fiabilité

camille

21/11/2020

Contents

Introduction

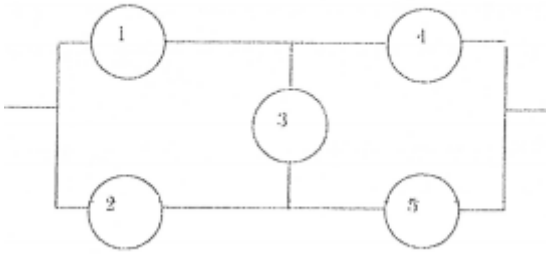
Étude d'un système complexe

Nous allons dans cette première partie de projet, étudier un système composé de deux blocks 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 block 1

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



Nous allons commencer par implémenter sa fonction de structure qui nous renverra l'état du block 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 avec les autres composants qui fonctionnent, on obtient le résultat suivant :

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

0

La fonction nous renvoie 0, donc le block 1 ne fonctionne pas. Ce qui est normal car si les deux premiers composants qui sont en parallèle 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 :

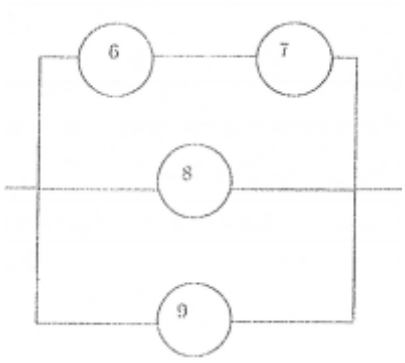
```
phi_1(c(1,0,0,1,0))
```

1

La fonction nous renvoie 1, donc le block 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 :



Nous allons implémenter sa fonction de structure avec le code ci-dessous :

```
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 block 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 :

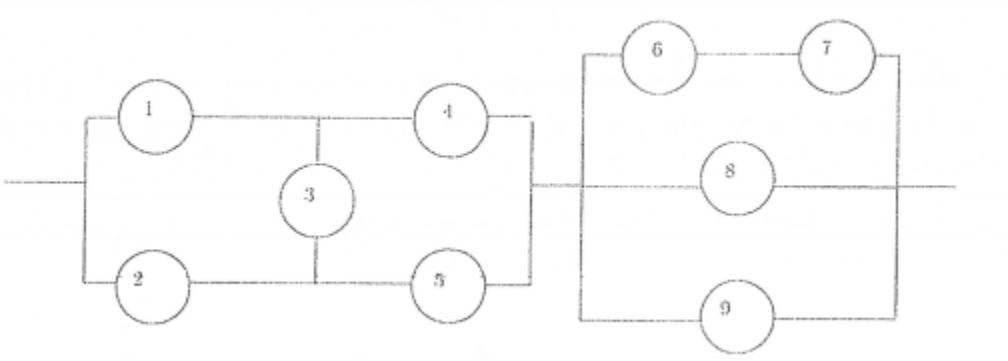
```
phi_2(c(0,0,1,0))
```

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 implémenté par le code suivant :

```
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 block 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.

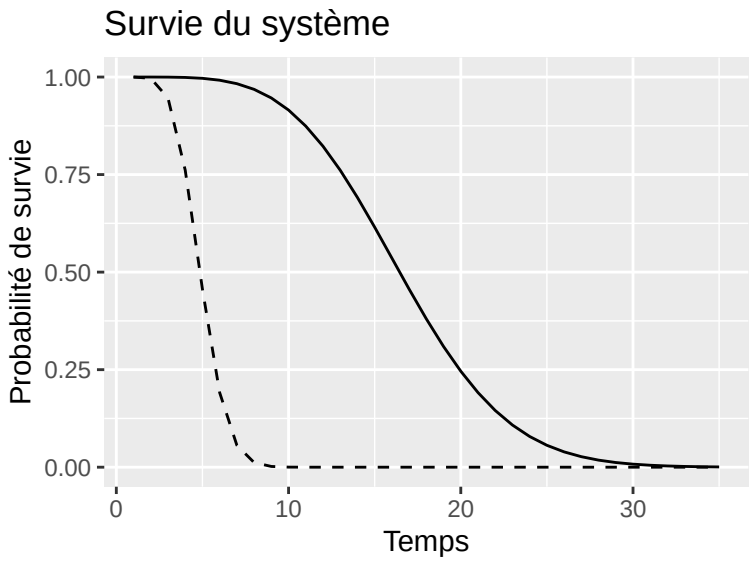
Simulations

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 composant.

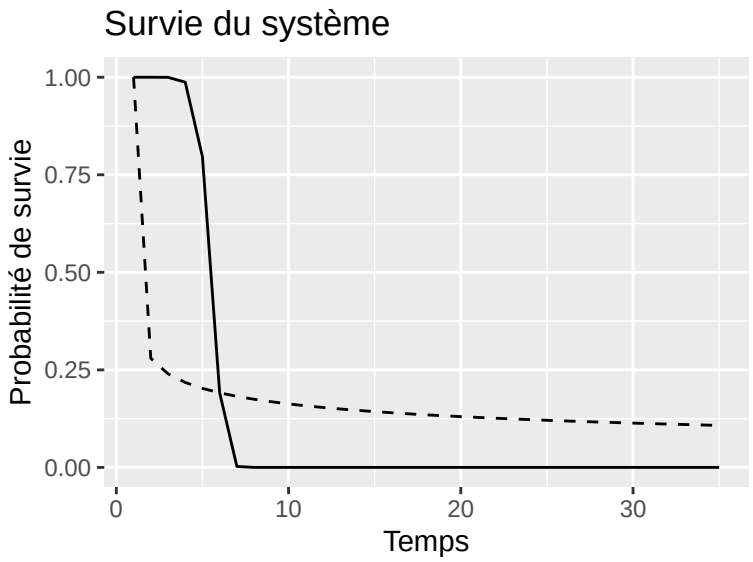
La courbe en pointillé représente la cas où $\lambda = 5$, pour tous les composant.

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 composant.

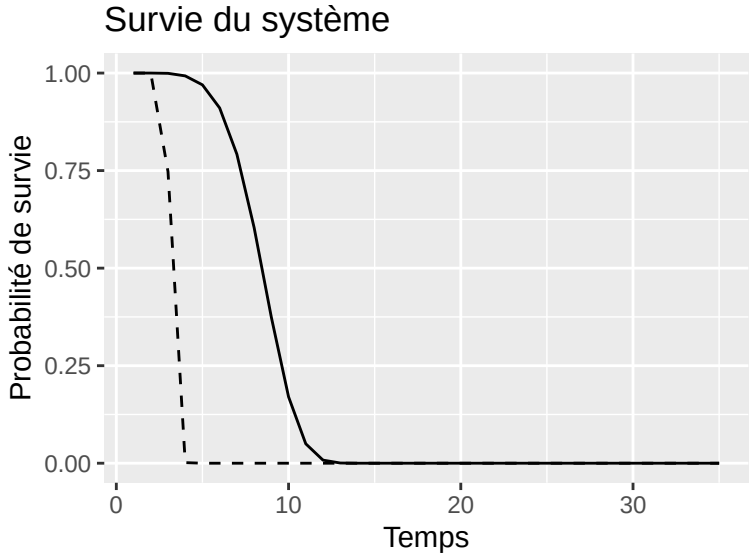
La courbe en pointillé représente la cas où $\beta = 0.1$, pour tous les composant.

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 pointillé 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 diminuait 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 diminition de probabilité de survie dans le temps entre ces deux simulations.

Étude de trajectoires pour d'autres fonctions

Implémentation de la fonction

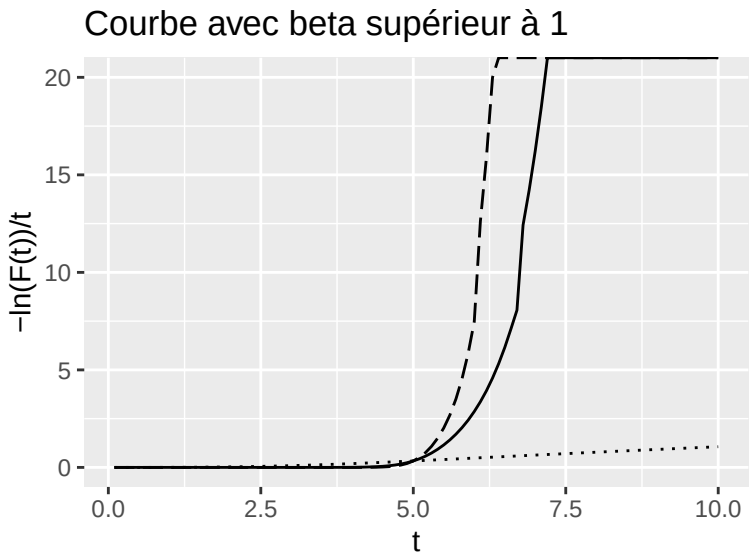
Nous allons implémenter une fonction $t \mapsto -ln\bar{F}(t)/t$. Le code est le suivant :

```
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.

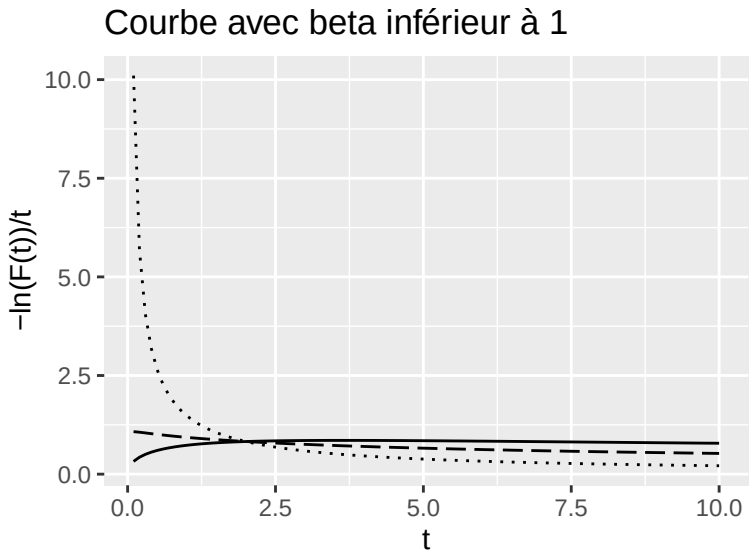
La courbe en pointillé représente la première simulation.

La courbe en trait-pointillé 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 pointillé représente la première simulation.

La courbe en trait-pointillé 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 utilisé 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 paramètre 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

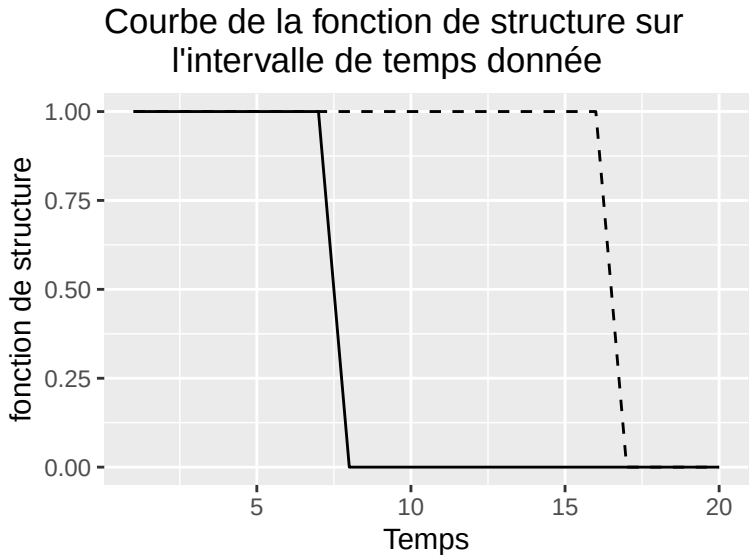
Pour une première simulation, on prend :

- $\lambda = 2$ 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 représenté par la courbe en trait plein. La deuxième simulation est représenté par la courbe en pointillé.



Pour la première simulation, on observe que a $t = 8$ le système tombe en panne.
Pour la première simulation, on observe que a $t = 17$ 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 auquelle 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 $t = 13$
Donc il vit 4.8484848 jours. (13 eme valeur de l'intervalle donnée)

Estimation de la durée de vie T1 du block 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 block 1 sur un intervalle de temps donnée) et une fonction qui renvoie l'espérence du temps de survie pour le block 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 block 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 block 2 sur un intervalle de temps donnée) et une fonction qui renvoie l'espérence du temps de survie pour le block 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 4.4444444 jours. (12 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)
```

```
## [1] 12.42
```

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 block 1

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

On implémente une fonction nomée *n_realisation_T_b*1 qui réalise plusieurs fois la fonction qui calcule la durée de vie du block 1 et d'enregistrer toutes les réalisations dans un vecteur.

On implémente aussi une fonction nomée *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 aléatoires 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 block 2

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

On implémente une fonction nomée *n_realisation_T_b*2 qui réalise plusieurs fois la fonction qui calcule la durée de vie du block 1 et d'enregistrer toutes les réalisations dans un vecteur.

On implémente aussi une fonction nomée *mu_*2 qui calcule la moyenne de toute ces réalisations.

Ces fonctions sont en annexe car elle sont assez similaire à celle du système et du block 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 vrai pour $\beta > 1$ mais pas pour $\beta \leq 1$

Pour béta 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 vérifié 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 réalisation.

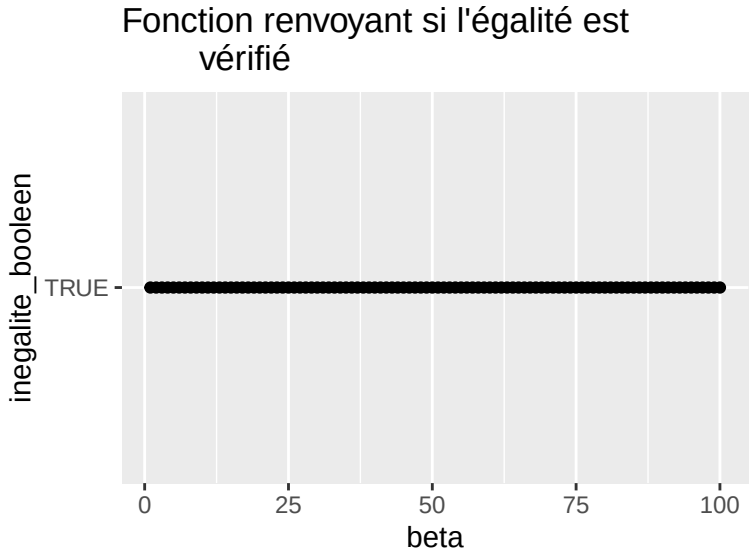
Le code le suivant :

```
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 :

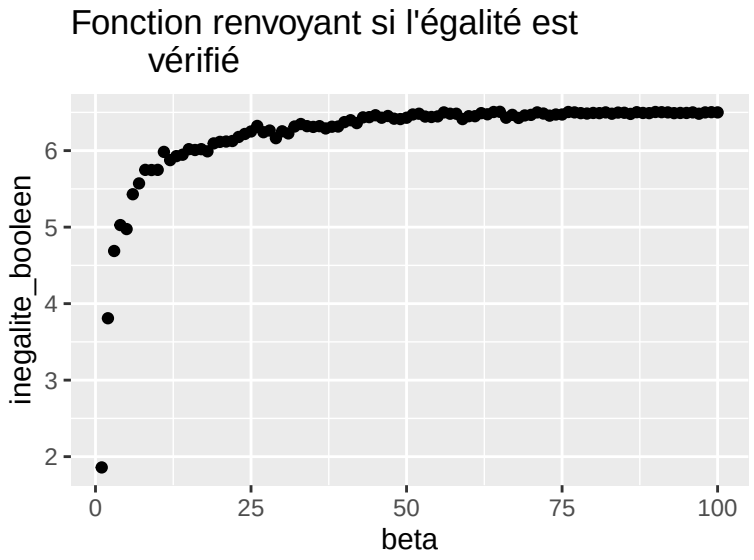


On peut aussi implémenter une autre fonction qui renvoie si $\mu - \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$

```
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 supérieur à 0 pour $\beta > 1$. Ainsi $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ est vérifié expérimentalement.

Pour béta inférieur à 1

Pour montrer que $\mu \geq \left(\frac{1}{\mu_1} + \frac{1}{\mu_2}\right)^{-1}$ n'est pas toujours vérifié 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 $\beta = 0.1 < 1$

Annexe

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
```

```r
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

```r
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
```

```r
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)
```

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)
}
```

Inégalité entre les espérences

```
g <- ggplot() +
  aes(x = 1:100, y = inegalite_booleen(1:100)) +
  geom_point()+
  labs(title="Fonction renvoyant si l'égalité est
        vérifié",
       x = "beta",
       y = "inegalite_booleen")
g
```

```
g <- ggplot() +
  aes(x = 1:100, y = inegalite(1:100)) +
  geom_point()+
  labs(title="Fonction renvoyant si l'égalité est
        vérifié",
       x = "beta",
       y = "inegalite_booleen")
g
```