survie dans le temps entre ces deux simulations.

Étude de trajectoires pour d'autres fonctions

Implémentation de la fonction

Nous allons implémenter une fonction $t \mapsto -ln\bar{F}(t)/t$. Le code est le suivant :

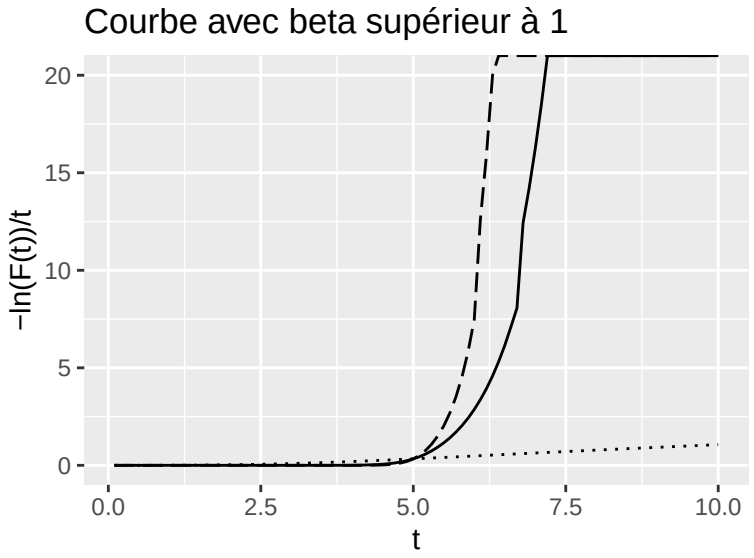
```
ln_surv <- function(lambda, beta, t){
  rep=NULL
  for(i in 1:length(t)){
    R1 <- 1-pweibull(t[i], shape = beta[1], scale = lambda[1])
    R2 <- 1-pweibull(t[i], shape = beta[2], scale = lambda[2])
    R3 <- 1-pweibull(t[i], shape = beta[3], scale = lambda[3])
```

```
    R4 <- 1-pweibull(t[i], shape = beta[4], scale = lambda[4])
    R5 <- 1-pweibull(t[i], shape = beta[5], scale = lambda[5])
    R6 <- 1-pweibull(t[i], shape = beta[6], scale = lambda[6])
    R7 <- 1-pweibull(t[i], shape = beta[7], scale = lambda[7])
    R8 <- 1-pweibull(t[i], shape = beta[8], scale = lambda[8])
    R9 <- 1-pweibull(t[i], shape = beta[9], scale = lambda[9])
    rep[i] <- -log(((R3*(1-(1-R1)*(1-R2))*(1-(1-R4)*(1-R5)))+(1-R3)*
                  (1-(1-R1*R4)*(1-R2*R5)))*(1-(1-R6*R7)*(1-R8)*
                    (1-R9)))/t[i]
  }
  return(rep)
}
```

Simulations

Pour une première étude avec $\beta > 1$ pour tous les composants on fixe $\lambda = 5$ et

- Pour une première simulation, $\beta = 2$ pour tous les composants
- Pour une deuxième simulation, $\beta = 10$ pour tous les composants
- Pour une troisième simulation, $\beta = 15$ pour tous les composants



La courbe en trait plein représente la deuxième simulation.

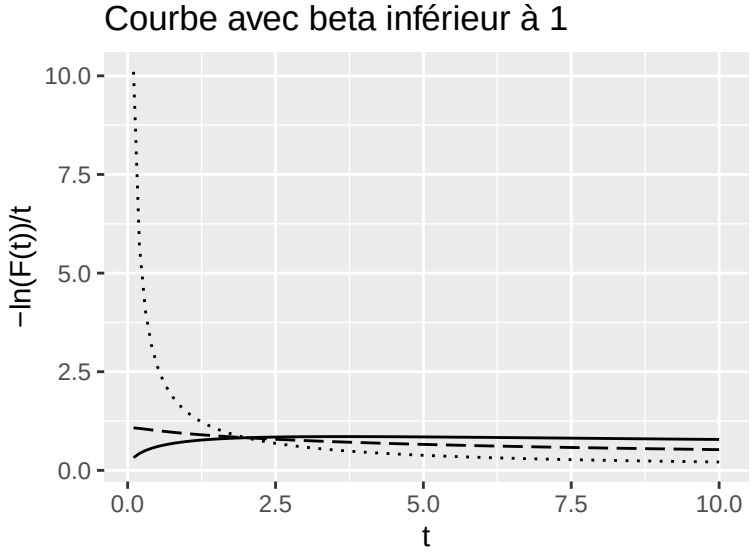
La courbe en pointillés représente la première simulation.

La courbe en traits-pointillés représente la troisième simulation.

On observe que plus le paramètre β est grand, plus la courbe augmente rapidemment.

Pour une deuxième étude avec β inférieur à 1, pour tous les composants on fixe $\lambda = 5$ et

- Pour une première simulation, $\beta = 0.1$ pour tous les composants
- Pour une deuxième simulation, $\beta = 0.5$ pour tous les composants
- Pour une deuxième simulation, $\beta = 0.7$ pour tous les composants



La courbe en trait plein représente la troisième simulation.

La courbe en pointillés représente la première simulation.

La courbe en traits-pointillés représente la deuxième simulation.

On observe que la courbe où $\beta = 0.1$ commence avec une valeur élevé mais décroit très vite à 0. Les autres courbes stagnent aux alentours de 0.5

Réalisation de la fonction de structure sur un intervalle de temps fixé.

Implémentation de la fonction

Nous allons désormais implémenter une réalisation de $\phi(X_t)$ qui est la fonction de structure sur un intervalle $[0, a]$.

Pour cela, nous allons utiliser la fonction quantile.

Définition : Soit G une fonction de répartition. On appelle fonction quantile la fonction $G^-(u) = inf\{x : G(x) \geq u\}$, $u \in]0, 1[$ Alors si U est une variable de loi uniforme sur $[0, 1]$, la variable $X = G^-(u)$ a pour fonction de répartition G.

Dans notre cas, on prend G, la fonction de répartition de la loi de weibull. Ainsi, $G(t, \lambda, \beta) = 1 - e^{-(t/\lambda)^\beta}$ est la fonction de répartition d'une loi de weibull de paramètres λ et β.

Sa fonction quantile est donc $G^-(u) = \lambda(-ln(1-u))^{1/\beta} = \lambda(-ln(u))^{1/\beta}$ Car U et $1-U$ ont la même loi. $G^-(u)$ suit une loi de weibull de paramètre λ et β.

En utilisant cette fonction quantile, on peut implémenter une réalisation de $\phi(X_t)$, sur l'intervalle $[0, a]$, avec le code suivant :

```
phi_t <- function(a, n, lambda, beta){
  t <- seq(0, a,length.out =n)
  y <- rep(0,n)
  u <- runif(9,0,1)
  x <- matrix(0, nrow = n, ncol = 9)

  for(i in 1:n){
    for(j in 1:9){
      if(t[i] < (lambda*(-log(u[j]))^(1/beta))){
        x[i,j] <- 1
      }
    }
    y[i] <- phi(x[i,])
  }
  return(y)
}
```

Ce code prend en argument, les paramètres λ et β des lois de weibull, et l'intervalle de temps sur lequel on veut étudier l'état du système.

Simulations

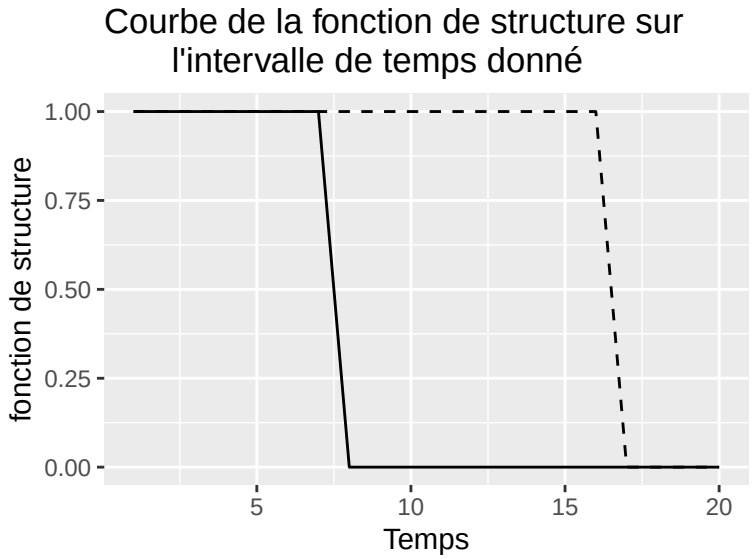
Pour une première simulation, on prend :

- $\lambda = 2$ pour tous les composants
- $\beta = 1$ pour tous les composants

Pour une deuxième simulation, on prend :

- $\lambda = 3$ pour tous les composants
- $\beta = 6$ pour tous les composants

La première simulation est représentée par la courbe en trait plein. La deuxième simulation est représentée par la courbe en pointillés.



Pour la première simulation, on observe que à $t = 8$ le système tombe en panne.
Pour la première simulation, on observe que à $t = 17$ le système tombe en panne.

Estimation de la durée de vie

Estimation de la durée de vie T du système

Dans cette partie, les intervalles de temps seront en jours. La fonction suivante renvoie la durée de vie du système :

```
duree_vie_sys <- function(a, n, lambda, beta){
  y<-phi_t(a, n, lambda, beta)
  d<-0
  for (i in n:1){
    if(y[i]==1 & i==n ){
      d=n
      break
    }
    if(y[i]==1){
      d=i
      break
    }
  }
  return(d)
}
```

Cette fonction prend en argument les paramètres λ et β ainsi que les informations necessaires pour l'intervalle de temps.

Elle nous retourne les derniers temps sur lesquelles le système aura fonctionné.

Par exemple, pour $\lambda = 5$, $\beta = 15$ et un intervalle composé de 100 valeurs equidistantes entre 0 et 40, on obtient

```
duree_vie_sys(40,100,5,15)
```

12

Le système vit de $t = 0$ à $t = 12$
Donc il vit 4.4444444 jours. (12 eme valeur de l'intervalle donnée)

Estimation de la durée de vie T1 du bloc 1

En ayant implémenté une fonction que l'on nomme *phi_t_b*1 qui renvoie une réalisation de $\phi_1(X_t)$ (fonction de structure pour le bloc 1 sur un intervalle de temps donné) et une fonction qui renvoie l'espérance du temps de survie pour le bloc 1 que l'on nomme *duree_vie_b*1, nous obtenons pour les mêmes paramètres choisis que pour le système :

```
duree_vie_b1(40,100,5,15)
```

12

Le système vit de $t = 0$ à $t = 12$
Donc il vit 4.4444444. (12 eme valeur de l'intervalle donnée)

Estimation de la durée de vie T2 du bloc 2

En ayant implémenté une fonction que l'on nomme *phi_t_b*2 qui renvoie une réalisation de $\phi_2(X_t)$ (fonction de structure pour le bloc 2 sur un intervalle de temps donné) et une fonction qui renvoie l'espérance du temps de survie pour le bloc 2 que l'on nomme *duree_vie_b*2, nous obtenons pour les mêmes paramètres que précédemment :

```
duree_vie_b2(40,100,5,15)
seq(0,40,length.out=100)
```

14

Le système vit de $t = 0$ à $t = 14$
Donc il vit 5.2525253 jours. (14 eme valeur de l'intervalle donnée)

Estimation des espérences des durées de vie

Pour le système

Nous cherchons à estimer $\mu = \mathbb{E}(T)$ à l'aide de la loi forte des grands nombres.

La fonction suivante permettra de réaliser plusieurs fois, la fonction qui calcule la durée de vie du système et d'enregistrer toutes les réalisations dans un vecteur.

```
n_realisation_T_sys<-function(a, n, lambda, beta,ntot){
  vect=NULL
```

```
  for(i in 1:ntot){
    vect[i]=duree_vie_sys(a, n, lambda, beta)
  }
  return(vect)
}
```

La fonction suivante calcule la moyenne de toute les réalisations.

```
mu=function(a, n, lambda, beta,ntot){
  v=n_realisation_T_sys(a, n, lambda, beta,ntot)
  1/ntot*sum(v)
}
```

Ainsi, pour 100 réalisations et pour $\lambda = 5$, $\beta = 15$ et un intervalle composé de 100 valeurs equidistantes entre 0 et 40, on obtient

```
mu(40,100,5,15,100)
```

12.4

On considére que $(T_i)_{i=1}^n$ est une variable aléatoires indépendante et identiquement distribuée.

D'après la loi forte des grands nombres, $\frac{1}{100}\sum_{1=1}^{100} T_i$ converge presque sûrement vers la constante $\mathbb{E}(T_1)$

Ainsi $\mathbb{E}(T) = 13.4$

Pour le bloc 1

Nous cherchons à estimer $\mu_1 = \mathbb{E}(T_1)$.

On implémente une fonction nommée *n_realisation_T_b1* qui réalise plusieurs fois la fonction qui calcule la durée de vie du bloc 1 et qui enregistre toutes les réalisations dans un vecteur.

On implémente aussi une fonction nommée *mu_1* qui calcule la moyenne de toute ces réalisations.

Ces fonctions sont en annexe car elle sont assez similaire à celle du système.

Ainsi, pour 100 réalisations et pour $\lambda = 5$, $\beta = 15$ et un intervalle composé de 100 valeurs equidistantes entre 0 et 40, on obtient :

```
mu_1(40,100,5,15,100)
```

12.59

On considére que $(T_{1_i})_{i=1}^n$ est une variable aléatoire indépendante et identiquement distribuée.

D'après la loi forte des grands nombres, $\frac{1}{100}\sum_{1=1}^{100} T_{1_i}$ converge presque sûrement vers la constante $\mathbb{E}(T_{1_1})$

Ainsi $\mathbb{E}(T_1) = 12.59$

Pour le bloc 2

Nous cherchons à estimer $\mu_2 = \mathbb{E}(T_2)$.

On implémente une fonction nommée *n_realisation_T_b2* qui réalise plusieurs fois la fonction qui calcule la durée de vie du bloc 1 et qui enregistre toutes les réalisations dans un vecteur.

On implémente aussi une fonction nommée *mu_2* qui calcule la moyenne de toute ces réalisations.

Ces fonctions sont en annexe car elle sont assez similaires à celles du système et du bloc 1.

Ainsi, pour 100 réalisations et pour $\lambda = 5$, $\beta = 15$ et un intervalle composé de 100 valeurs equidistantes entre 0 et 40, on obtient :

```
mu_2(40,100,5,15,100)
```

13.12

On considére que $(T_{2_i})_{i=1}^n$ est une variable aléatoires indépendante et identiquement distribuée.

D'après la loi forte des grands nombres, $\frac{1}{100}\sum_{1=1}^{100} T_{2_i}$ converge presque sûrement vers la constante $\mathbb{E}(T_{2_1})$

Ainsi $\mathbb{E}(T_2) = 13.12$

Inégalité entre les espérences

Désormais nous allons vérifier expérimentalement que $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ est vraie pour $\beta > 1$ mais pas pour $\beta \leq 1$

Pour beta supérieur strictement à 1

Nous implémentons une fonction qui renvoie $TRUE$ si $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ est vérifiée et $FALSE$ sinon. On fixera $\lambda = 5$, $\beta = 15$ et un intervalle composé de 100 valeurs equidistantes entre 0 et 40. Pour calculer les moyennes, la fonction réalisera 100 réalisations.

Le code est le suivant :

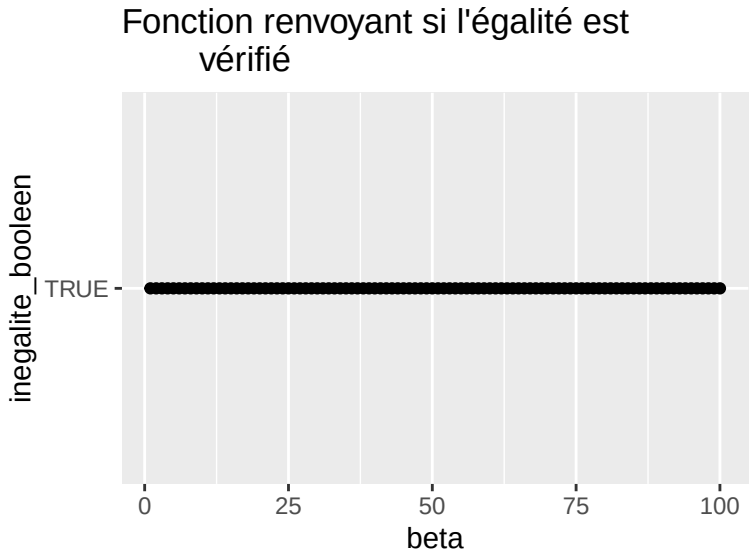
```
a_val=40
n_val=100
lambda_val=5
ntot_val=100

inegalite_booleen=function(beta,a=a_val, n=n_val, lambda=lambda_val,ntot=ntot_val){
  rep=NULL
  mu1=NULL
```

```
  mu2=NULL
  mu0=NULL
  s=NULL
  for(i in 1:length(beta)){
  mu1[i]=mu_1(a, n, lambda, beta[i],ntot)
  mu2[i]=mu_2(a, n, lambda, beta[i],ntot)
  mu0[i]=mu(a, n, lambda, beta[i],ntot)
  s[i]=((1/mu1[i])+(1/mu2[i]))^(-1)
  if(mu0[i]>=s[i]){

    rep[i]=TRUE
  }
  else{
    rep[i]=FALSE
  }
  }
  return(rep)
}
```

Ainsi, pour β qui varie entre 0 et 100, on obtient :



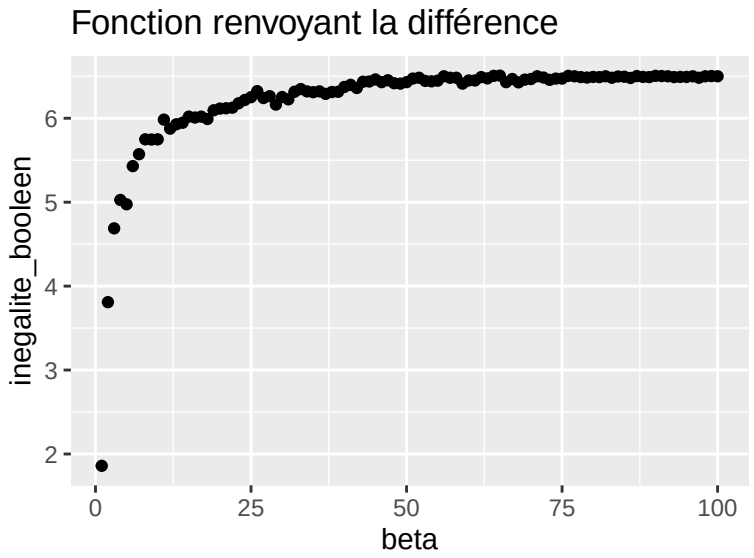
On peut aussi implémenter une autre fonction qui renvoie si $\mu - \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$

```
a_val=40
n_val=100
lambda_val=5
ntot_val=100

inegalite=function(beta,a=a_val, n=n_val, lambda=lambda_val,ntot=ntot_val){
  rep=NULL
```

```
  mu1=NULL
  mu2=NULL
  mu0=NULL
  s=NULL
  for(i in 1:length(beta)){
  mu1[i]=mu_1(a, n, lambda, beta[i],ntot)
  mu2[i]=mu_2(a, n, lambda, beta[i],ntot)
  mu0[i]=mu(a, n, lambda, beta[i],ntot)
  s[i]=((1/mu1[i])+(1/mu2[i]))^(-1)
  rep[i]=mu0[i]-s[i]
  }
  return(rep)
}
```

Ainsi, pour β qui varie entre 0 et 100, on obtient :



On observe que toutes les valeurs de la fonction sont supérieures à 0 pour $\beta > 1$. Ainsi $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ est vérifiée expérimentalement.

Pour beta inférieur à 1

Pour montrer que $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ n'est pas toujours vérifiée dans le cas où $\beta \leq 1$, nous allons trouver une valeur de β pour laquelle, l'inégalité n'est pas vérifiée. Ainsi pour $\beta = 0.1$, on obtient

```
inegalite_booleen(0.1)
```

FALSE

Ainsi, $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ n'est pas vérifiée pour $\beta = 0.1 < 1$

Disponibilité

La disponibilité moyenne sur un intervalle de temps donné peut être évaluée par le rapport :

$$\frac{MTBF}{MTTR + MTBF}$$

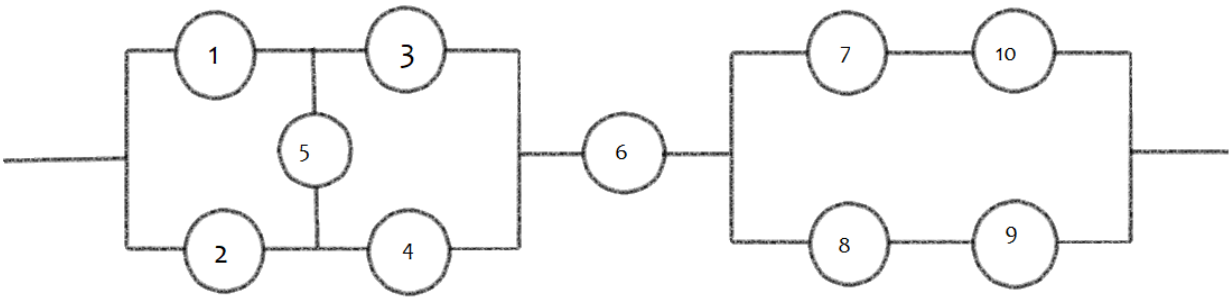
où MTBF est la moyenne des temps de bon fonctionnement entre défaillances consécutives et MTTR ou encore Moyenne des Temps Techniques de Réparation.

Étude d'un système complexe avec possibilité de réparations de composants

Dans cette seconde partie, nous étudierons un autre système complexe, composé de 10 composants. Le but de cette partie sera de faire de nombreuse simulation et voir ce qui se passe dans le cas où un composant est réparable. Nous étudierons aussi l'impacte financier des pannes pour une entreprise. Dans cette partie, nous éviterons de mettre les morceaux de code car nous en avons déja beaucoup mis dans la partie précédente et dans cette partie, les code sont plus ou moins similaires.

Fonction de structure

Le système que nous allons simuler a 10 composants. Voici le schéma du système :



Il a pour fonction de structure :

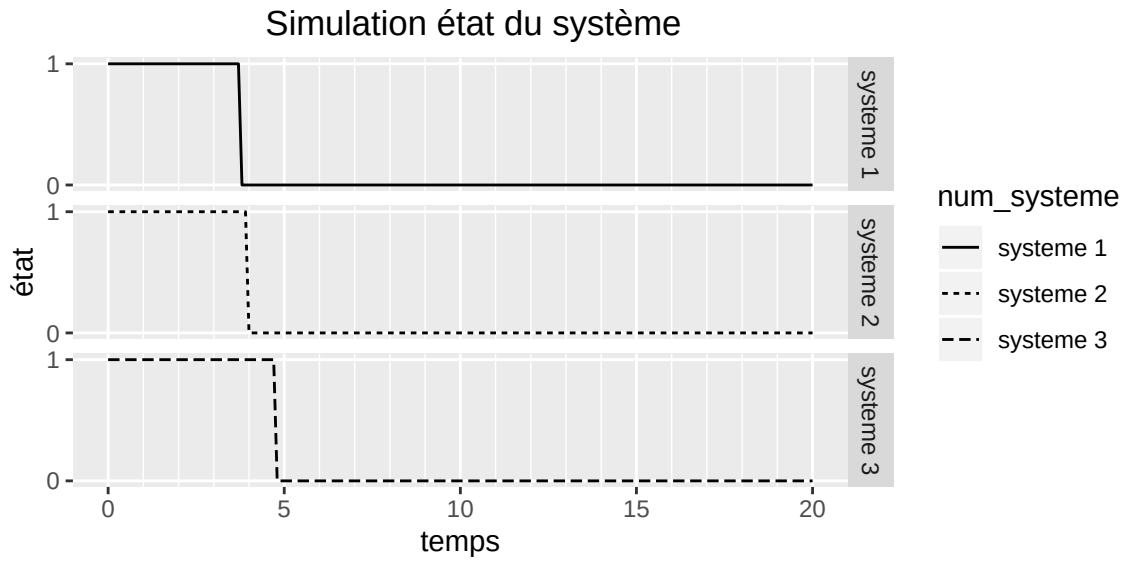
$$\phi(x) = [x_5(1-(1-x_1)(1-x_2))(1-(1-x_3)(1-x_4)) + (1-x_5)(1-(1-x_1x_3)(1-x_2x_4))]* \\ x_6[1-(1-x_7x_{10})(1-x_8x_9)]$$

Trajectoire de l'état du système et espérence de la durée de vie

Trajectoire de l'état du système

Nous allons maintenant simuler l'état du système. On considérera que les composants du système suivent des lois de Weibull de paramètre $\lambda = 5$ et $\beta = 5$.

La trajectoire du système est :



En fixant les paramètres à $\lambda = 5$ et $\beta = 5$, le premier système a une durée de vie de t = 3,8, le deuxième de t = 4 et le troisième de t = 4.8.

On va maintenant utiliser la loi forte des grands nombres pour estimer l'espérance de la durée de vie de notre système. On simule le temps de vie de moyen sur 10000 systèmes.

Estimation de l'espérance de la durée de vie et intervalle de confiance

Nous allons désormais coder une fonction (qui se trouve en annexe), que va nous renvoyer l'espérance de la durée de vie T du système. On utilisera la loi forte des grands nombres tout comme dans le TP1, car on considére que $(T_i)_{i=1}^n$ est une variable aléatoires indépendante et identiquement distribuée. On obtient

```
## [1] 3.93306
```

On estime l'espérance du temps de vie de notre système à 3,933.

On implémente l'intervalle de confiance au seuil de 95% suivant :

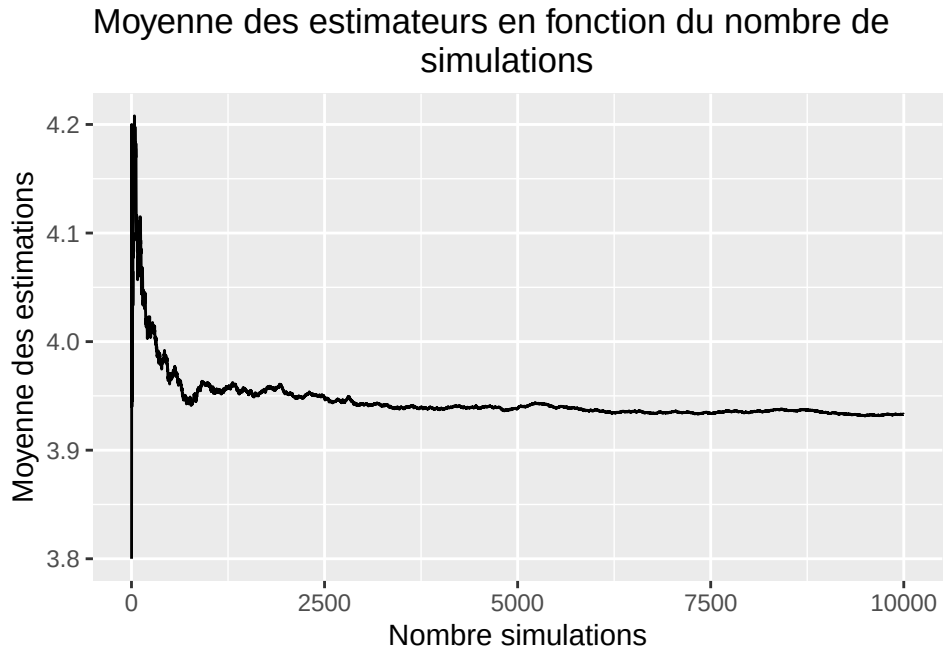
$$IC = \left[\frac{1}{n}\sum_{i=1}^{n} t_i - 1.96\frac{(var(t_i))^{1/2}}{n^{1/2}}; \frac{1}{n}\sum_{i=1}^{n} t_i + 1.96\frac{(var(t_i))^{1/2}}{n^{1/2}}\right]$$

On obtient

```
## [1] 3.919898 3.946222
```

Son intervalle de confiance à 95% est $[3,920; 3.946]$.

On va maintenant représenter notre estimation de l'espérance de la durée de vie de notre système en fonction du nombre de simulations choisies.



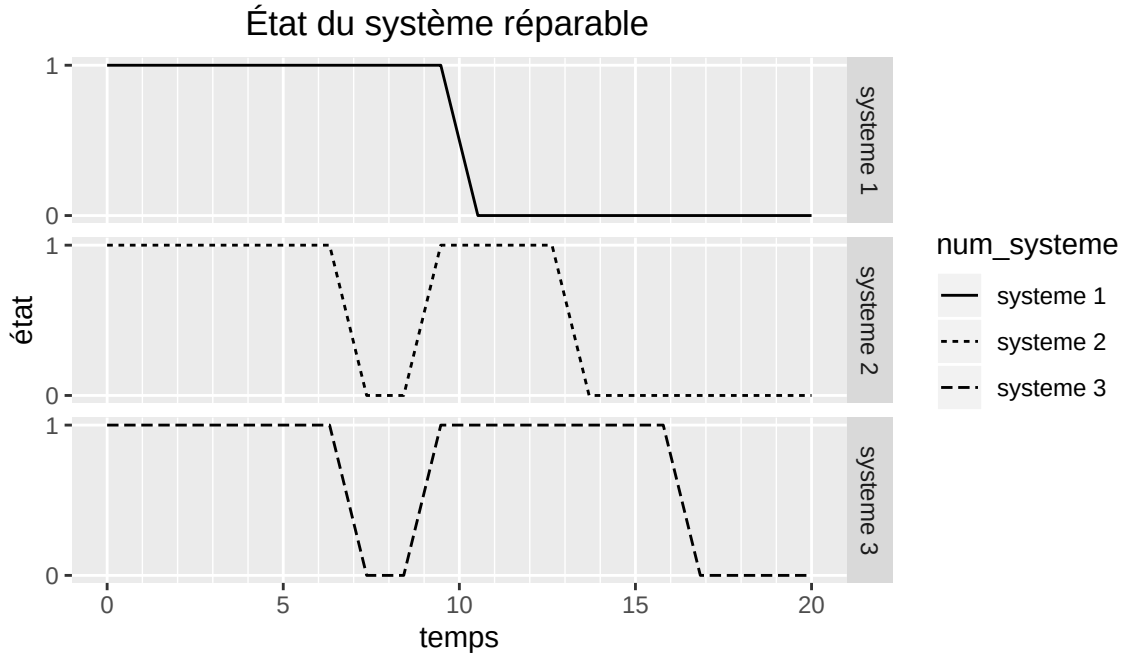
On voit sur ce graphe que le temps de vie moyen converge entre 3.9 et 4. Notre estimation a l'air de se stabiliser à partir de 3000 simulations.

Réparation du système

Trajectoire de l'état du système avec réparation

Le composant critique que nous avons choisis est le composant x_6. En effet, il est un composant critique car il est en série avec tout le reste du système et donc une panne de ce composant entraine forcément une panne du système. Nous allons maintenant implémenter le fait que ce composant soit réparable. On part du principe qu'un technicien vient tous les intervalles de temps δ pour le réparer : si le composant est en panne, il le répare, sinon il ne fait rien. Dans les deux cas, le déplacement du technicien implique un coût.

Nous allons tracer quelques trajectoires de l'état de notre nouveau système réparable. Cette fois-ci, on fixe $\lambda = 15$ et $\beta = 3$.



- Pour la première trajectoire, on voit que le système tombe en panne au temps t = 10 et il n'y a pas de réparation ensuite, du moins pas de réparation qui remette le système en marche.
- Pour la deuxième trajectoire, on voit que le système tombe une première fois en panne au temps t = 7, puis il y a réparation au temps t = 9, puis le système retombe en panne au temps t = 13. On peut donc dire que la réparation du composant critique a permis au système de tenir un peu plus longtemps que prévu (il a gagné t = 4 de durée de vie).
- Pour le troisième système, on remarque qu'il tombe une première fois en panne et qu'il est réparé en même temps que le système 2 (aux temps t = 7 puis t = 9), seulement ce système met plus de temps que le précédent à retomber en panne (il retombe en panne à t = 16 contre t = 13 pour le précédent). Dans ce système, le fait que le composant critique soit réparable a doublé la durée de vie de notre système (il passe d'une durée de vie de 6 à 12).

Nous allons maintenant regarder avec quel intervalle de temps δ il faut demander au technicien d'intervenir pour que le système soit le plus rentable possible. La fonction de récompense est la suivante :

$$R(\delta) = a\mathbb{E}(T) - C\mathbb{E}(N)$$

. Le coefficients a correspond au gain du système associé au temps, et le coefficient C correspond au coût d'intervention du technicien. N est le nombre moyen d'intervention du technicien avant l'instant T de panne du système.

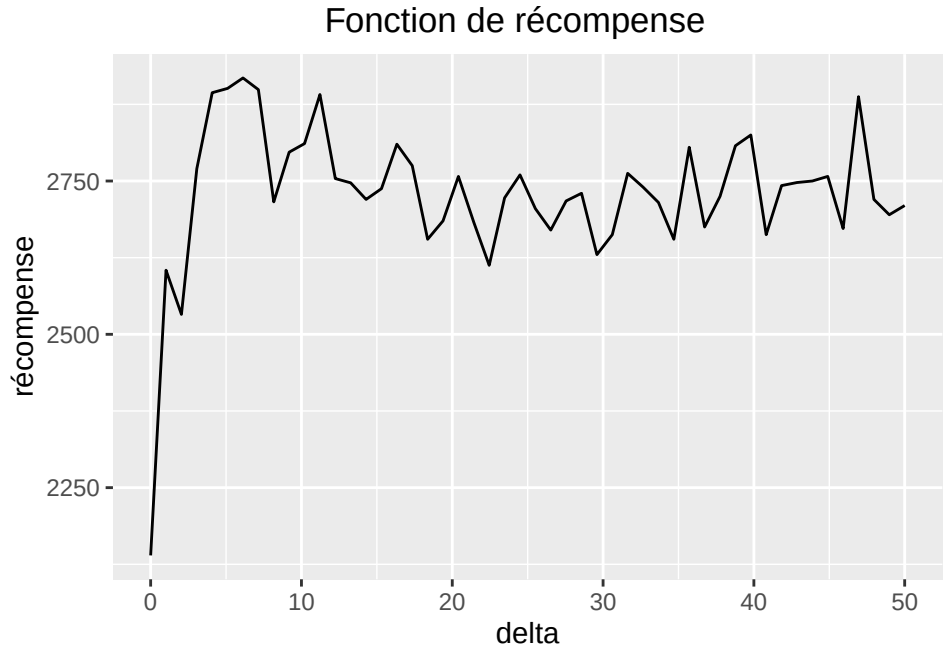
On implémente cette fonction, puis on cherche à la maximiser par rapport à δ. Comme précédemment les valeurs de $\mathbb{E}(T)$ et $\mathbb{E}(N)$ sont estimées par la loi des grands nombres.

Fonction de récompense

Pour la maximisation, on va reprendre les paramètres utilisés précédemment : on fixe $\lambda = 15$, $\beta = 3$. Pour les coefficients de gain associé au temps et de cout d'intervention du technicien, on les fixe arbitrairement : on fixe $a = 250$ et $C = 100$.

```
## $maximum
## [1] 78.01667
##
## $objective
## [1] 2807.5
```

Lorsqu'on cherche à maximiser la fonction de récompense, on obtient qu'il faut que le technicien intervienne tous les 78 temps pour avoir un résultat optimal, et ce pour une récompense de 2807. Lorsque qu'on regarde l'évolution de la fonction recompense en fonction de δ, on obtient :



Ici on se limite à δ allant de 1 à 50 pour limiter le temps de compilation. On voit sur le graphe que la fonction récompense a une allure croissante : plus l'intervalle de temps entre deux interventions du technicien est élevé (donc moins on demande au technicien d'intervenir), plus la récompense est élevée. On en déduit donc que l'intervention du technicien n'est pas rentable en vue de notre système.

Conclusion

Ce projet est intéressant dans le sens où on a mis en pratique des connaissances que nous avions aquis théoriquement cette année.
On a pu visualiser comment évoluent les risques, les durées de vie d'un système en fonction de différents paramètres de la modélisation. La partie informatique dans ce projet est assez importante, il faut implémenter beaucoup de fonctions. Nous avons remarquer que le coté financier peut évoluer considérablement en fonction des risques de défaillance. Il est très important de trouver un bon compromis entre le cout de la main d'oeuvre pour réparer un composant et le gain que peut obtenir une industrie sur une période fixée.

Annexe

TP1

Fonction de survie Pour le système

```
loi = pweibull
lambda=20*rep(1,9)
beta=2*rep(1,9)
t=0:35

loi = pweibull
lambda1=5*rep(1,9)
beta1=2*rep(1,9)
t=0:35


g <- ggplot() +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda, beta,t))) +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda1, beta1,t)),
            linetype = "dashed") +

  labs(title = "Survie du système",
       x = "Temps",
       y = "Probabilité de survie")
g
```

```
loi = pweibull
lambda=5*rep(1,9)
beta=0.1*rep(1,9)
t=0:35

loi = pweibull
lambda1=5*rep(1,9)
beta1=5*rep(1,9)
t=0:35

g <- ggplot() +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda, beta,t)),
            linetype = "dashed") +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda1, beta1,t))) +


  labs(title = "Survie du système",
       x = "Temps",
```

```
       y = "Probabilité de survie")
g
```

```
loi = pweibull
lambda=c(10,2,2,2,10,10,10,10,10)
beta=5*rep(1,9)
t=0:35

loi = pweibull
lambda1=c(10,2,2,10,2,2,2,10,2)
beta1=5*rep(1,9)
t=0:35

g <- ggplot() +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda, beta,t)),
            linetype = "dashed") +
  geom_line(aes(x = 1:35, y = survie_sys(loi, lambda1, beta1,t))) +


  labs(title = "Survie du système",
       x = "Temps",
       y = "Probabilité de survie")
g
```

Étude de trajectoires pour d'autres fonctions

```
lambda=5*rep(1,9)
beta=2*rep(1,9)
t=seq(0.1, 10, 0.1)

lambda1=5*rep(1,9)
beta1=10*rep(1,9)

lambda2=5*rep(1,9)
beta2=15*rep(1,9)

g <- ggplot() +
  geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda, beta, t)),
            linetype = "dotted") +
    geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda1, beta1, t)))+
        geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda2, beta2, t)),
                  linetype = "longdash") +
```

```
  labs(title="Courbe avec beta supérieur à 1",
       x = "t",
       y = "-ln(F(t))/t")

g
```

```
lambda=2*rep(1,9)
beta=0.1*rep(1,9)
t=seq(0.1, 10, 0.1)

lambda1=2*rep(1,9)
beta1=0.5*rep(1,9)

lambda2=2*rep(1,9)
beta2=0.7*rep(1,9)


g <- ggplot() +
  geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda, beta, t)),
            linetype = "dotted") +
    geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda1, beta1, t)),
              linetype = "longdash")+
        geom_line(aes(x = seq(0.1, 10, 0.1), y = ln_surv(lambda2, beta2, t))) +


  labs(title="Courbe avec beta inférieur à 1",
       x = "t",
       y = "-ln(F(t))/t")

g
```

Réalisation de la fonction de structure sur un intervalle de temps fixé.

```
set.seed(1234)
g <- ggplot() +
  geom_line(aes(x = 1:20, y = phi_t(4,20,2,1))) +
  geom_line(aes(x = 1:20, y = phi_t(4,20,3,6)),linetype = "dashed") +
  labs(title="Courbe de la fonction de structure sur
       l'intervalle de temps donnée",
       x = "Temps",
       y = "fonction de structure")
g
```

Estimation de la durée de vie

```
phi_t_b1 <- function(a, n, lambda, beta){
  t <- seq(0, a,length.out =n)
  y <- rep(0,n)
  u <- runif(5,0,1)
  x <- matrix(0, nrow = n, ncol = 5)

  for(i in 1:n){
    for(j in 1:5){
      if(t[i] < (lambda*(-log(u[j]))^(1/beta))){
        x[i,j] <- 1
      }
    }
    y[i] <- phi_1(x[i,])
  }
  return(y)
}
```

```
duree_vie_b1 <- function(a, n, lambda, beta){
  y<-phi_t_b1(a, n, lambda, beta)
  d<-0
  for (i in n:1){
    if(y[i]==1 & i==n ){
      d=n
      break
    }
    if(y[i]==1){
      d=i
      break
    }
  }
  return(d)
}
```

```
phi_t_b2 <- function(a, n, lambda, beta){
  t <- seq(0, a,length.out =n)
  y <- rep(0,n)
  u <- runif(4,0,1)
  x <- matrix(0, nrow = n, ncol = 4)

  for(i in 1:n){
    for(j in 1:4){
      if(t[i] < (lambda*(-log(u[j]))^(1/beta))){
        x[i,j] <- 1
```

```
      }
    }
    y[i] <- phi_2(x[i,])
  }
  return(y)
}
```

```
duree_vie_b2 <- function(a, n, lambda, beta){
  y<-phi_t_b2(a, n, lambda, beta)
  d<-0
  for (i in n:1){
    if(y[i]==1 & i==n ){
      d=n
      break
    }
    if(y[i]==1){
      d=i
      break
    }
  }
  return(d)
}
```

Estimation des espérences des durées de vie Pour le block 1

```
n_realisation_T_b1<-function(a, n, lambda, beta,ntot){
  vect=NULL
  for(i in 1:ntot){
    vect[i]=duree_vie_b1(a, n, lambda, beta)
  }
  return(vect)
}

mu_1=function(a, n, lambda, beta,ntot){
  v=n_realisation_T_b1(a, n, lambda, beta,ntot)
  1/ntot*sum(v)
}
mu_1(40,100,5,15,100)
```

Estimation des espérences des durées de vie pour le block 2

```
n_realisation_T_b2<-function(a, n, lambda, beta,ntot){
  vect=NULL
  for(i in 1:ntot){
```

```
    vect[i]=duree_vie_b2(a, n, lambda, beta)
  }
  return(vect)
}

mu_2=function(a, n, lambda, beta,ntot){
  v=n_realisation_T_b2(a, n, lambda, beta,ntot)
  1/ntot*sum(v)
}
```

projet

question 1

```
phi <- function(x){
  rep <- (x[5]*((1-(1-x[1])*(1-x[2]))*(1-(1-x[3])*(1-x[4])))+(1-x[5])*
            (1-(1-x[1]*x[3])*(1-x[2]*x[4])))*x[6]*(1-(1-x[7]*x[10])*(1-x[8]*x[9]))
  return(rep)
}
```

Question 2

```
simu_etat_systeme <- function(lambda, beta, t){
  X <- matrix(data = rep(0, 10*length(t)), ncol = 10, nrow = length(t))
  S <- rep(0,length(t))
  u <- runif(10, 0, 1)
  w <- rweibull(10, lambda, beta)
  max <- max(t)

  for(i in 1:length(t)){
    for(j in 1:10){
      if(t[i] < w[j]){
        X[i,j] <- 1
      }
    }
    S[i] <- phi(X[i,])
    if((S[i] == 0) && (S[i-1] == 1)){
      max <- t[i]
    }
  }
  return(list(etat = S, last = max))
}
```

```r
set.seed(568)
lambda <- 5
beta <- 5
t <- seq(0, 20, 0.1)

set.seed(1)
systeme1 <- simu_etat_systeme(lambda, beta, t)
systeme2 <- simu_etat_systeme(lambda, beta, t)
systeme3 <- simu_etat_systeme(lambda, beta, t)
df1 <- data.frame(systeme = c(systeme1[[1]], systeme2[[1]], systeme3[[1]]),
                  num_systeme = c(rep("systeme 1",length(t)),
                                  rep("systeme 2",length(t)),
                                  rep("systeme 3",length(t))),
                  t = rep(t, 3))

g1 <- df1 %>%
  ggplot() +
  geom_line(aes(x = t, y = systeme, linetype = num_systeme)) +
  labs(title = "simulation état du système",
       x = "temps",
       y = "état") +
  scale_y_continuous(breaks = c(0, 1), minor_breaks = c(0,1)) +
  scale_x_continuous(breaks = seq(0, 20, 5), minor_breaks = seq(0, 20, 1)) +
  facet_grid(num_systeme ~ .)
g1
```

```r
simu_T <- function(n){
  simu <- rep(-1, n)
  for(i in 1:n){
    simu[i] <- simu_etat_systeme(lambda, beta, t)[[2]]
  }
  return(simu)
}

set.seed(1)
n <- 10000
simu <- simu_T(n)

esperance <- mean(simu)

confint <- c(mean(simu) - 1.96 * sd(simu)/sqrt(n), mean(simu) +
               1.96 * sd(simu)/sqrt(n))

esperance
```

```
confint
```

```
nb.simu <- matrix(1:n, n, 1)
esp.cumul <- cumsum(simu)/nb.simu
df.si <- data.frame(nb.simu, esp.cumul)
ggplot(df.si,aes(x=nb.simu, y=esp.cumul))+
 geom_line()+
 xlab("Nombre simulations")+
 ylab("Moyenne des estimations")
```

Question 3

```
fonction_interval=function(n,inter,i){
  t=NULL
  d=NULL
  for (v in seq(1,n,inter)) {
    if(v==i){
      t[v]=1
    }
    else{
      t[v]=0
    }
  }
  t=t[!is.na(t)]
  w=sum(t)
  if(w==0){
    d=0
  }
  else{
    d=1
  }
  return(d)
}

phi2 <- function(a, n, lambda, beta,inter){
  t <- seq(0, a,length.out =n)
  y <- rep(0,n)
  u <- runif(10,0,1)
  x <- matrix(0, nrow = n, ncol = 10)

  for(i in 1:n){
    for(j in c(1,2,3,4,5,7,8,9,10)){
      if(t[i] < (lambda*(-log(u[j]))^(1/beta))){
        x[i,j] <- 1
```

```
      }
    }
  }
  d=0
  for(i in 1:n){
    if(t[i-d] < (lambda*(-log(u[6]))^(1/beta))){
      x[i,6] <- 1
    }
    if(fonction_interval(n,inter,i)==1 & x[i,6]==0){
      d <- i
      u[6] <- runif(1,0,1)
    }
  }
  # y[i] <- phi(x[i,])
  return(x)
}

phi3 <- function(a, n, lambda, beta,inter){
  y=NULL
  matrice=phi2(a, n, lambda, beta,inter)
  for (i in 1:n) {
    y[i]=phi(matrice[i,])
  }
  return(y)
}
```

```
a <- 20
n <- 20
lambda <- 15
beta <- 3
inter <- 4

set.seed(10560)
df2 <- data.frame(systeme = c(phi3(a,n,lambda,beta,inter),
                              phi3(a,n,lambda,beta,inter),
                              phi3(a,n,lambda,beta,inter)),
                  num_systeme = c(rep("systeme 1",n), rep("systeme 2",n),
                                  rep("systeme 3",n)),
                  t = rep(seq(0, a, length.out = n), 3))

g2 <- df2 %>%
  ggplot() +
  geom_line(aes(x = t, y = systeme, linetype = num_systeme)) +
  labs(title = "état du système réparable",
```

```
       x = "temps",
       y = "état") +
  scale_y_continuous(breaks = c(0, 1), minor_breaks = c(0,1)) +
  scale_x_continuous(breaks = seq(0, 20, 5), minor_breaks = seq(0, 20, 1)) +
  facet_grid(num_systeme ~ .)
g2
```

```
# Esperance de T
tp_i_de_panne <- function(a, n, lambda, beta,inter){
  y=NULL
  d=0
  matrice=phi2(a, n, lambda, beta,inter)
  for (i in 1:n) {
    y[i]=phi(matrice[i,])
  }
  for (i in n:1){
    if(y[i]==1 & i==n ){
      d=n
      break
    }
    if(y[i]==1){
      d=i+1
      break
    }
  }
  return(d)
}

nbtot_realisation_t=function(a, n, lambda, beta,inter,nbtot){
  vector=NULL
  for(i in 1:nbtot){
    vector[i]=tp_i_de_panne(a, n, lambda, beta,inter)
  }
  return(vector)
}

E_T=function(a, n, lambda, beta,inter,nbtot){
  t=nbtot_realisation_t(a, n, lambda, beta,inter,nbtot)
  moyenne=(1/nbtot)*sum(t)
  return(moyenne)
}
```

```
# Esperance de N
nb_intervention <- function(a, n, lambda, beta,inter){
  t <- seq(0, a,length.out =n)
```

```
  y <- rep(0,n)
  u <- runif(10,0,1)
  x <- matrix(0, nrow = n, ncol = 10)

  for(i in 1:n){
    for(j in c(1,2,3,4,5,7,8,9,10)){
      if(t[i] < (lambda*(-log(u[j]))^(1/beta))){
        x[i,j] <- 1
      }
    }
  }
  d=0
  intervention=0
  for(i in 1:n){
    if(t[i-d] < (lambda*(-log(u[6]))^(1/beta))){
      x[i,6] <- 1
    }
    if(fonction_interval(n,inter,i)==1){
      intervention <- intervention + 1
      if(x[i,6] == 0){
        d <- i
        u[6] <- runif(1,0,1)
      }
    }
  }
  return(intervention)
}

nb_realisation_intervention=function(a, n, lambda, beta,inter,nbtot){
  vector=NULL
  for(i in 1:nbtot){
    vector[i]=nb_intervention(a, n, lambda, beta,inter)
  }
  return(vector)
}

moyenne_intervention=function(a, n, lambda, beta,inter,nbtot){
  moyenne=(1/nbtot)*sum(nb_realisation_intervention(a, n, lambda, beta,inter,nbtot))
  return(moyenne)
}
```

```
# Fonction récompense
a_val <- 200
n_val <- 200
```

```r
lambda_val <- 15
beta_val <- 3
nbtot_val <- 100
cout_val <- 100
gain_val <- 250

set.seed(1)
recompense<-function(inter, a=a_val, n=n_val,lambda=lambda_val,
                     beta=beta_val,nbtot=nbtot_val,
                     cout=cout_val,gain=gain_val){
  gain_moyen <- gain*E_T(a, n, lambda, beta,inter,nbtot)
  cout_moyen <- cout*moyenne_intervention(a, n, lambda, beta,inter,nbtot)
  r <- gain_moyen - cout_moyen
  return(r)
}
solution <- optimize(recompense,c(0, 200),maximum = TRUE)
solution
```

```
## $maximum
## [1] 78.01667
##
## $objective
## [1] 2807.5
```

```r
a_val <- 50
n_val <- 50
lambda_val <- 15
beta_val <- 3
nbtot_val <- 100
cout_val <- 100
gain_val <- 250

set.seed(1)
value <- 0
for(i in 1:50){
  value[i] <- recompense(i)
}

df3 <- data.frame(value,
                  t = seq(0, a_val, length.out = n_val))

g3 <- df3 %>%
  ggplot() +
  geom_line(aes(x = t, y = value)) +
  labs(title = "fonction de récompense",
```

```
       x = "delta",
       y = "récompense")
g3
```