Filtre tes connaissances

Conception de filtres pour le réduction du bruit du capteur de distance

Daniel Lavallée Mathieu Sévégny Tristan Lafontaine Émile Bois
Vincent Kilaknowvski

2024-10-31

Contents

Configuration de l'environnement

Lecture des données

```
## Fonction qui vérifie si un packet est installé et qui l'installe avant
## de le charger au besoin.
loadPackage <- function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package, quiet = TRUE)
    library(package, character.only = TRUE, quietly = TRUE)
  }
  else library(package, character.only = TRUE, quietly = TRUE)
}
```

```r
loadPackage("tidyverse")
loadPackage("ggplot2")
loadPackage("readxl")
loadPackage("Metrics")
loadPackage("signal")
loadPackage("TTR")
loadPackage("caret")
```

Lecture des données du capteur de distance (données réelle vs données bruitées).

```r
## Chemin vers les données
path_donnees_obstacle <- "../data/donnees_detecteur_obstacle(more).csv"

## Lecture des données
donnees_detecteur_obstacle <- read_csv(path_donnees_obstacle)
donnees_detecteur_obstacle <- donnees_detecteur_obstacle %>%
  rename(real_distance_cm = `Sample Name`)

## dt moyen
dt_moyen <- mean(donnees_detecteur_obstacle$dt)
```

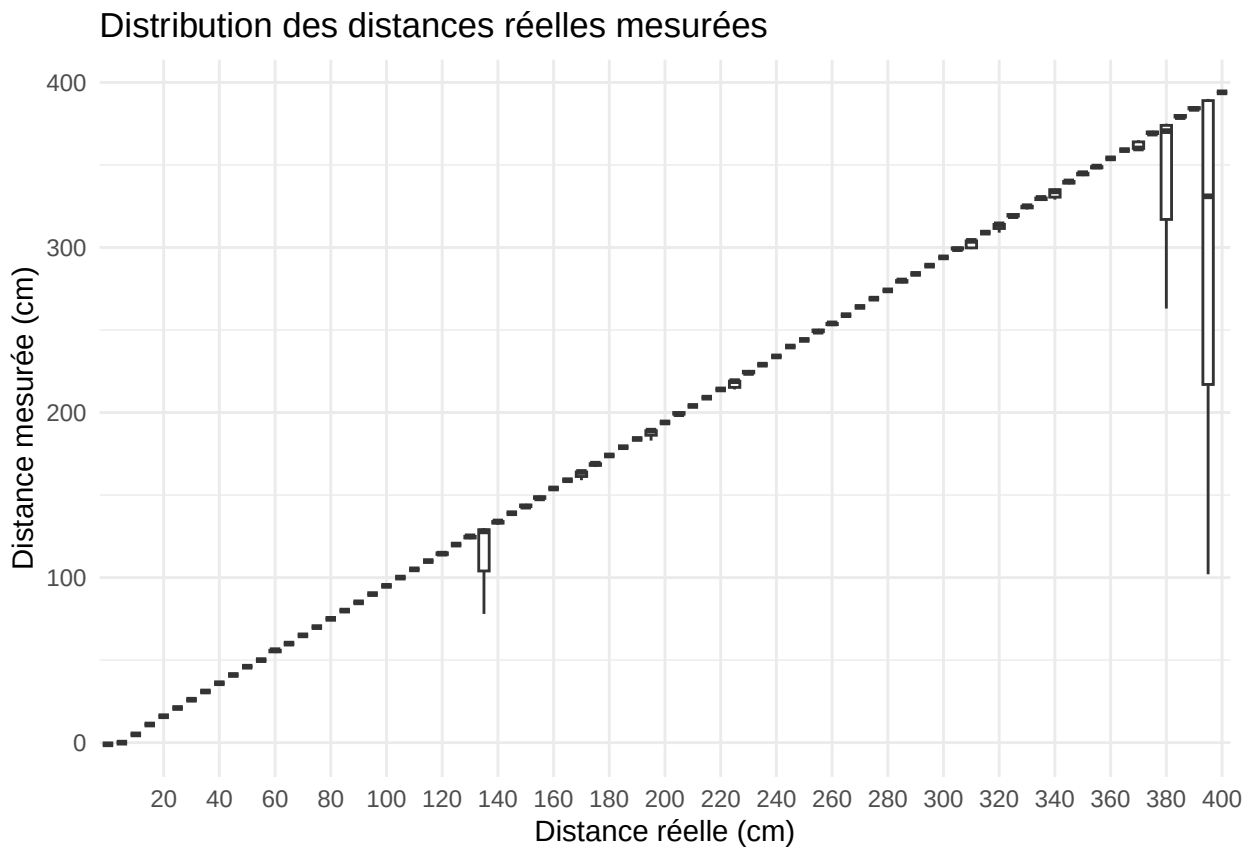
Nettoyage des données

```r
## Suppression des colonnes inutiles et transformation des données
donnees_detecteur_obstacle <- donnees_detecteur_obstacle %>%
  pivot_longer(cols = starts_with("Sample"), names_to = "Sample_No", values_to = "measured_distance_cm")
  select(., -dt, -Sample_No)

## Aggrégation des données par la moyenne des distances mesurées
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle %>%
  group_by(real_distance_cm) %>%
  summarise(measured_distance_cm = mean(measured_distance_cm, na.rm = TRUE))
```

Visualisation des données

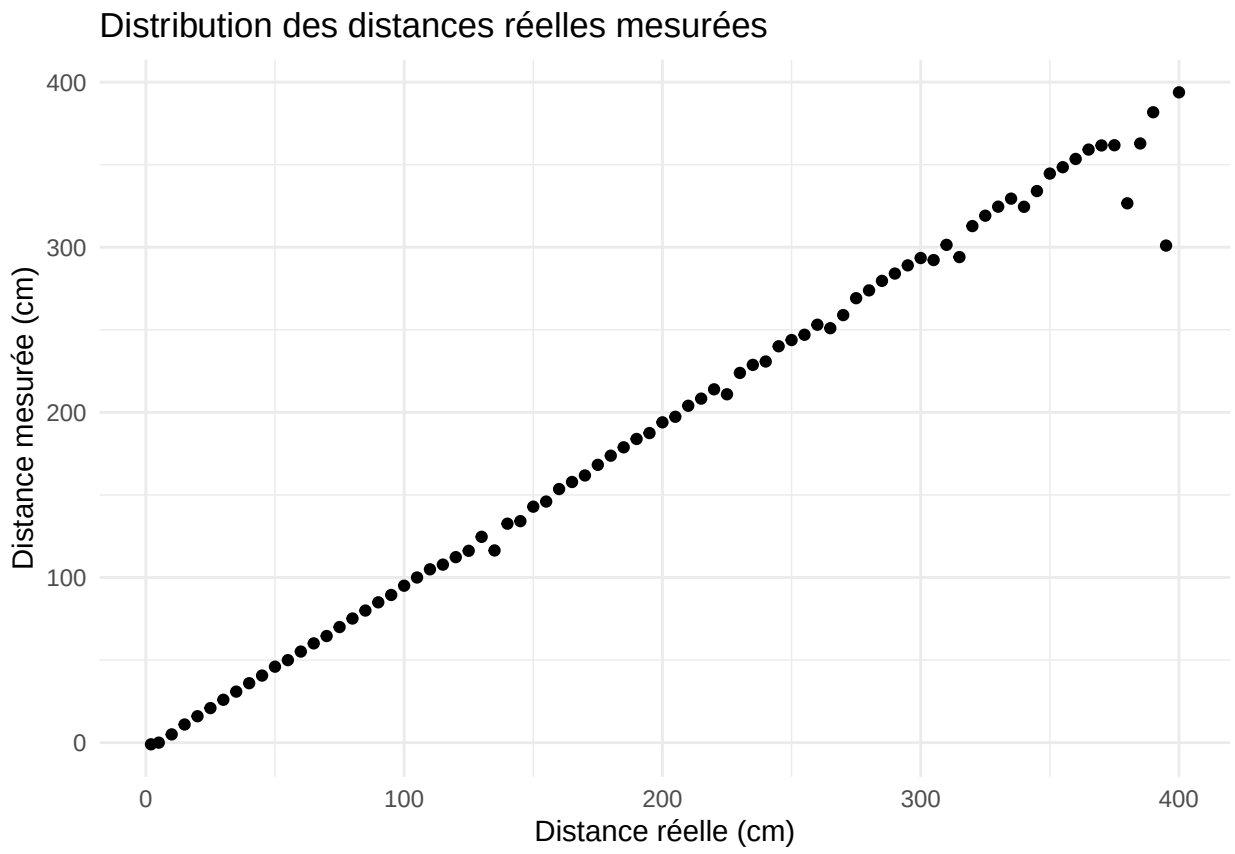
Distruibution des données bruitées.

```r
## Distribution des données
ggplot(donnees_detecteur_obstacle, aes(x = factor(real_distance_cm), y = measured_distance_cm)) +
  geom_boxplot(outlier.shape = NA) +
  labs(title = "Distribution des distances réelles mesurées",
       x = "Distance réelle (cm)",
       y = "Distance mesurée (cm)") +
  scale_x_discrete(breaks = seq(0, 400, by = 4)) +
  theme_minimal()
```



Données agrégées par la moyenne des distances mesurées.

```
## Distribution des données agrégées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = real_distance_cm, y = measured_distance_cm)) +
  geom_point() +
  labs(title = "Distribution des distances réelles mesurées",
       x = "Distance réelle (cm)",
       y = "Distance mesurée (cm)") +
  theme_minimal()
```



Conception d'un filtre Butterworth

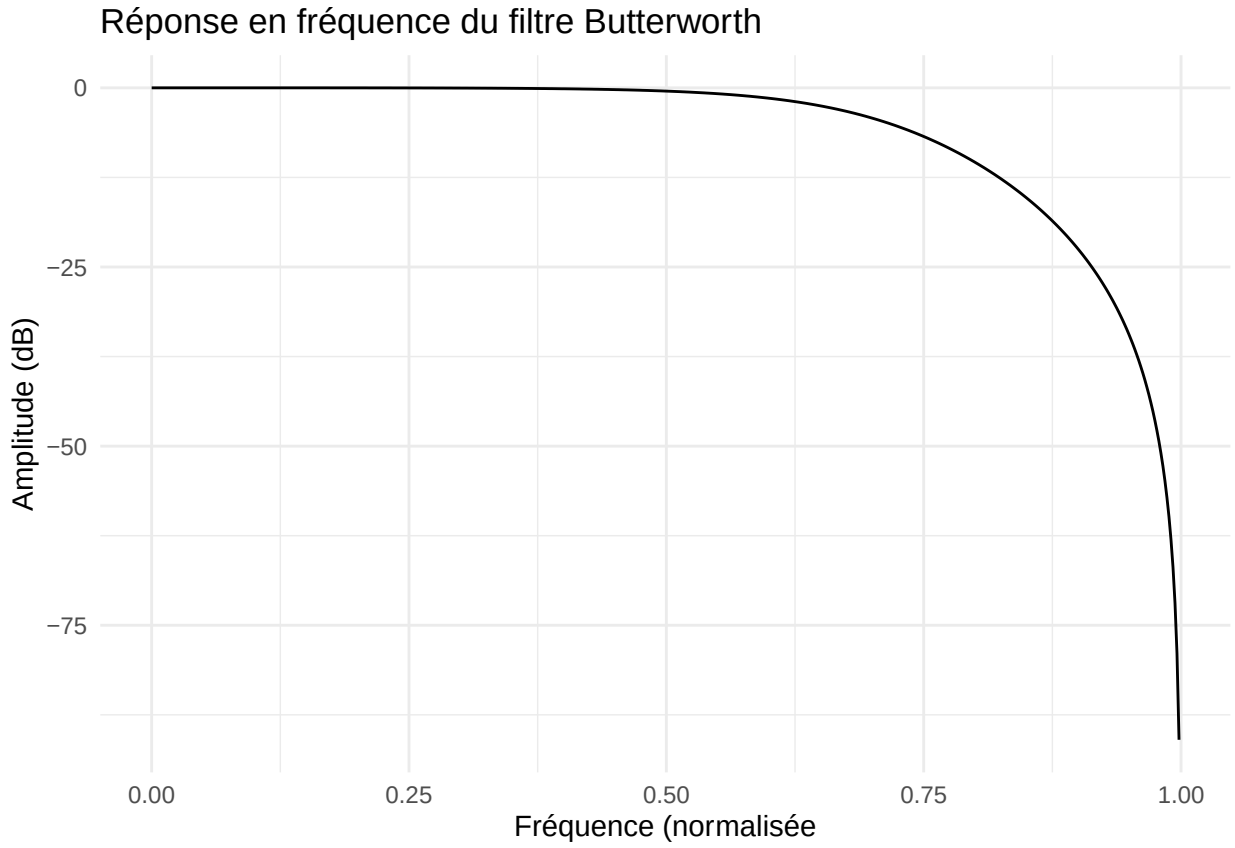
```
fe <- 1/dt_moyen          # Fréquence d'échantillonnage
fc <- fe/3                # Fréquence de coupure
order_butter <- 2         # Ordre du filtre
cutoff <- fc/(fe/2)       # Fréquence de coupure normalisée

# Conception du filtre Butterworth
butter_filter <- butter(order_butter, cutoff, type = "low")

# Calcul de la réponse en fréquence
freq_response <- freqz(butter_filter, Fs = 2)  # Fs = 2 pour des fréquences normalisées (entre 0 et 1)

## Affichage des coefficients du filtre Butterworth
print(paste("Le numérateur du filtre Butterworth est : ", butter_filter$b))
```

```
## [1] "Le numérateur du filtre Butterworth est :  0.465153077165047"
## [2] "Le numérateur du filtre Butterworth est :  0.930306154330093"
## [3] "Le numérateur du filtre Butterworth est :  0.465153077165047"
```

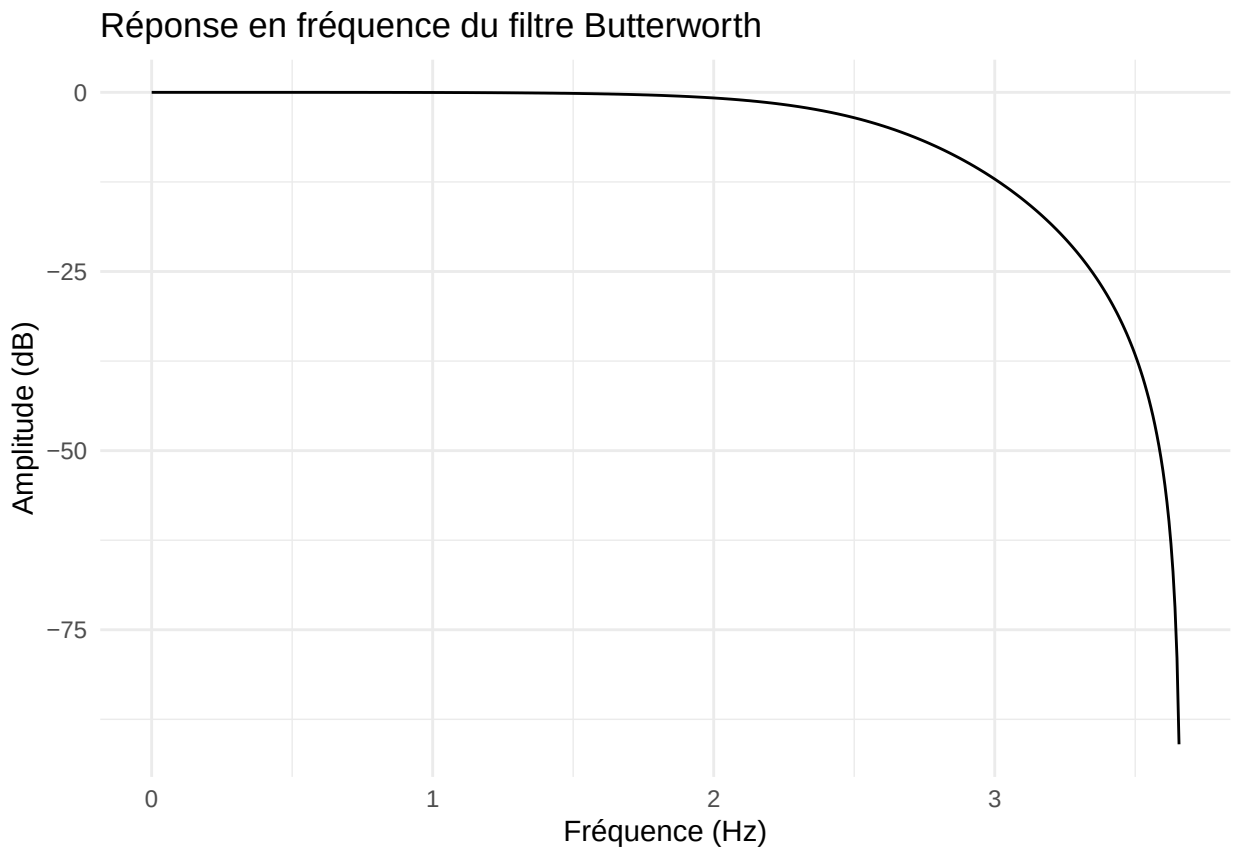
```
print(paste("Le dénominateur du filtre Butterworth est : ", butter_filter$a))
```

```
## [1] "Le dénominateur du filtre Butterworth est :  1"
## [2] "Le dénominateur du filtre Butterworth est :  0.620204102886728"
## [3] "Le dénominateur du filtre Butterworth est :  0.240408205773457"
```

```r
## Création d'un data frame pour la réponse en fréquence
# Fréquence normalisée
freq_response_butter <- data.frame(f = freq_response$f,
                h = abs(freq_response$h))
# Fréquence en Hz
freq_response_butter_hz <- data.frame(f = freq_response$f * (fe/2),
                h = abs(freq_response$h))

## Tracer la réponse en fréquence (normalisée)
ggplot(freq_response_butter, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (normalisée") +
  ylab("Amplitude (dB)") +
  ggtitle("Réponse en fréquence du filtre Butterworth") +
  theme_minimal()
```




```r
## Tracer la réponse en fréquence (en Hz)
ggplot(freq_response_butter_hz, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (Hz)") +
  ylab("Amplitude (dB)") +
  ggtitle("Réponse en fréquence du filtre Butterworth") +
  theme_minimal()
```

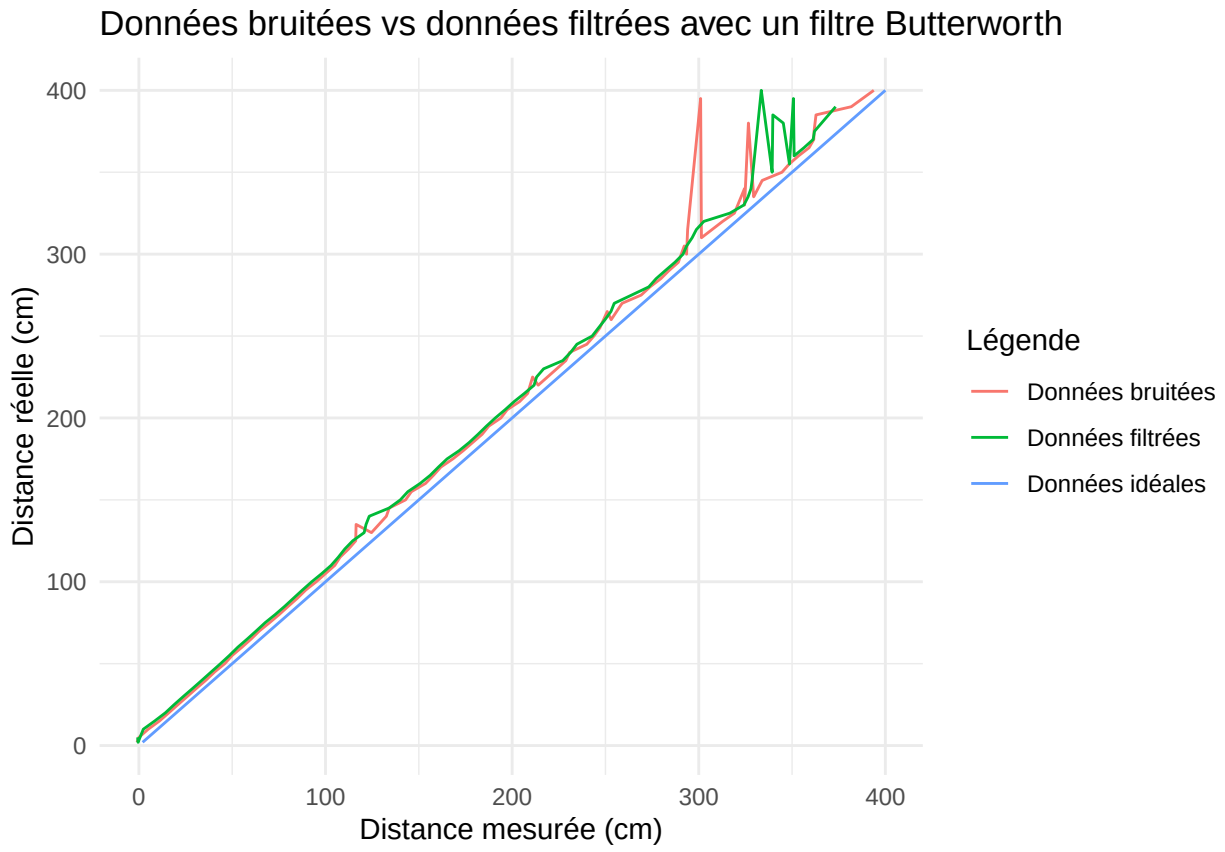


L'équation du **filtre de Butterworth** est alors donnée par :

$$h(s) = \frac{0.4651531s^2 + 0.9303062s + 0.4651531}{s^2 + 0.6202041s + 0.2404082}$$

Appliquons le filtrage sur les données bruitées.

```
# Filtrage des données bruitées
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(measured_distance_cm_butter = signal::filter(butter_filter, measured_distance_cm))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = measured_distance_cm_butter, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un filtre Butterworth",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  theme_minimal()
```



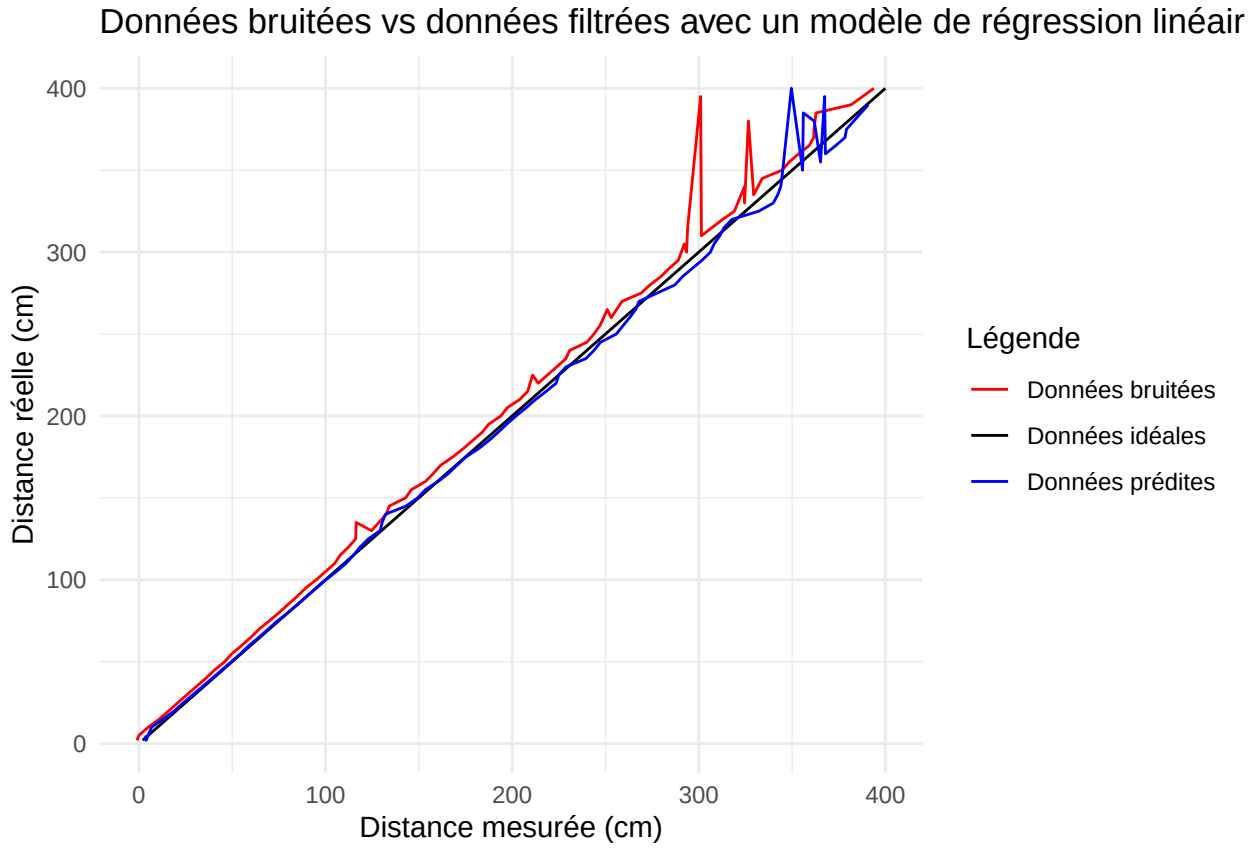

```
## Création du modèle de régression linéaire
modele_butter <- lm(real_distance_cm ~ measured_distance_cm_butter, data = donnees_detecteur_obstacle_ag

## Prédiction des valeur réelles avec le filtre de Butterworth
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_butter = predict(modele_butter))
```

L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 4.3171359 + 1.0354036x$$

Voici le modèle de prévision pour la régression linéaire avec le filtre Butterworth.

```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_butter, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Conception d'un filtre Chebyshev de type 1

```
order_chebyshev1 <- 2                # Ordre du filtre Chebyshev de type 1
ripple <- 0.5                        # Ondulation maximale en dB

# Conception du filtre Chebyshev de type 1
chebyshev1_filter <- cheby1(order_chebyshev1, ripple, cutoff, type = "low")

# Affichage des coefficients du filtre Chebyshev de type 1
print(paste("Le numérateur du filtre Chebyshev de type 1 est : ", chebyshev1_filter$b))
```

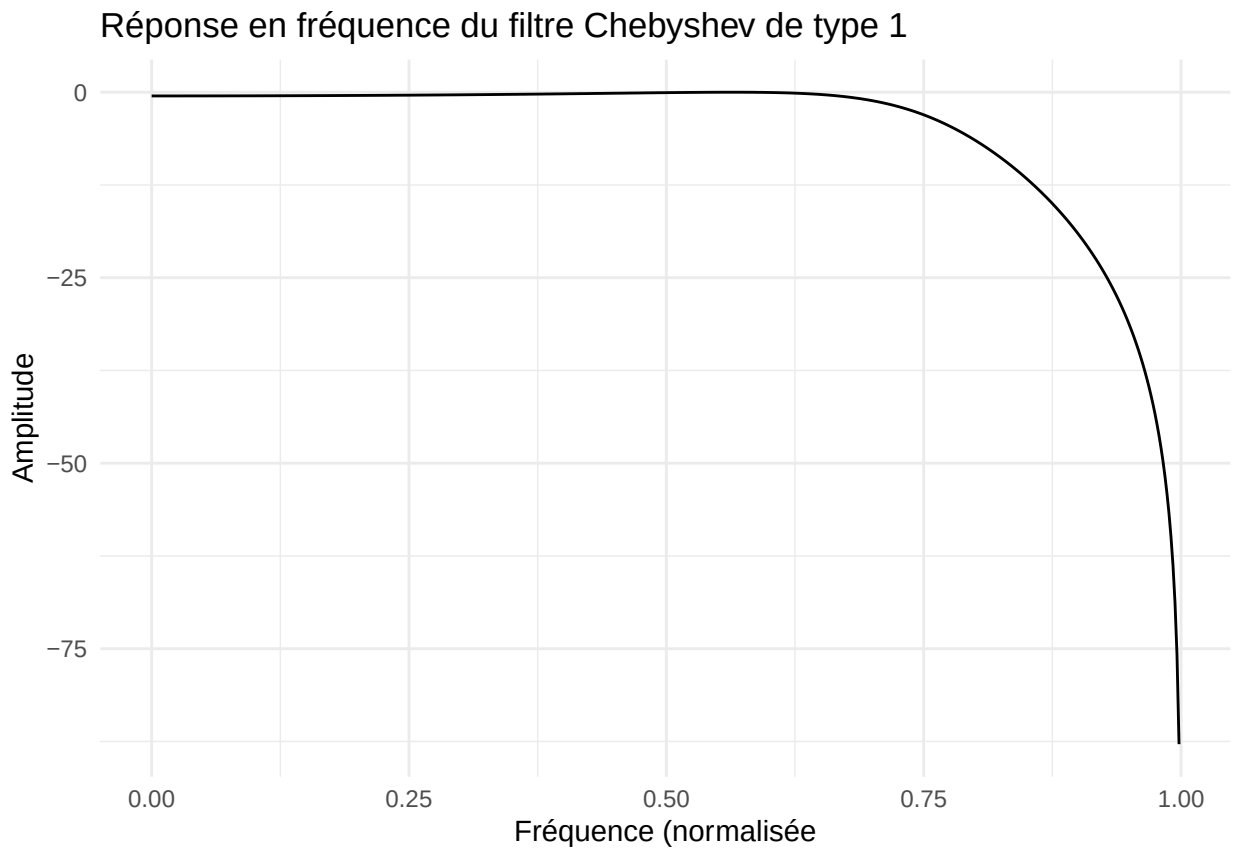
```
## [1] "Le numérateur du filtre Chebyshev de type 1 est :  0.535574540620439"
## [2] "Le numérateur du filtre Chebyshev de type 1 est :  1.07114908124088"
## [3] "Le numérateur du filtre Chebyshev de type 1 est :  0.535574540620439"
```

```
print(paste("Le dénominateur du filtre Chebyshev de type 1 est : ", chebyshev1_filter$a))
```

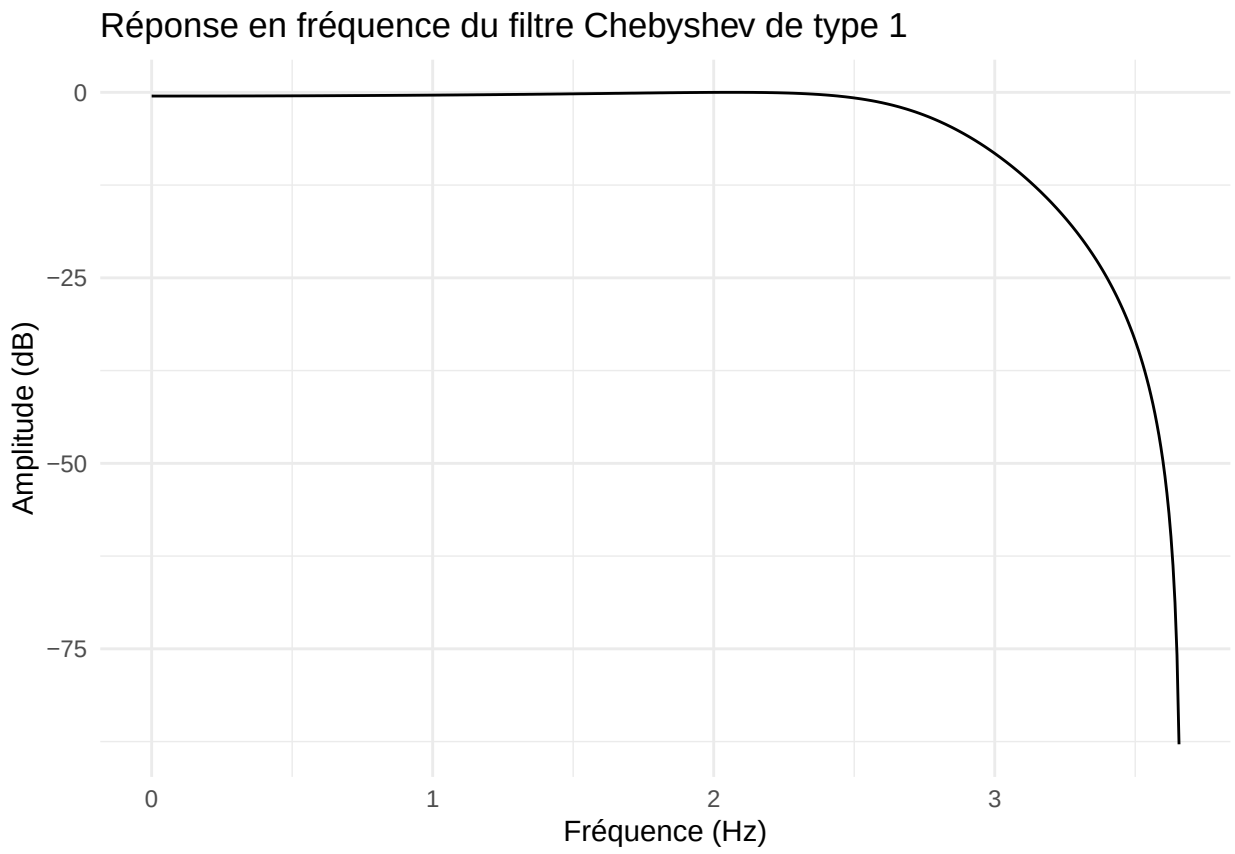
```
## [1] "Le dénominateur du filtre Chebyshev de type 1 est :  1"
## [2] "Le dénominateur du filtre Chebyshev de type 1 est :  0.885175597605059"
## [3] "Le dénominateur du filtre Chebyshev de type 1 est :  0.3840617114442"
```

```
# Réponse en fréquence
freq_response_chebyshev1 <- freqz(chebyshev1_filter, Fs = 2)
freq_response_chebyshev1 <- data.frame(f = freq_response_chebyshev1$f,
                h = abs(freq_response_chebyshev1$h))
freq_response_chebyshev1_hz <- data.frame(f = freq_response_chebyshev1$f * (fe/2),
```

```
                h = abs(freq_response_chebyshev1$h))

## Tracer la réponse en fréquence (normalisée)
ggplot(freq_response_chebyshev1, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (normalisée") +
  ylab("Amplitude") +
  ggtitle("Réponse en fréquence du filtre Chebyshev de type 1") +
  theme_minimal()
```




```
## Tracer la réponse en fréquence (en Hz)
ggplot(freq_response_chebyshev1_hz, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (Hz)") +
  ylab("Amplitude (dB)") +
  ggtitle("Réponse en fréquence du filtre Chebyshev de type 1") +
  theme_minimal()
```

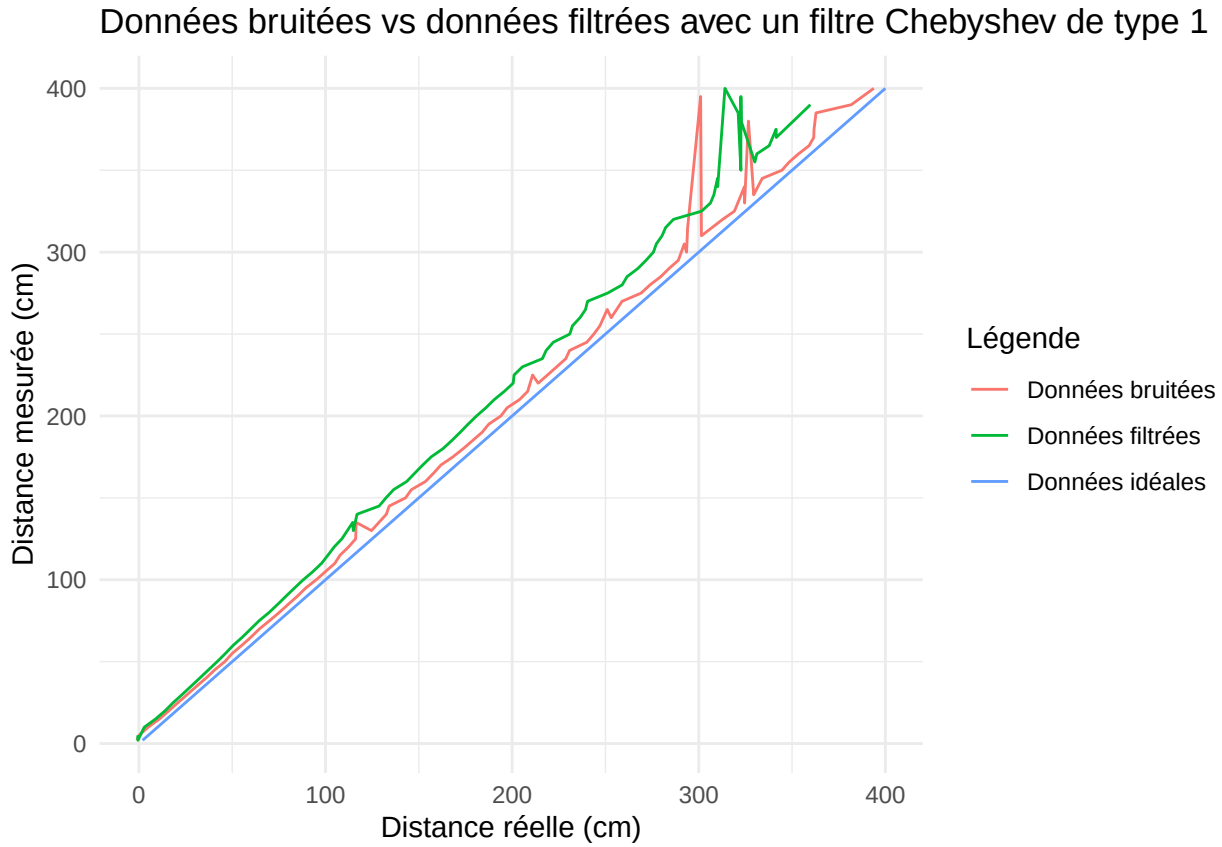


L'équation du **filtre de Chebyshev de type 1** est alors donnée par :

$$h(s) = \frac{0.5355745s^2 + 1.0711491s + 0.5355745}{s^2 + 0.8851756s + 0.3840617}$$

Appliquons maintenant le filtre de Chebyshev de type 1 sur les données bruitées.

```
# Filtrage des données bruitées
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(measured_distance_cm_chebyshev1 = signal::filter(chebyshev1_filter, measured_distance_cm))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = measured_distance_cm_chebyshev1, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un filtre Chebyshev de type 1",
       x = "Distance réelle (cm)",
       y = "Distance mesurée (cm)",
       color = "Légende") +
  theme_minimal()
```



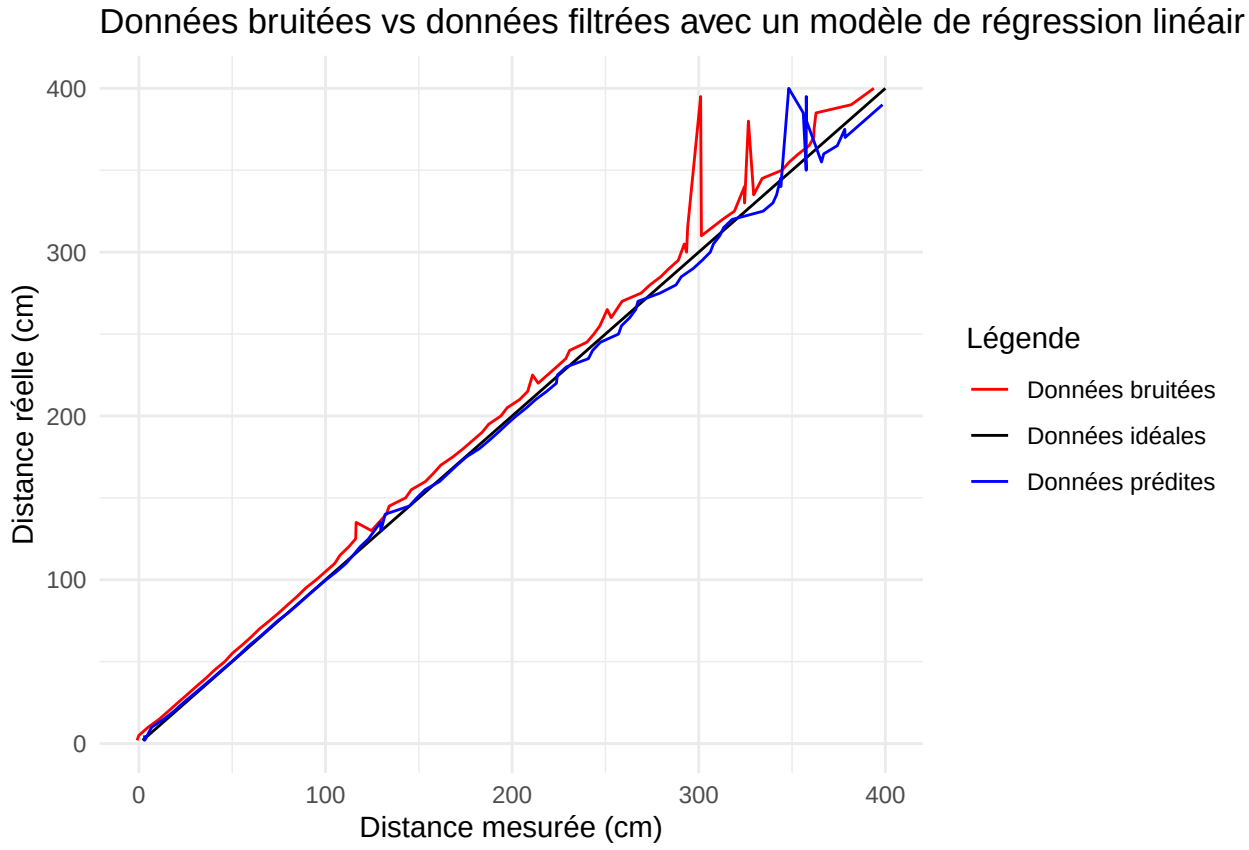

```
## Création du modèle de régression linéaire avec le filtre de Chebyshev de type 1
modele_chebyshev1 <- lm(real_distance_cm ~ measured_distance_cm_chebyshev1, data = donnees_detecteur_obs

## Prédiction des valeur réelles avec le filtre de Chebyshev de type 1
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_chebyshev1 = predict(modele_chebyshev1))
```

L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 3.6791309 + 1.0971551x$$

Voici le modèle de prévision pour la régression linéaire avec le filtre Chebyshev de type 1.

```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_chebyshev1, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Conception d'un filtre Chebyshev de type 2

```
order_chebyshev2 <- 2                # Ordre du filtre Chebyshev de type 2
stopband_attenuation <- 30           # Atténuation du bande d'arrêt en dB

# Conception du filtre Chebyshev de type 2
chebyshev2_filter <- cheby2(order_chebyshev2, stopband_attenuation, cutoff, type = "low")

# Affichage des coefficients du filtre Chebyshev de type 2
print(paste("Le numérateur du filtre Chebyshev de type 2 est : ", chebyshev2_filter$b))
```

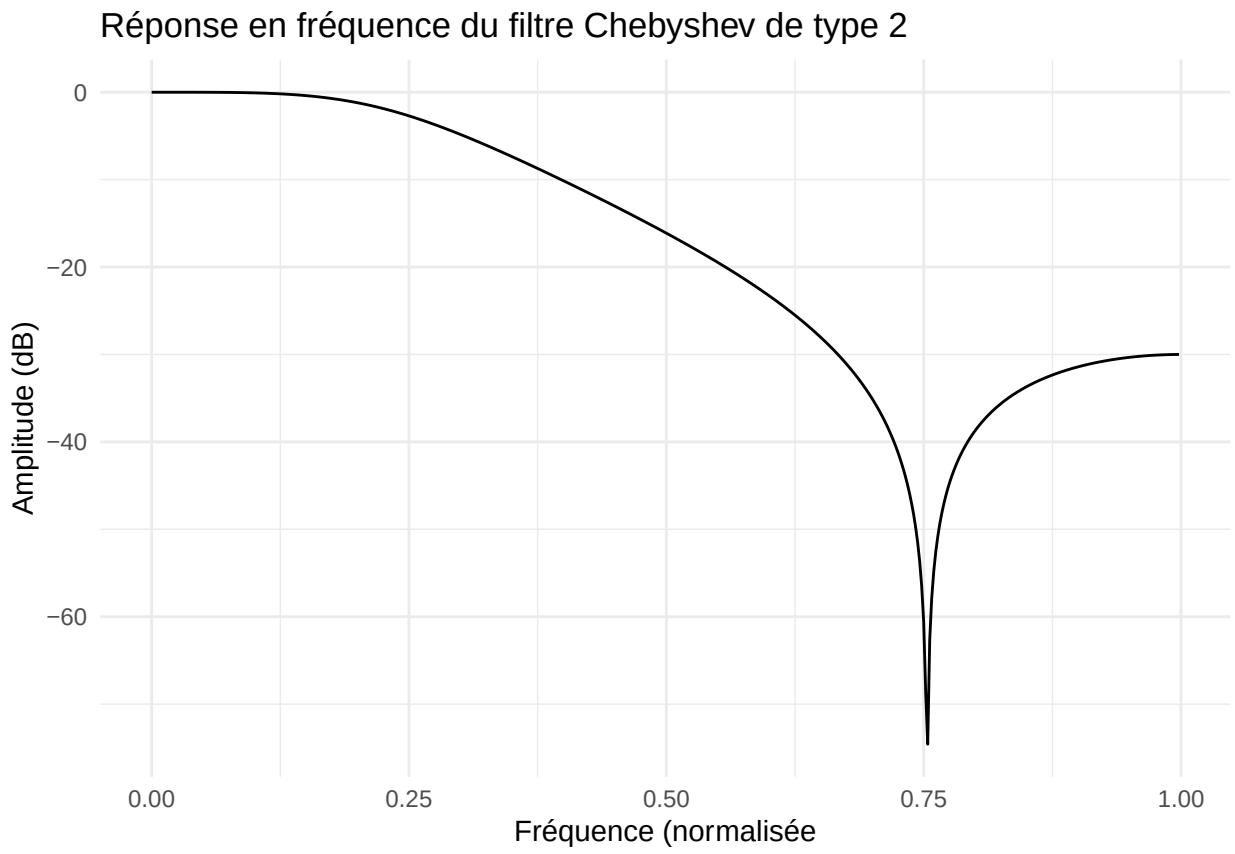
```
## [1] "Le numérateur du filtre Chebyshev de type 2 est :  0.123255993256761"
## [2] "Le numérateur du filtre Chebyshev de type 2 est :  0.176079990366802"
## [3] "Le numérateur du filtre Chebyshev de type 2 est :  0.123255993256761"
```

```
print(paste("Le dénominateur du filtre Chebyshev de type 2 est : ", chebyshev2_filter$a))
```

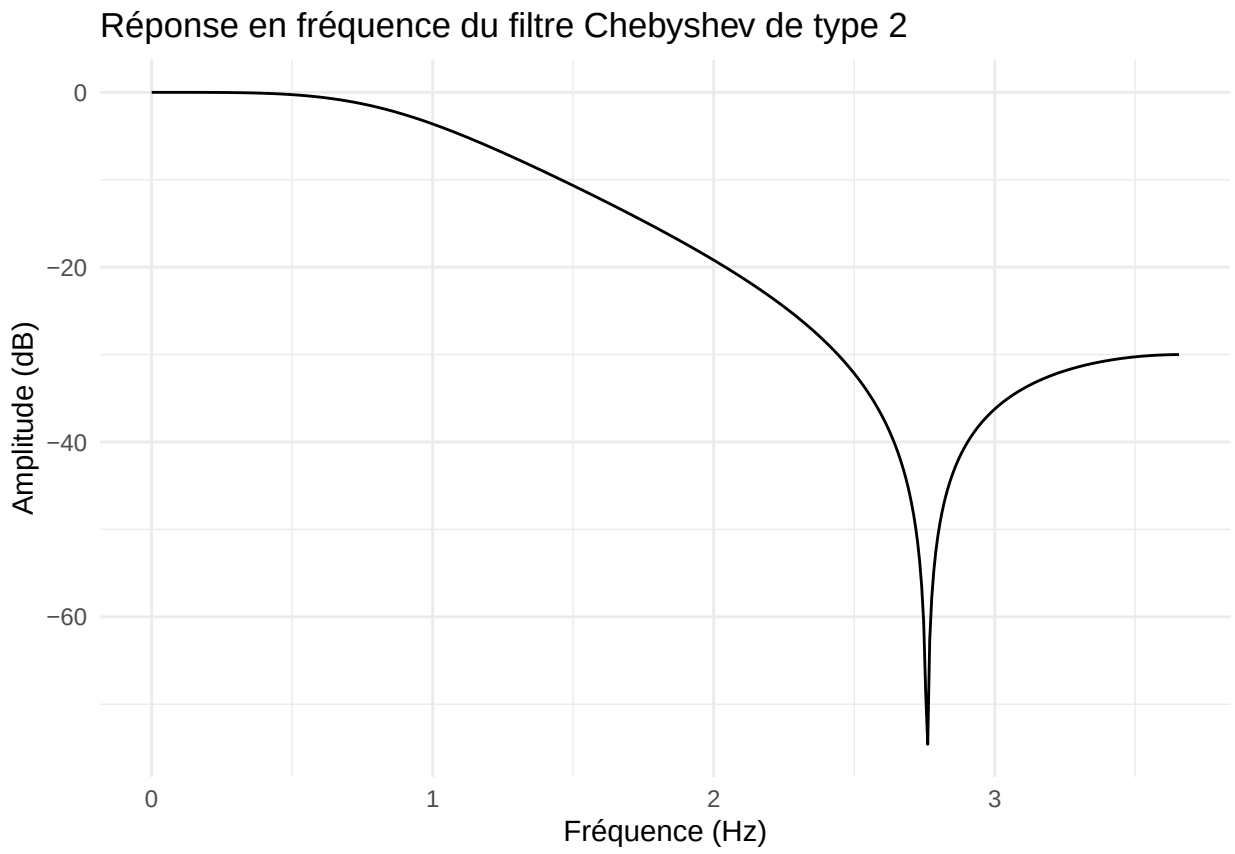
```
## [1] "Le dénominateur du filtre Chebyshev de type 2 est :  1"
## [2] "Le dénominateur du filtre Chebyshev de type 2 est :  -0.902331651439041"
## [3] "Le dénominateur du filtre Chebyshev de type 2 est :  0.324923628319367"
```

```
# Réponse en fréquence
freq_response_chebyshev2 <- freqz(chebyshev2_filter, Fs = 2)
freq_response_chebyshev2 <- data.frame(f = freq_response_chebyshev2$f,
                h = abs(freq_response_chebyshev2$h))
freq_response_chebyshev2_hz <- data.frame(f = freq_response_chebyshev2$f * (fe/2),
```

```
                h = abs(freq_response_chebyshev2$h))

## Tracer la réponse en fréquence (normalisée)
ggplot(freq_response_chebyshev2, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (normalisée") +
  ylab("Amplitude (dB)") +
  ggtitle("Réponse en fréquence du filtre Chebyshev de type 2") +
  theme_minimal()
```




```
## Tracer la réponse en fréquence (en Hz)
ggplot(freq_response_chebyshev2_hz, aes(x = f, y = 20 * log10(h))) +
  geom_line() +
  xlab("Fréquence (Hz)") +
  ylab("Amplitude (dB)") +
  ggtitle("Réponse en fréquence du filtre Chebyshev de type 2") +
  theme_minimal()
```

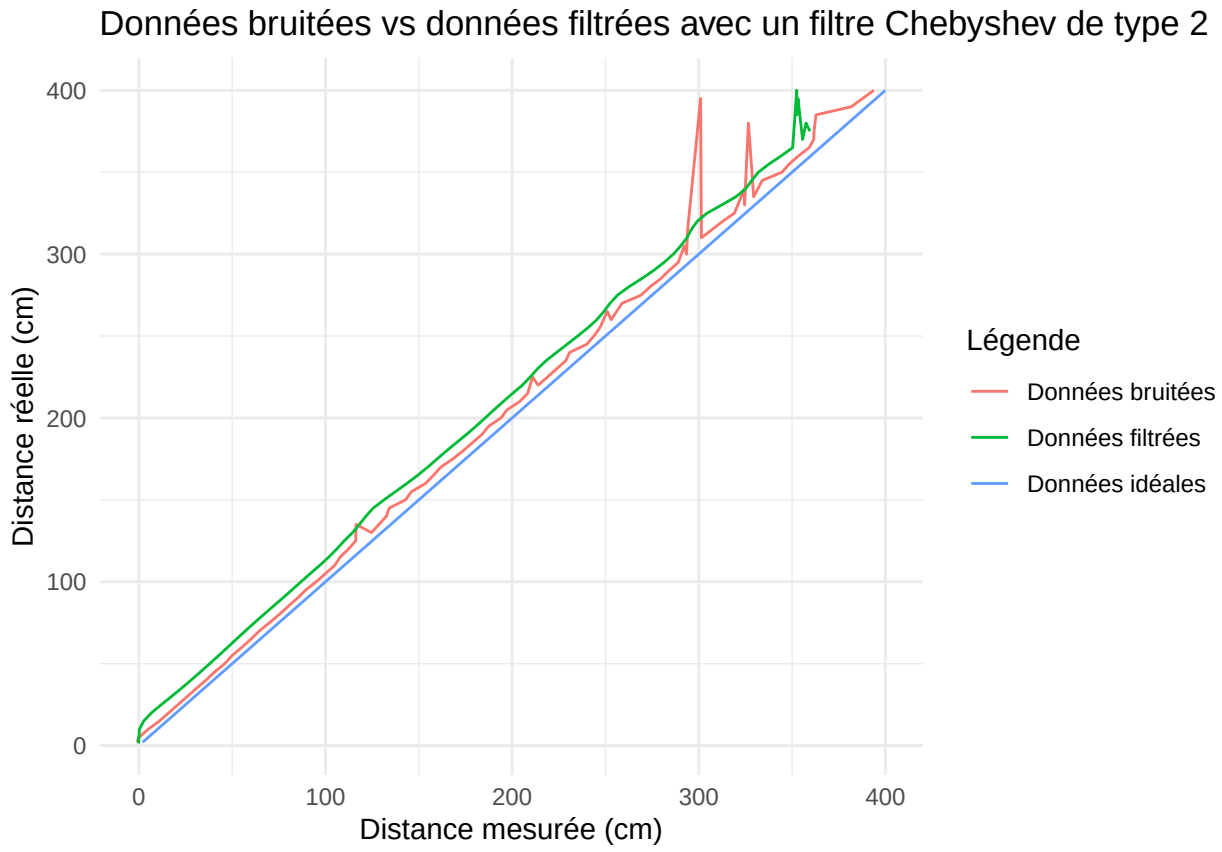


L'équation du **filtre de Chebyshev de type 2** est alors donnée par :

$$h(s) = \frac{0.123256s^2 + 0.17608s + 0.123256}{s^2 + -0.9023317s + 0.3249236}$$

Appliquons maintenant le filtre de Chebyshev de type 2 sur les données bruitées.

```
# Filtrage des données bruitées
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(measured_distance_cm_chebyshev2 = signal::filter(chebyshev2_filter, measured_distance_cm))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = measured_distance_cm_chebyshev2, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un filtre Chebyshev de type 2",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  theme_minimal()
```



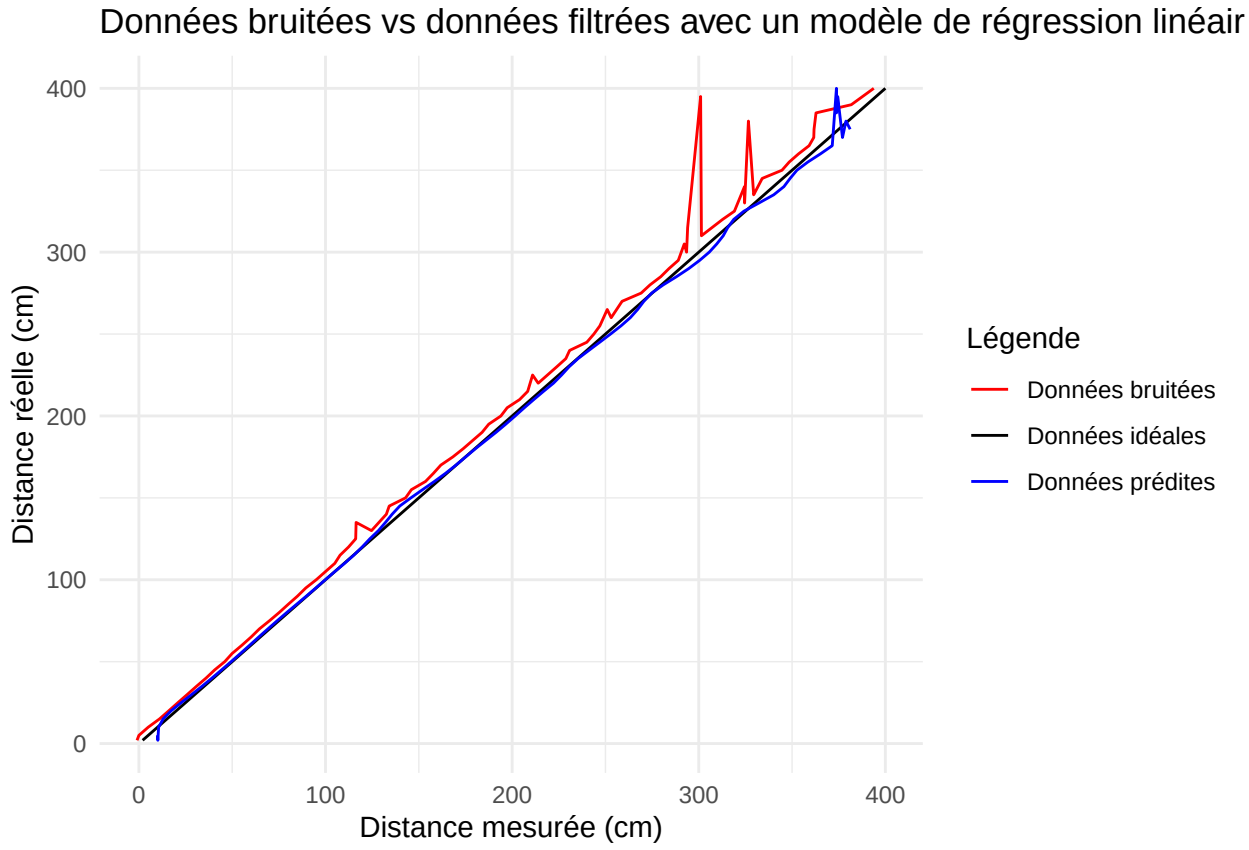

```
## Création du modèle de régression linéaire avec le filtre de Chebyshev de type 2
modele_chebyshev2 <- lm(real_distance_cm ~ measured_distance_cm_chebyshev2, data = donnees_detecteur_obs

## Prédiction des valeur réelles avec le filtre de Chebyshev de type 2
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_chebyshev2 = predict(modele_chebyshev2))
```

L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 10.2824305 + 1.0311763x$$

Voici le modèle de prévision pour la régression linéaire avec le filtre Chebyshev de type 2.

```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_chebyshev2, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Conception d'un filtre elliptique

```
order_elliptique <- 2                # Ordre du filtre elliptique
ripple_passband <- 0.5               # Ondulation maximale en dB dans la bande de passage
stopband_attenuation_elliptique <- 40 # Atténuation du bande d'arrêt en dB

# Conception du filtre elliptique
elliptic_filter <- ellip(order_elliptique, ripple_passband, stopband_attenuation_elliptique, cutoff, ty

# Affichage des coefficients du filtre elliptique
print(paste("Le numérateur du filtre elliptique est : ", elliptic_filter$b))
```

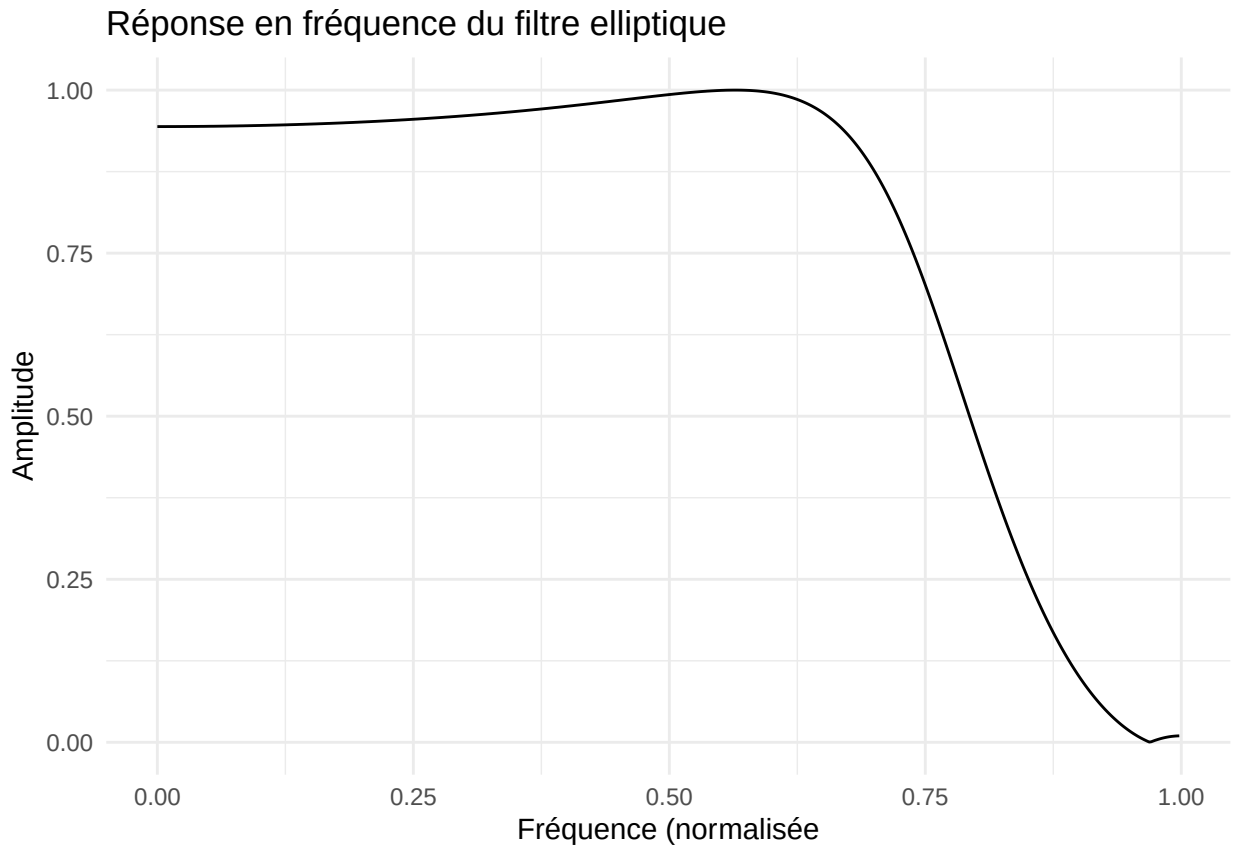
```
## [1] "Le numérateur du filtre elliptique est :  0.538503349959469"
## [2] "Le numérateur du filtre elliptique est :  1.07201963425761"
## [3] "Le numérateur du filtre elliptique est :  0.53850334995947"
```

```
print(paste("Le dénominateur du filtre elliptique est : ", elliptic_filter$a))
```

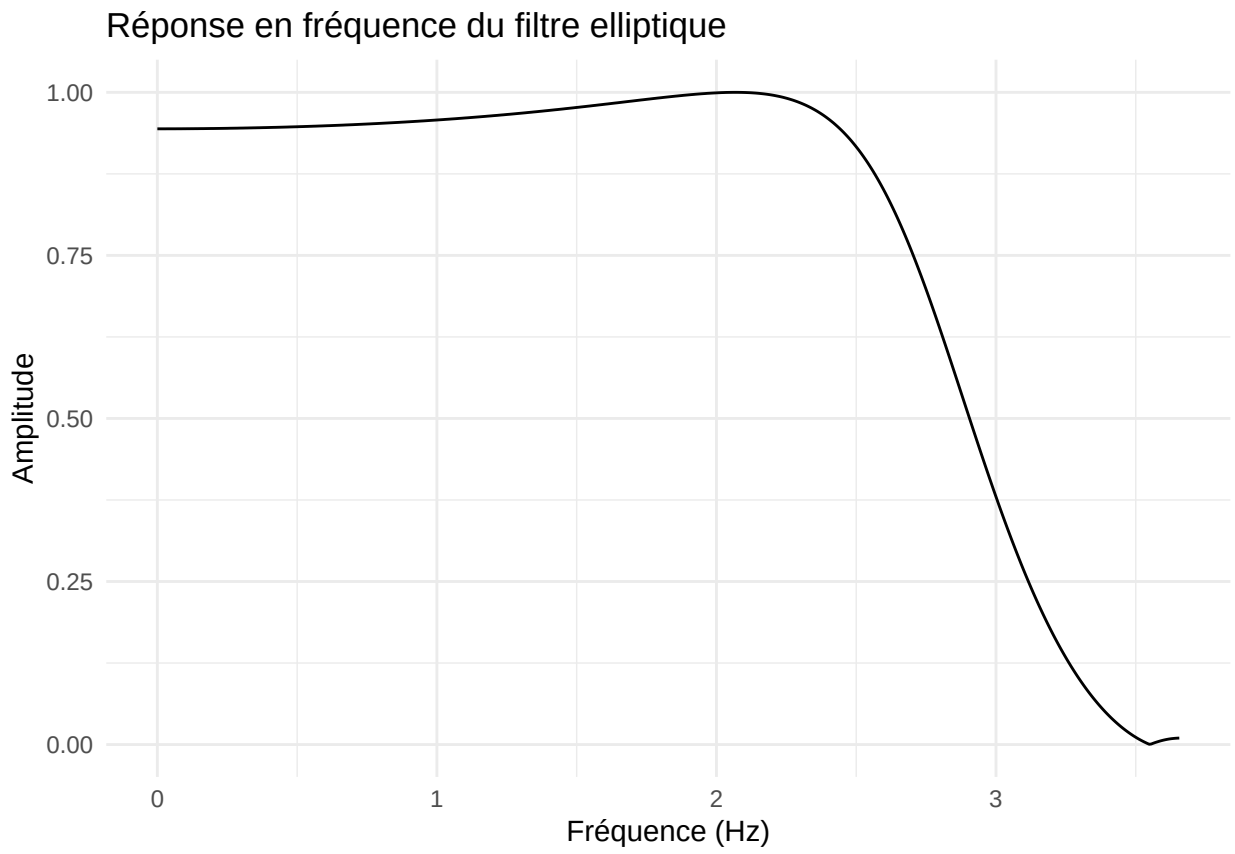
```
## [1] "Le dénominateur du filtre elliptique est :  1"
## [2] "Le dénominateur du filtre elliptique est :  0.888820394688326"
## [3] "Le dénominateur du filtre elliptique est :  0.387543755292276"
```

```
# Réponse en fréquence
freq_response_elliptic <- freqz(elliptic_filter, Fs = 2)
freq_response_elliptic <- data.frame(f = freq_response_elliptic$f,
                h = abs(freq_response_elliptic$h))
```

```r
freq_response_elliptic_hz <- data.frame(f = freq_response_elliptic$f * (fe/2),
                h = abs(freq_response_elliptic$h))

## Tracer la réponse en fréquence (normalisée)
ggplot(freq_response_elliptic, aes(x = f, y = h)) +
  geom_line() +
  xlab("Fréquence (normalisée") +
  ylab("Amplitude") +
  ggtitle("Réponse en fréquence du filtre elliptique") +
  theme_minimal()
```




```r
## Tracer la réponse en fréquence (en Hz)
ggplot(freq_response_elliptic_hz, aes(x = f, y = h)) +
  geom_line() +
  xlab("Fréquence (Hz)") +
  ylab("Amplitude") +
  ggtitle("Réponse en fréquence du filtre elliptique") +
  theme_minimal()
```

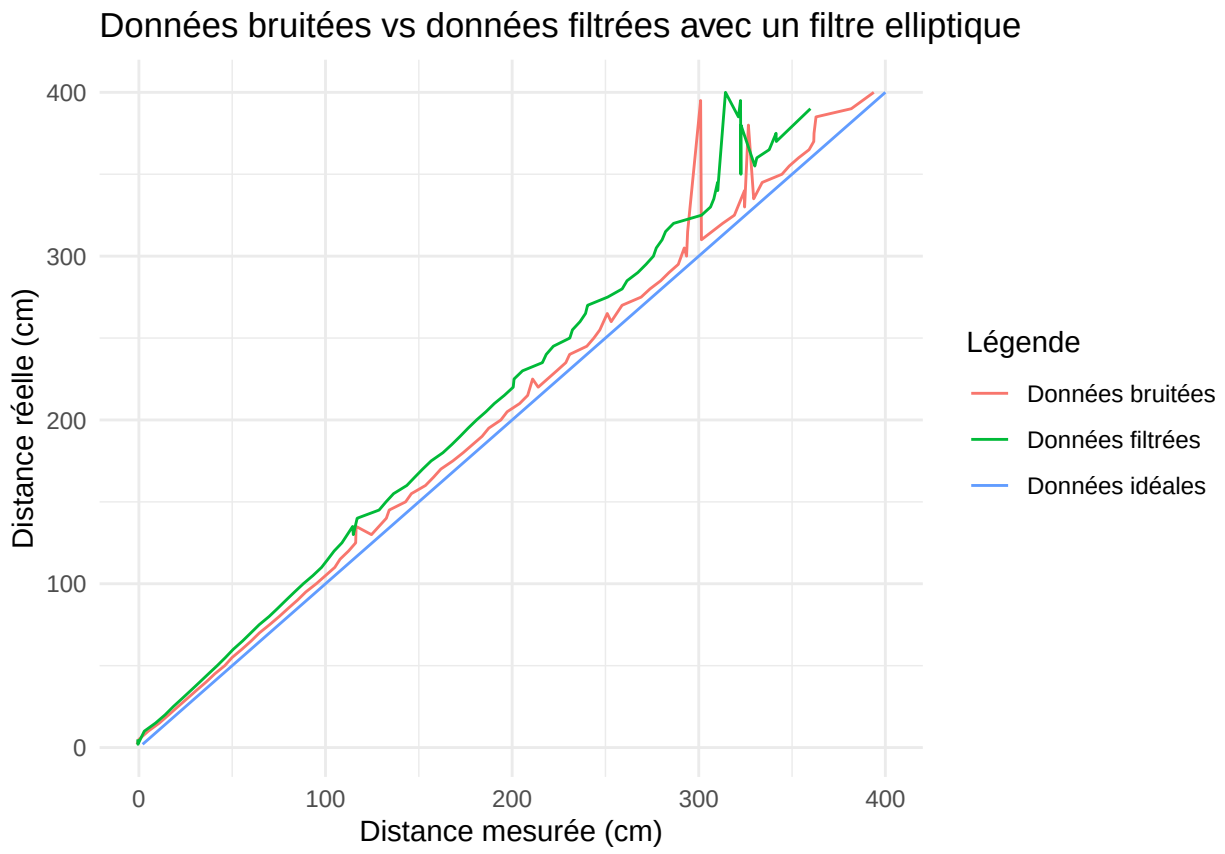


L'équation du **filtre elliptique** est alors donnée par :

$$h(s) = \frac{0.5385033s^2 + 1.0720196s + 0.5385033}{s^2 + 0.8888204s + 0.3875438}$$

Appliquons maintenant le filtre elliptique sur les données bruitées.

```
# Filtrage des données bruitées
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(measured_distance_cm_elliptic = signal::filter(elliptic_filter, measured_distance_cm))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = measured_distance_cm_elliptic, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un filtre elliptique",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  theme_minimal()
```



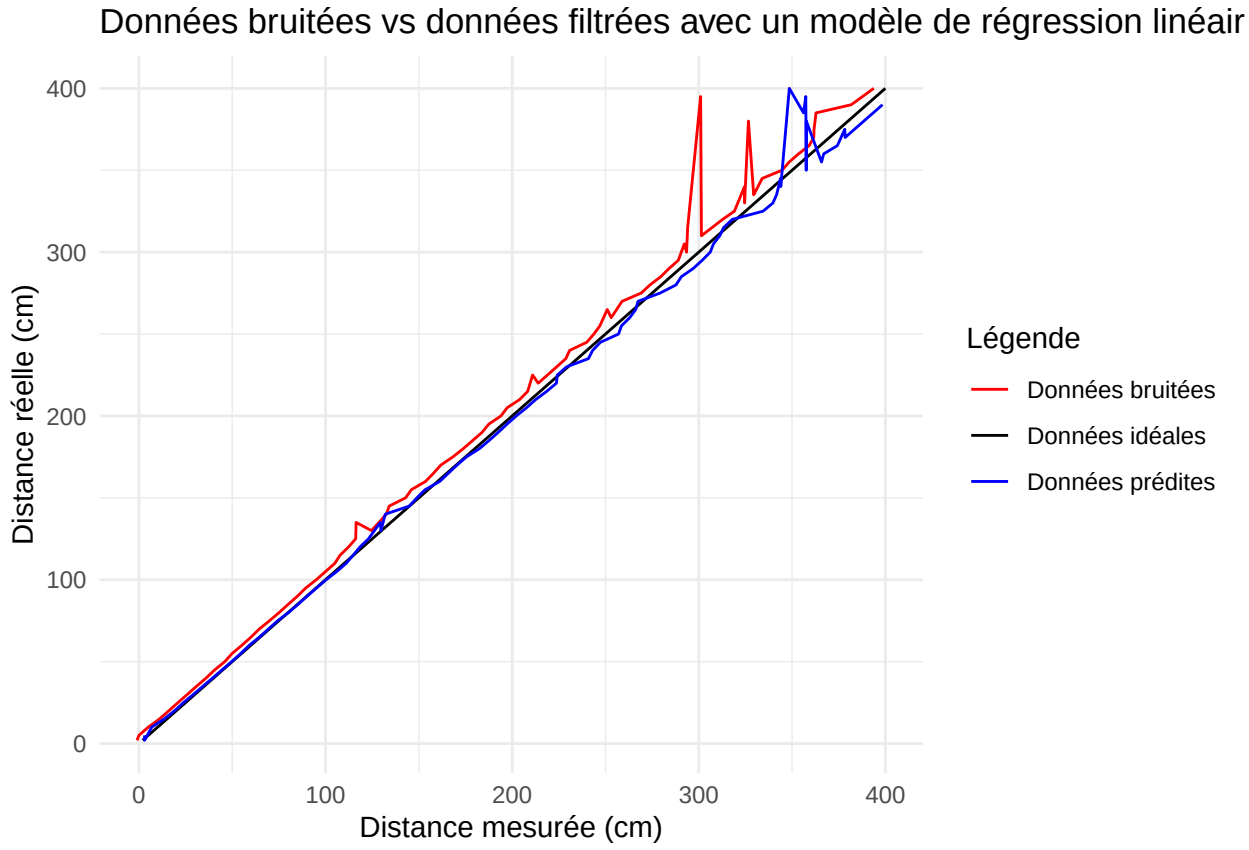

```
## Création du modèle de régression linéaire avec le filtre elliptique
modele_elliptic <- lm(real_distance_cm ~ measured_distance_cm_elliptic, data = donnees_detecteur_obstacl

## Prédiction des valeur réelles avec le filtre elliptique
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_elliptic = predict(modele_elliptic))
```

L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 3.6667733 + 1.0971535x$$

Voici le modèle de prévision pour la régression linéaire avec le filtre elliptique.

```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_elliptic, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Création d'un modèle de régression linéaire sans filtre

```
modele_regression_lineaire <- lm(measured_distance_cm ~ real_distance_cm, data = donnees_detecteur_obsta

modele_regression_lineaire$coefficients
```

```
##      (Intercept) real_distance_cm
##       -1.6671026        0.9638523
```

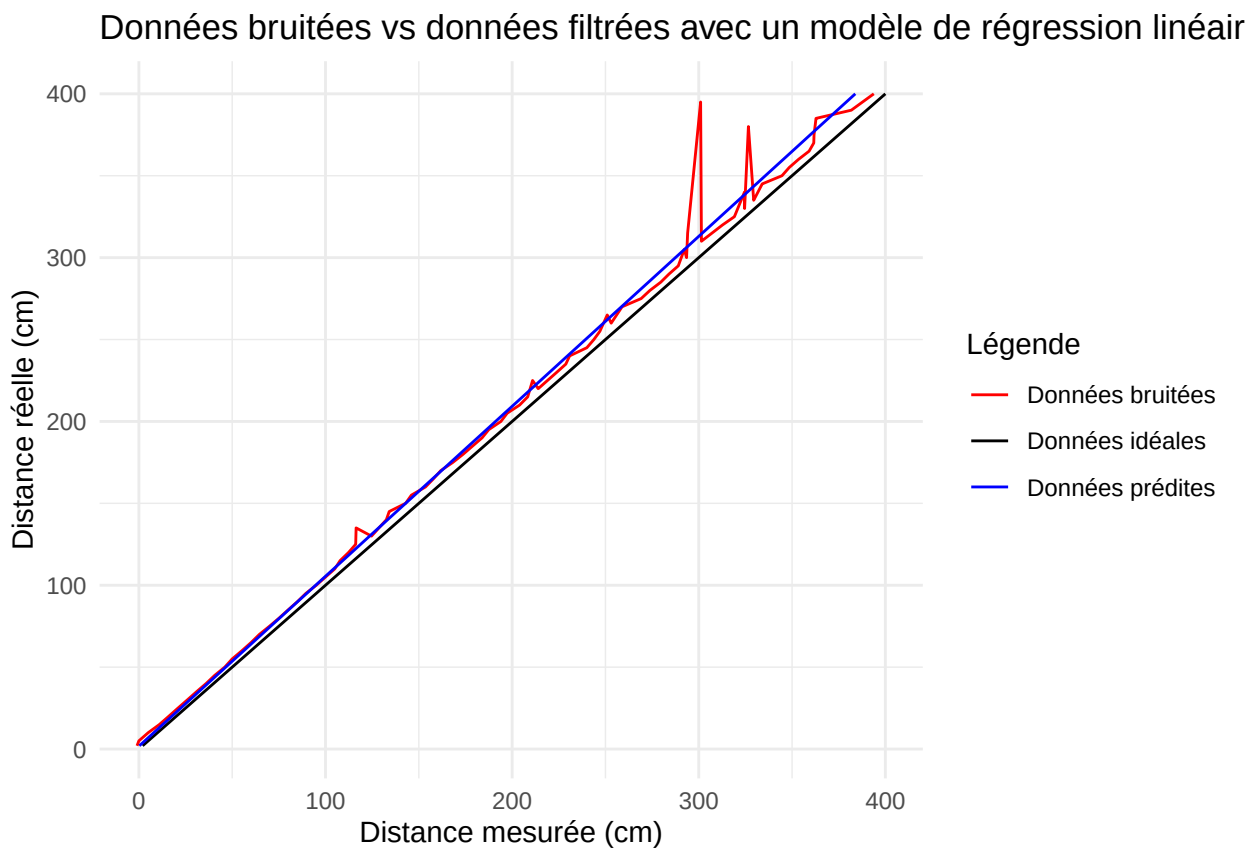
L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = -1.6671026 + 0.9638523x$$

Appliquons maintenant le modèle de régression linéaire sur les données bruitées.

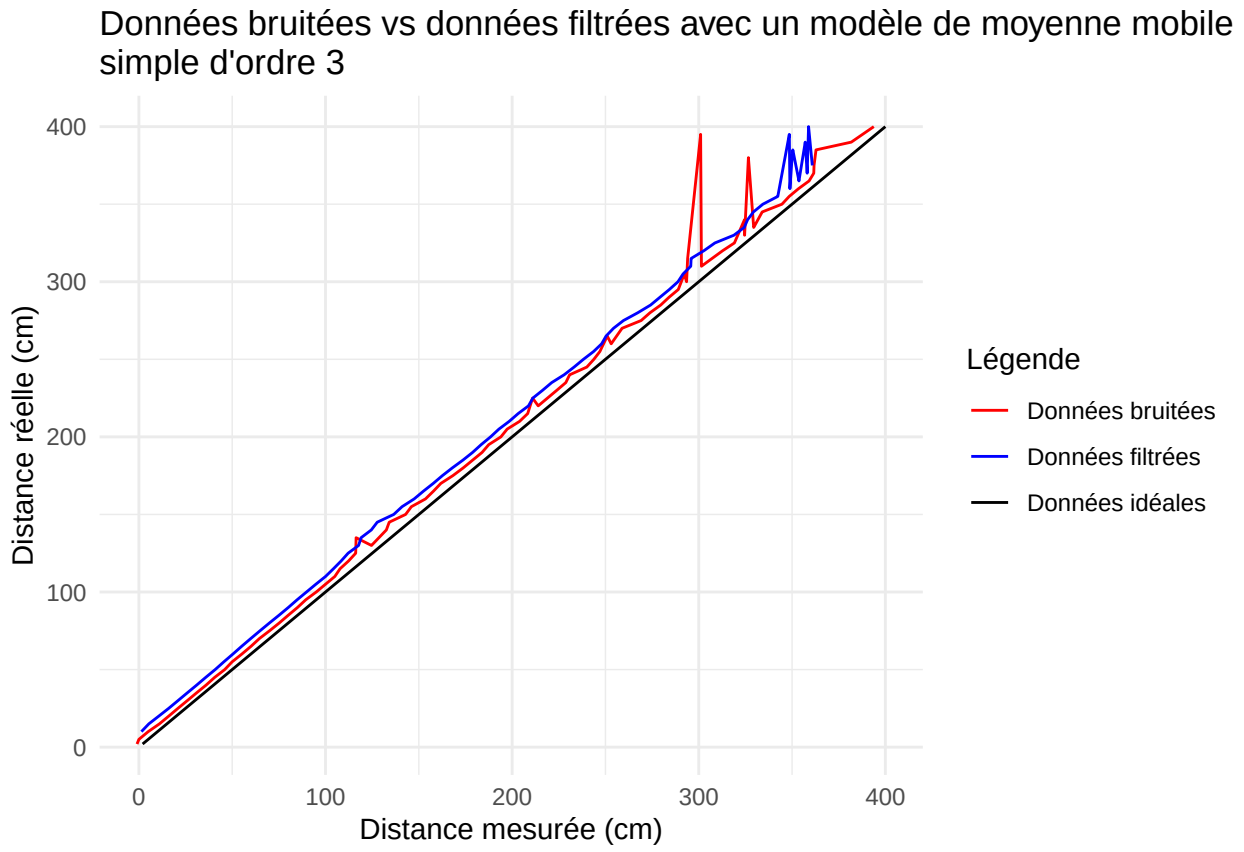
```
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_regression_lineaire = predict(modele_regression_lineaire))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_regression_lineaire, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
```

```
      y = "Distance réelle (cm)",
      color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Création d'un modèle de moyenne mobile simple d'order 3

```
modele_moyenne_mobile_simple_3 <- SMA(donnees_detecteur_obstacle_aggregated$measured_distance_cm, n = 3)

donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_moyenne_mobile_simple_3 = modele_moyenne_mobile_simple_3)

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_moyenne_mobile_simple_3, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de moyenne mobile\nsimple d'ordre 3
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données filtr
  theme_minimal()
```



Appliquons maintenant un modèle de régression linéaire sur la moyenne mobile simple d'ordre 3.

```
## Création du modèle de régression linéaire avec la moyenne mobile simple d'ordre 3
modele_moyenne_mobile_simple_3_predict <- lm(real_distance_cm ~ modele_moyenne_mobile_simple_3, data = d
coef(modele_moyenne_mobile_simple_3_predict)
```

```
##                     (Intercept) modele_moyenne_mobile_simple_3
##                       7.601984                        1.031440
```

```
## Prédiction des valeur réelles avec la moyenne mobile simple d'ordre 3
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_moyenne_mobile_simple_3_predict = ifelse(is.na(modele_moyenne_mobile_simple_3), NA, pred
```

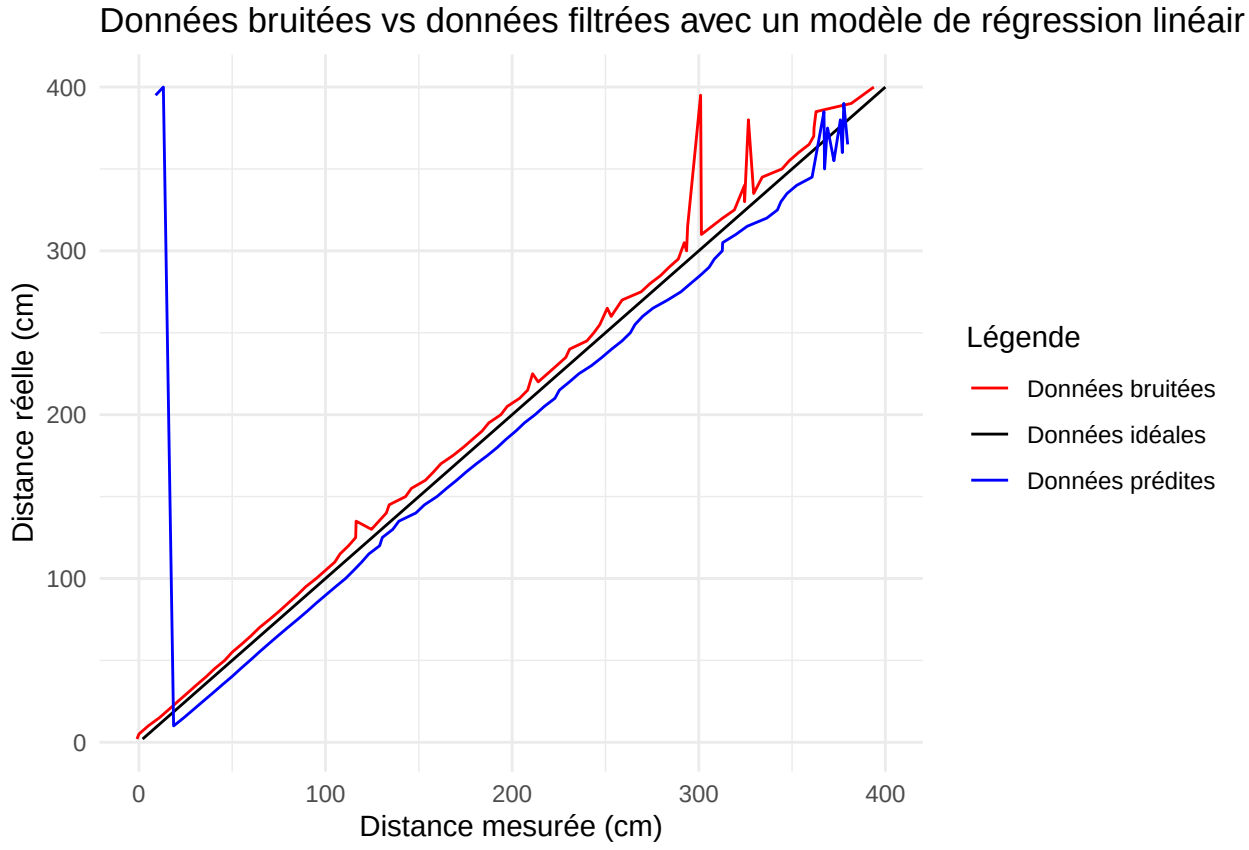
L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 7.601984 + 1.0314399x$$

Voici le modèle de prévision pour la régression linéaire avec la moyenne mobile simple d'ordre 3.

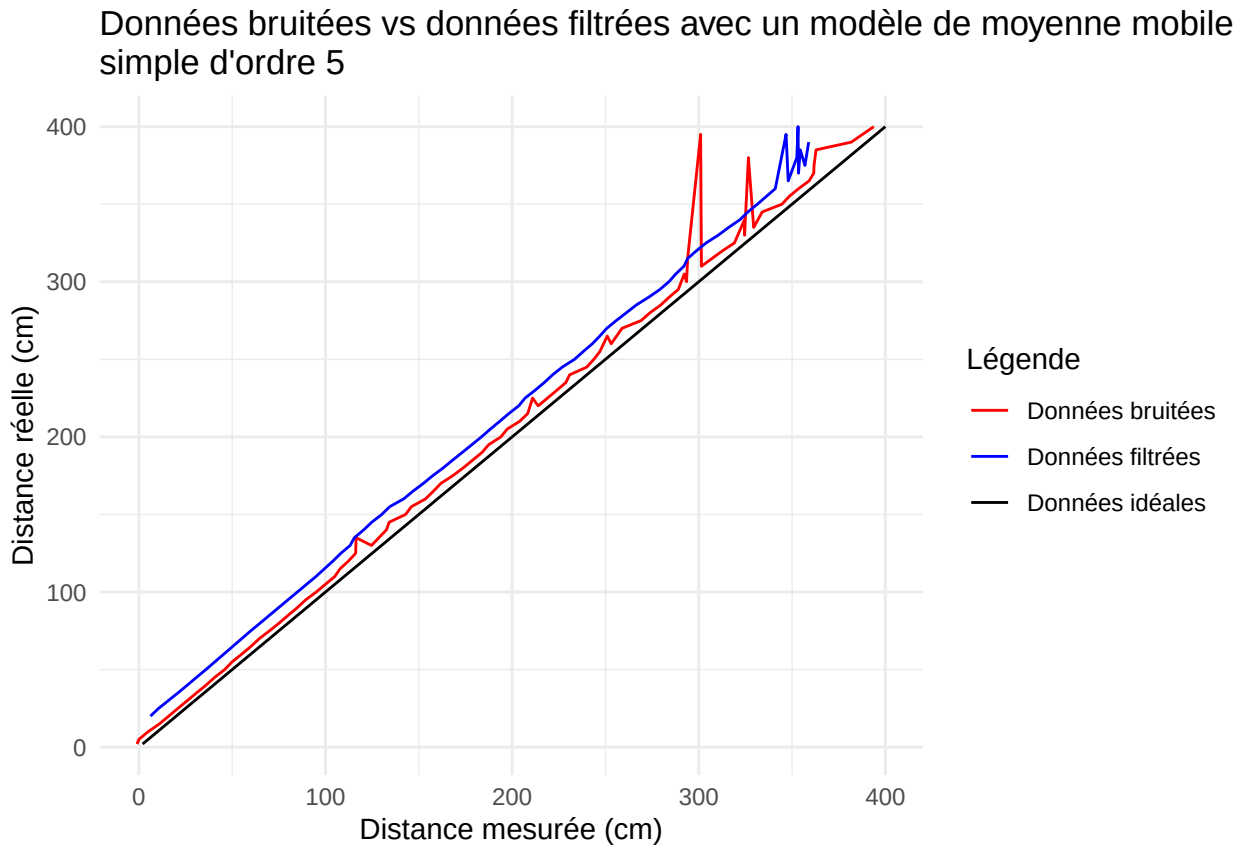
```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_moyenne_mobile_simple_3_predict, y = real_distance_cm, col = "Données prédite
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
```

```
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Création d'un modèle de moyenne mobile simple d'ordre 5

```
modele_moyenne_mobile_simple_5 <- SMA(donnees_detecteur_obstacle_aggregated$measured_distance_cm, n = 5)

donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_moyenne_mobile_simple_5 = modele_moyenne_mobile_simple_5)

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_moyenne_mobile_simple_5, y = real_distance_cm, col = "Données filtrées")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle de moyenne mobile\nsimple d'ordre 5
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données filtr
  theme_minimal()
```



Appliquons maintenant un modèle de régression linéaire sur la moyenne mobile simple d'ordre 5.

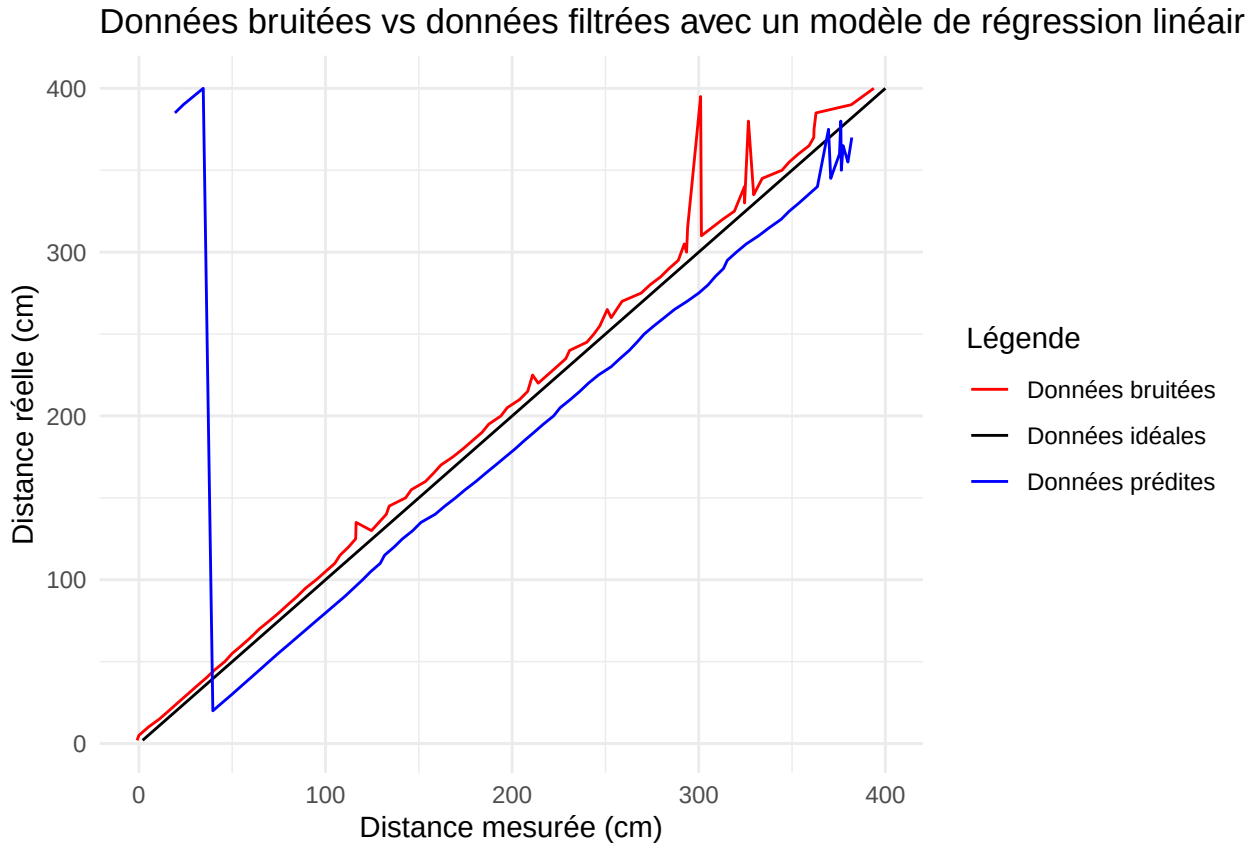
```
## Création du modèle de régression linéaire avec la moyenne mobile simple d'ordre 5
modele_moyenne_mobile_simple_5_predict <- lm(real_distance_cm ~ modele_moyenne_mobile_simple_5, data = d

## Prédiction des valeur réelles avec la moyenne mobile simple d'ordre 5
donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_moyenne_mobile_simple_5_predict = ifelse(is.na(modele_moyenne_mobile_simple_5), NA, pred
```

L'équation du **modèle de régression linéaire** est alors donnée par :

$$y = 12.9215781 + 1.0283281x$$

Voici le modèle de prévision pour la régression linéaire avec la moyenne mobile simple d'ordre 5.

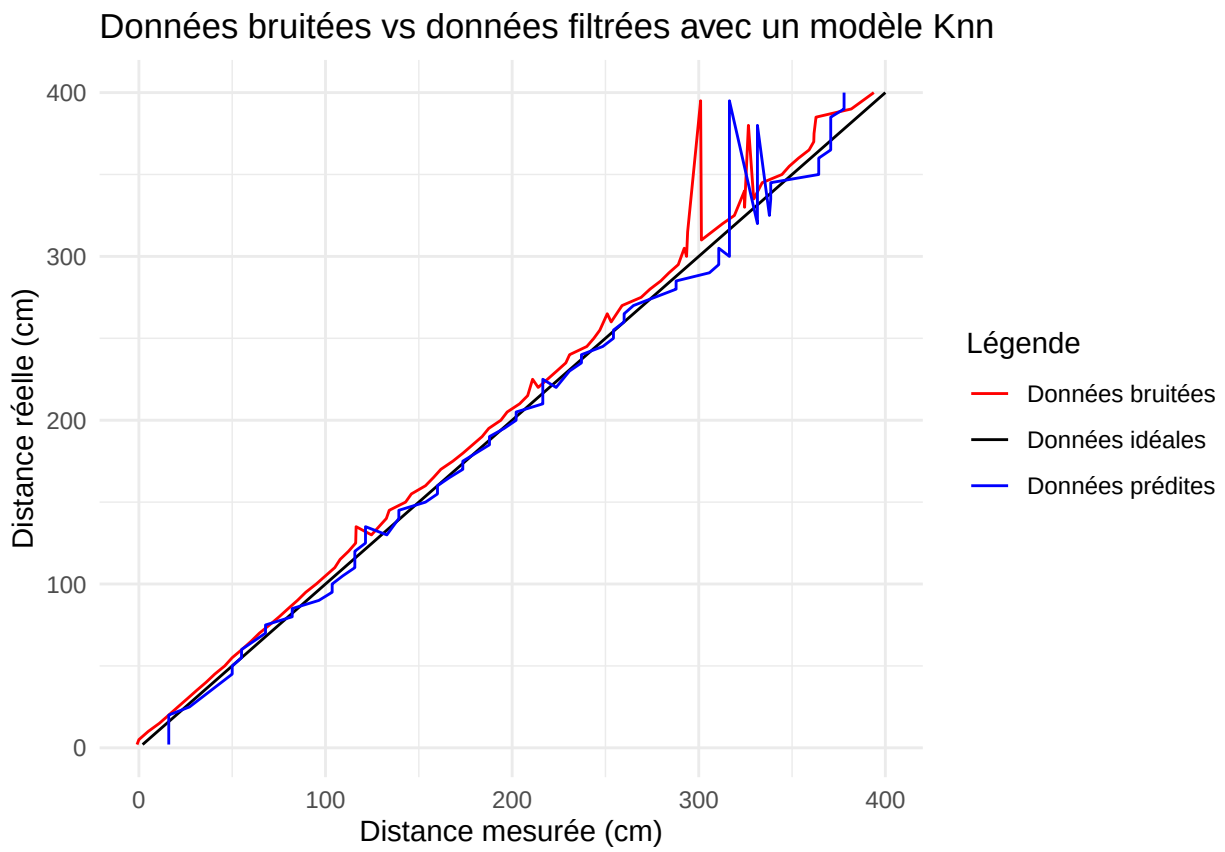
```
## Tracer les données bruitées vs les données prédites par le modèle de régression linéaire
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_moyenne_mobile_simple_5_predict, y = real_distance_cm, col = "Données prédite
  labs(title = "Données bruitées vs données filtrées avec un modèle de régression linéaire",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Création d'un modèle Knn

```
## Pratitionnement des données en train et test
set.seed(123)
train_indices <- createDataPartition(donnees_detecteur_obstacle_aggregated$real_distance_cm, p = 0.8, li

data_knn_test <- donnees_detecteur_obstacle_aggregated[-train_indices,] %>%
  select(real_distance_cm, measured_distance_cm)
data_knn_train <- donnees_detecteur_obstacle_aggregated[train_indices,] %>%
  select(real_distance_cm, measured_distance_cm)

control <- trainControl(method = "cv", number = 10)

file_model_knn = "../data/model_knn.rds"

if (file.exists(file_model_knn)) {
  model_knn <- readRDS(file_model_knn)
} else {
  tune_grid_knn <- expand.grid(k = 1:10)

  ## Entrainement du modèle Knn
  model_knn <- train(real_distance_cm ~ measured_distance_cm, data = data_knn_train, method = "knn", tr
  saveRDS(model_knn, file_model_knn)
}
```

```r
## Prédiction avec le modèle Knn
predictions_knn <- predict(model_knn, data_knn_test)

donnees_detecteur_obstacle_aggregated <- donnees_detecteur_obstacle_aggregated %>%
  mutate(modele_knn = ifelse(is.na(measured_distance_cm), NA, predict(model_knn, .)))

## Tracer les données bruitées vs les données filtrées
ggplot(donnees_detecteur_obstacle_aggregated, aes(x = measured_distance_cm)) +
  geom_line(aes(x = real_distance_cm, y = real_distance_cm, col = "Données idéales")) +
  geom_line(aes(x = measured_distance_cm, y = real_distance_cm, col = "Données bruitées")) +
  geom_line(aes(x = modele_knn, y = real_distance_cm, col = "Données prédites")) +
  labs(title = "Données bruitées vs données filtrées avec un modèle Knn",
       x = "Distance mesurée (cm)",
       y = "Distance réelle (cm)",
       color = "Légende") +
  scale_color_manual(values = c("Données idéales" = "black", "Données bruitées" = "red", "Données prédi
  theme_minimal()
```



Comparaison des modèles

```r
## Erreur sur le filtre Butterworth
RMSE_butter <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacle_
MAE_butter <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacle_ag

## Erreur sur le filtre Chebyshev de type 1
```

```r
RMSE_chebyshev1 <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obsta
MAE_chebyshev1 <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacl

## Erreur sur le filtre Chebyshev de type 2
RMSE_chebyshev2 <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obsta
MAE_chebyshev2 <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacl

## Erreur sur le filtre elliptique
RMSE_elliptic <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacl
MAE_elliptic <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacle_a

## Erreur sur le modèle de régression linéaire
RMSE_regression_lineaire <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecte
MAE_regression_lineaire <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteu

## Erreur sur le modèle de moyenne mobile simple d'ordre 3
valid <- moyenne_mobile_simple_3 <- !is.na(donnees_detecteur_obstacle_aggregated$modele_moyenne_mobile_s
RMSE_moyenne_mobile_simple_3 <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm[valid], donn
MAE_moyenne_mobile_simple_3 <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm[valid], donnee

## Erreur sur le modèle de moyenne mobile simple d'ordre 5
valid <- moyenne_mobile_simple_5 <- !is.na(donnees_detecteur_obstacle_aggregated$modele_moyenne_mobile_s
RMSE_moyenne_mobile_simple_5 <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm[valid], donn
MAE_moyenne_mobile_simple_5 <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm[valid], donnee

## Error on the Knn model
RMSE_knn <- rmse(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacle_aggr
MAE_knn <- mae(donnees_detecteur_obstacle_aggregated$real_distance_cm, donnees_detecteur_obstacle_aggreg

## Création d'un data frame pour les erreurs
erreurs <- data.frame(modele = c("Butterworth", "Chebyshev de type 1", "Chebyshev de type 2", "Elliptiqu
                      RMSE = c(RMSE_butter, RMSE_chebyshev1, RMSE_chebyshev2, RMSE_elliptic, RMSE_regres
                      MAE = c(MAE_butter, MAE_chebyshev1, MAE_chebyshev2, MAE_elliptic, MAE_regression_l

print(erreurs)
```

```
##                            modele      RMSE       MAE
## 1                     Butterworth 14.350809 11.008969
## 2             Chebyshev de type 1 25.162007 21.065897
## 3             Chebyshev de type 2 17.186707 16.019051
## 4                      Elliptique 25.148349 21.054370
## 5             Régression linéaire  9.849407  8.897531
## 6 Moyenne mobile simple d'ordre 3 15.153396 13.618987
## 7 Moyenne mobile simple d'ordre 5 19.208796 18.350649
## 8                             Knn 12.351971  6.661376
```