Modélisation de l'information contenue dans les défenses des narvals dans le but d'estimer leur durée de vie

Yanis BEN BELGACEM, Vadim BERTRAND, Angélique SAILLET

Sommaire

Comme nous avons pu le voir précédemment, le narval est une espèce de cétacés vivant dans l'océan Arctique. Ces animaux d'une durée de vie moyenne de 50 ans, possèdent deux dents. Chez les femelles, les dents restent à l'intérieur de la boîte crânienne, tandis que pour les mâles, la canine gauche s'allonge et prend la forme d'une corne, comme le montre la Figure 1. Elle commence à pousser au travers de la lèvre supérieure gauche dès l'âge d'un an lors de la puberté et croît jusqu'à la maturité sexuelle, entre 8 et 9 ans. Cette défense torsadée possède des fonctionnalités et propriétés uniques dans la nature. Elle contient des millions de terminaisons nerveuses, ce qui en fait un organe de sensoriel très développé [1].

Figure 1: Vue de face d'un narval et de sa dent. [2]

Certains chercheurs danois, comme Eva Garde, s'intéressent plus particulièrement à l'étude de cette défense. Leur objectif est d'estimer la durée de vie du narval à travers l'information contenue dans cette dent. Pour mener cette étude, plusieurs découpes latérales des dents d'animaux décédés ont été réalisées. Comme nous pouvons le voir sur la Figure 2, ces découpes se présentent sous la forme d'une séquence de sillons ou de couches et détiennent plusieurs types d'informations. En effet, nous remarquons la présence de marqueurs saisonniers sur les sillons au cours de la croissance des dents. Ces derniers créent des motifs sinusoïdaux. La fréquence et la forme de ces sinusoïdes varient d'une année à l'autre selon la variabilité de la durée ou de l'intensité des saisons [3]. L'information portée par les motifs à l'intérieur des défenses est donc logiquement liée à la durée de vie de l'animal.

Figure 2: Présentation de l'allure d'une section en longueur d'une dent de narval. [3]

Le premier objectif pour cette problématique est le choix d'un modèle sinusoïdal pouvant représenter l'information contenue dans la dent de l'animal. À partir de cette forme de modèle et d'observations, le deuxième objectif sur lequel nous allons nous concentrer est celui de l'estimation des paramètres du modèle

sinusoïdal. Nous présentons donc dans les parties suivantes, le modèle envisagé ici ainsi que notre démarche d'estimation de ces paramètres à partir d'un algorithme SAEM.

1 Modèle sinusoïdal

Comme nous l'avons évoqué précédemment, les motifs sinusoïdaux observés sont le reflet de la variabilité des saisons, ainsi ce motif n'est pas répété identiquement en fonction du temps. Ces variations complexifient donc la modélisation de cette information.

Les observations le long de la défense sont notées Y_i pour $i = 1, \dots, n$, avec la position correspondante sur la dent notée x_i. Le modèle est le suivant :

$$Y_i = f(x_i, \varphi) + \varepsilon_i$$

avec ε_i un bruit aléatoire suivant une loi normale de moyenne 0 et de variance ω^2.

La fonction de régression $f(x, \varphi)$ est une fonction périodique sinusoïdale telle que :

$$f(x, \varphi) = A \sin(g(x) + b) + B \sin(2g(x) + 2b + \frac{\pi}{2})$$

avec

$$g(x) = ax + \xi_x$$

et finalement ξ_x, un processus aléatoire d'Ornstein-Uhlenbeck, tel que :

$$d\xi_x = -\beta \xi_x dx + \sigma dW_x$$

Dans ce cadre, l'objectif est donc d'estimer les paramètres θ :

- $\varphi = (A, B, a, b)$,
- $\psi = e^{-\beta\Delta}$, où Δ est l'intervalle de temps entre deux observations,
- et $\gamma^2 = \frac{\sigma^2}{2\beta}(1 - \psi^2)$.

Une réalisation de ce modèle est présentée sur la Figure 3.

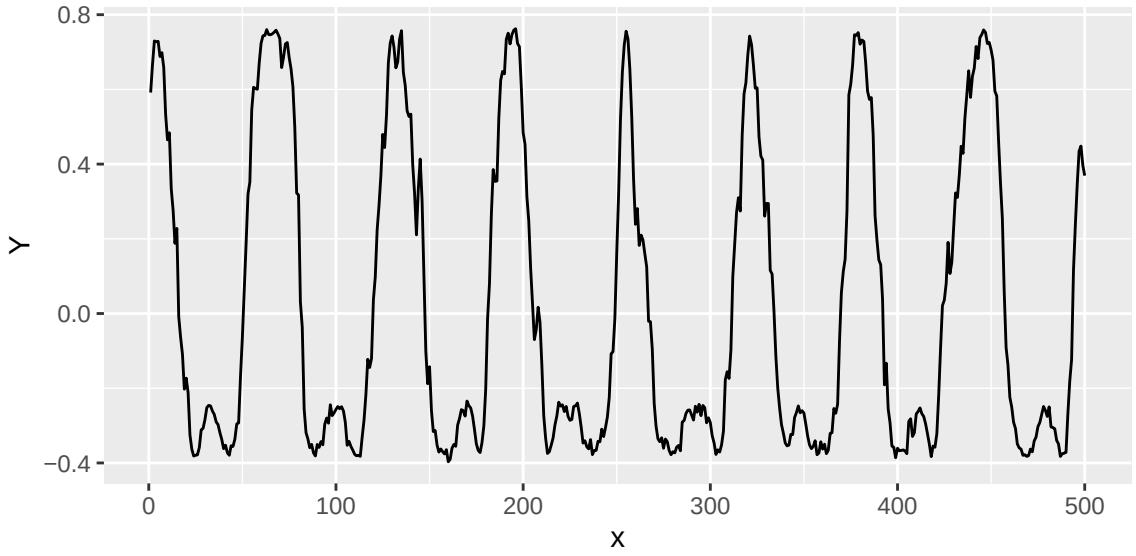

Figure 3: Simulation des observations Y, avec les paramètres suivants : $A = 0.5$, $B = -0.25$, $b = 1$, $a = 0.1$, $\beta = 0.05$, $\sigma = 0.1$, $\omega = 0.01$ et $\delta = 1$.

2 Estimation des paramètres à partir d'un algorithme SAEM

Afin d'estimer les paramètres θ du modèle présenté dans la partie précédente, nous avons implémenté une procedure reposant sur l'algorithme SAEM. Nous allons d'abord présenter le principe d'un algorithme EM, puis celui de son approximation stochastique : l'algorithme SAEM. Durant sa présentation, nous détaillerons l'algorithme MCMC et le filtre particulaire avant de présenter l'algorithme complet. Nous présenterons ensuite, dans une dernière partie, la simulation ainsi que les résultats obtenus par l'algorithme MCMC, par le filtre particulaire, plus largement par l'algorithme SAEM et finalement par le plan d'expérience mis en place.

2.1 Algorithme EM

L'algorithme EM est basé sur la log-vraisemblance complète du modèle. La solution du processus sous-jacent ξ_x s'écrit

$$\xi_{x+\Delta} = \xi_x \psi + \int_x^{x+\Delta} \sigma e^{\beta(s-x)} dW_s$$

de sorte que la densité de transition soit définie telle que

$$p(\xi_{x+\delta}|\xi_x) = \mathcal{N}(\xi_x \psi, \frac{\sigma^2}{2\beta}(1-\psi^2))$$

Ainsi, la log-vraisemblance complète s'écrit de la manière suivante :

$$\begin{aligned}
\log L(Y,\xi,\theta) &= \sum_{i=1}^{n} \log p(Y_i|\xi_i) + \sum_{i=1}^{n} \log p(\xi_i|\xi_{i-1}) + \log p(\xi_1) \\
&= -\sum_{i=1}^{n} \frac{(Y_i - f(x_i,\varphi))^2}{2\omega^2} - \frac{n}{2}\log(\omega^2) \\
&\quad -\sum_{i=1}^{n} \frac{(\xi_i - \xi_{i-1}\psi)^2}{\frac{\sigma^2}{\beta}(1-\psi^2)} - \frac{n}{2}\log(\frac{\sigma^2}{2\beta}(1-\psi^2)) \\
&= -\sum_{i=1}^{n} \frac{(Y_i - f(x_i,\varphi))^2}{2\omega^2} - \frac{n}{2}\log(\omega^2) \\
&\quad -\sum_{i=1}^{n} \frac{(\xi_i - \xi_{i-1}\psi)^2}{2\gamma^2} - \frac{n}{2}\log(\gamma^2)
\end{aligned}$$

Pour chaque itération k, l'algorithme EM procède aux deux étapes suivantes, étant donné la valeur courante des paramètres θ^k.

- étape E : calcul de $Q(\theta, \theta^k)$, [Expliquer ce que c'est ?]
- étape M : actualisation des paramètres $\theta^{k+1} = \arg\max_\theta Q(\theta, \theta^k)$.

Pour actualiser les paramètres, nous avons besoin des statistiques exhaustives. Ces statistiques sont obtenues à partir du théorème de factorisation [?] et contiennent toute l'information de la vraisemblance. Leurs définitions sont les suivantes :

$$
\begin{aligned}
S_1(\xi_i) &= \frac{1}{n}\sum_{i=1}^{n}(Y_i - f(x_i(\xi_i),\varphi))^2 \\
S_2(\xi_i) &= \sum_{i=1}^{n}\xi_{i-1}\xi_i \\
S_3(\xi_i) &= \sum_{i=1}^{n}\xi_{i-1}^2 \\
S_4(\xi_i) &= \sum_{i=1}^{n}\xi_i^2
\end{aligned}
$$

L'actualisation des paramètres dépend directement de ces statistiques.

2.2 Algorithme SAEM

Dans notre cas, la distribution conditionnelle $p(\xi|Y;\theta^k)$ n'est pas explicite en raison de la non-linéarité de notre fonction de régression $f(x,\varphi)$. Nous allons donc utiliser un algorithme MCMC pour simuler selon cette distribution. Cela conduit à une version stochastique de l'algorithme EM, à savoir l'algorithme SAEM. Cet algorithme utilise les étapes de l'algorithme EM auxquelles s'ajoute une étape d'apprentissage probabiliste.

Effectivement, pour chaque itération k, les étapes sont les suivantes :

- étape E : simulation d'une nouvelle trajectoire de ξ^k à l'aide d'un algorithme MCMC considérant $p(\xi|Y;\theta^k)$ comme une distribution stationnaire,
- étape SA : approximation stochastique des statistiques exhaustives :

$$
\begin{aligned}
s_1^k &= s_1^{k-1} + \alpha_k(S_1(\xi^k) - s_1^{k-1}) \\
s_2^k &= s_2^{k-1} + \alpha_k(S_2(\xi^k) - s_2^{k-1}) \\
s_3^k &= s_3^{k-1} + \alpha_k(S_3(\xi^k) - s_3^{k-1}) \\
s_4^k &= s_4^{k-1} + \alpha_k(S_4(\xi^k) - s_4^{k-1})
\end{aligned}
$$

- étape M : actualisation de θ^k, à partir des formules suivantes qui utilisent les statistiques exhaustives s^k :

$$
\begin{aligned}
\widehat{\varphi}^k &= \arg\min_{\varphi}\sum_{i=1}^{n}\left(y_i - f(x_i(\xi_i^k),\varphi)\right)^2 \\
\widehat{\psi}^k &= \frac{s_2^k}{s_3^k} \\
\widehat{\omega^2}^k &= s_1^k \\
\widehat{\gamma^2}^k &= \frac{1}{n}(\widehat{\psi^2}^k s_3^k - 2\widehat{\psi}^k s_2^k + s_4^k)
\end{aligned}
$$

2.2.1 Simulation de ξ^k

2.2.1.1 Identifiabilité Afin de justifier l'intérêt de l'utilisation d'un algorithme MCMC ou d'un filtre particulaire, nous nous sommes intéressé à l'identifiabilité du modèle sinusoïdal. Nous pouvons observer sur la Figure 4 une trajectoire du processus ξ_x cible ainsi que trois autres simulations de trajectoire du processus ξ_x pour des valeurs de paramètres fixées, obtenues selon le procédé suivant :

$\xi \leftarrow (0, \xi_2, ..., \xi_n)$ ▷ Initialisation de la première valeur
for $i \in \{1, ..., n\}$ **do**
  $\xi_i = \xi_{i-1} * \psi + \epsilon_\xi$, avec $\epsilon_\xi \sim N(0, \gamma^2)$
end for

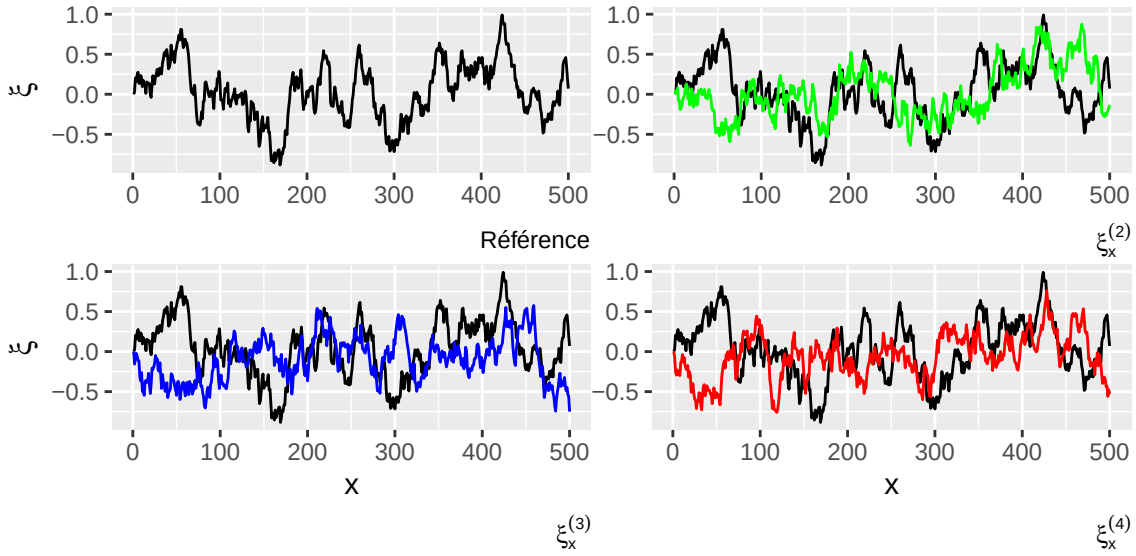

Figure 4: Présentation d'une trajectoire du processus ξ_x cible selon certains paramètres (en noir) et de trois autres trajectoires de ce processus simulées à partir des mêmes paramètres (en vert, bleu et rouge).

Nous observons sur la Figure 4 que plusieurs réalisations du processus ξ conduisent à des trajectoires sensiblement différentes. Ce résultat a un impact direct sur les observations Y puisque comme nous pouvons le voir sur la Figure 5, les observations Y correspondant à des réalisations différentes ne collent pas parfaitement à la distribution cible. Effectivement, la variabilité de simulation du paramètre ξ entraine une variabilité sur les observations Y qui se traduire par un décalage avec la distribution cible. Les observations de Y sont donc sensibles à la trajectoire du processus ξ_x associé.

Nous ne pouvons donc pas nous contenter de tirer une réalisation de ξ_k dans l'étape de simulation de notre algorithme, c'est pour cette raison que nous avons intégré à l'algorithme SAEM une étape MCMC ou filtre particulaire.

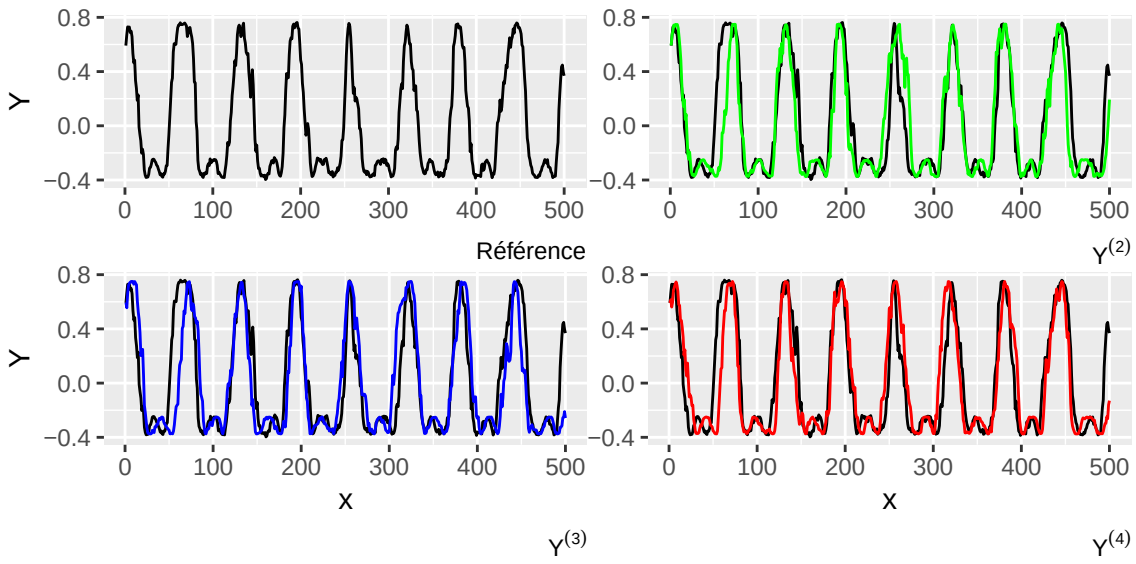

Figure 5: Distributions des Y calculées à partir des trajectoires de ξ_x présentées sur la Figure **??**fig:sim-xis).

2.2.1.2 Algorithme MCMC L'objectif de cet algorithme MCMC (Markov Chain Monte Carlo) est de simuler une trajectoire du processus ξ_x à partir des observations Y_i ainsi que des paramètres γ^2, ψ et ω^2. L'algorithme programmé est plus précisément un algorithme de Gibbs - Metropolis Hasting avec marche aléatoire.

Effectivement, après l'initialisation d'une trajectoire $\xi_0 = (\xi_1, ..., \xi_n)$, l'algorithme procède à M itérations. La trajectoire du processus simulée peut donc s'écrire $\xi_M = (\xi_1, ..., \xi_n)$.

Plus précisément, pour chaque itération k, on calcule pour chaque position x_i, une valeur courante candidate ξ_c avec une marche aléatoire : $\xi_c = \xi_k - 1 + N(0, \delta_i^2)$. Cela introduit un nouveau paramètre $\delta = (\delta_1, ..., \delta_n)$ contrôlant la variance de la marche aléatoire. Pour chacun de ses candidats, une log-probabilité d'acceptation est calculée de la façon suivante :

$$log(\alpha) = min(log(\frac{L(Y, \xi_c)}{L(Y, \xi_{k-1})}), 1)$$

avec :

$$\begin{aligned} \log(\frac{L(Y, \xi_c)}{L(Y, \xi_{k-1})}) &= log(L(Y, \xi_c)) - log(L(Y, \xi_{k-1})) \\ &= -\frac{1}{2\omega^2}\sum_{i=1}^{n}(Y_i - f_\varphi(\xi_c))^2 - \frac{1}{2\frac{\gamma^2}{2}}\sum_{i=1}^{n}(\xi_c - \xi_{k-1}\psi)^2 \\ &\quad + \frac{1}{2\omega^2}\sum_{i=1}^{n}(Y_i - f_\varphi(\xi_{k-1}))^2 + \frac{1}{2\frac{\gamma^2}{2}}\sum_{i=1}^{n}(\xi_{k-1} - \xi_{k-2}\psi)^2 \end{aligned}$$

À partir de la valeur de cette log-probabilité ainsi que d'une réalisation d'une loi uniforme prenant ses valeurs entre 0 et 1, le candidat est soit rejeté, soit accepté auquel cas, il remplace la valeur considérée à l'itération $k-1$.

De plus, nous avons choisi de rendre le paramètre δ adaptatif en fonction du taux d'acceptation acc_rate_i pour chaque point au fil des itérations k. Cela ajoute donc une étape d'actualisation à l'algorithme précédent, ce qui donne finalement l'Algorithme 1 :

Algorithm 1 Algorithme MCMC de simulation d'une trajectoire du processus ξ_x.

$\delta \leftarrow (\delta_1, ..., \delta_n)$ ▷ Initialisation du delta adaptatif
$\delta_{AR} \leftarrow 0.1$ ▷ Pas d'évolution du delta adaptatif
$acc\_rate \leftarrow (acc\_rate_1, ..., acc\_rate_n)$ ▷ Initialisation du vecteur de taux d'acceptation
$acc\_rate_{target} \leftarrow 0.23$ ▷ Taux d'acceptation visé
for $k \in \{1, ..., M\}$ **do**
  for $i \in \{1, ..., n\}$ **do**
    $\xi_c = \xi_k - 1 + N(0, \delta_i^2)$ ▷ Simulation de la valeur courante de ξ
    $log\alpha = min(log(\frac{L(Y,\xi_c)}{L(Y,\xi_{k-1})}), 1)$ ▷ Calcul de la probabilité d'acceptation
    $u \sim U[0, 1]$ ▷ Tirage d'une réalisation de loi uniforme
    if $\log u \leq \log \alpha$ **then**
      $\xi_k = \xi_c$
    else
      $\xi_k = \xi_{k-1}$
    end if
    $acc\_rate_i \leftarrow$ mise à jour du taux d'acceptation
    if $acc\_rate_i < acc\_rate_{target} * (1 - .1)$ **then**
      $\delta_i \leftarrow \delta_i * (1 - delta_{AR})$ ▷ Réduction du delta adaptatif
    else if $acc\_rate_i > acc\_rate_{target} * (1 - .1)$ **then**
      $\delta_i \leftarrow \delta_i * (1 + delta_{AR})$ ▷ Augmentation du delta adaptatif
    end if
  end for
end for

2.2.1.3 Filtre particulaire

2.3 Algorithme SAEM complet

Au principe général d'un algorithme SAEM à Q itérations, nous avons ajouté deux paramètres M_{max} & α_{min} :

- Pour chaque itération q de l'algorithme SAEM, au moins une itération de l'algorithme MCMC est effectuée. Afin de pouvoir améliorer les performances au début de notre algorithme, nous avons décidé de fixer ce nombre d'itérations à 5 pour les M_{max} premières itérations du SAEM, puis ensuite l'algorithme n'accomplis plus qu'une seule itération du MCMC. Le paramètre M_{max} est donc un hyper-paramètre de l'algorithme SAEM à choisir au préalable.
- Le deuxième paramètre concerne l'approximation stochastique effectuée pour chacune des itérations q. Effectivement, durant les premières itérations les approximations sont relativement éloignées de la valeur cible et donc sensiblement différentes entre elles. Au contraire, dans les dernières itérations, étant donné le phénomène de convergence que nous devons observer, les approximations sont censées être plus proches de la valeur cible et également entre elles. Afin de prendre en compte ce phénomène, nous faisons varier la valeur du paramètre de mémoire α permettant de tenir compte des valeurs précédemment estimées au fil des itérations à partir du paramètre α_{min} de la façon suivante :
  - dans un premier temps $\alpha = 1$ pour les α_{min} premières itérations, ce qui permet, d'appliquer la formule complète d'approximation,
  - puis, pour les (Q-α_{min}) dernières itérations, les alphas sont calculés de la façon suivante :

$$\alpha_{\alpha_{min}:Q} = \frac{1}{l^{0.8}}, \text{ avec } l = 1 : (Q - \alpha_{min})$$

Ainsi le paramètre α décroît au fil des itérations à partir de la ${\alpha_{min}}^{\text{ième}}$ itération. Cette procédure permet de réduire l'importance du terme associé à α et d'augmenter donc l'influence de la valeur précédente pour ces itérations-là.

En prenant en compte ces deux paramètres supplémentaires, l'algorithme se présente comme l'Algorithme 2 :

Algorithm 2 Algorithme SAEM complet

$s_1 \leftarrow (s_1^1, ..., s_1^Q)$
$s_2 \leftarrow (s_2^1, ..., s_2^Q)$
$s_3 \leftarrow (s_3^1, ..., s_3^Q)$
$s_4 \leftarrow (s_4^1, ..., s_4^Q)$
$\widehat{\varphi} \leftarrow (\widehat{\varphi}^1, ..., \widehat{\varphi}^Q)$
$\widehat{\psi} \leftarrow (\widehat{\psi}^1, ..., \widehat{\psi}^Q)$
$\widehat{\omega^2} \leftarrow (\widehat{\omega^2}^1, ..., \widehat{\omega^2}^Q)$
$\widehat{\gamma^2} \leftarrow (\widehat{\gamma^2}^1, ..., \widehat{\gamma^2}^Q)$
$\alpha \leftarrow (\alpha_1, ..., \alpha_Q)$ ▷ Initialisation du paramètre d'approximation α
$\alpha_{1:\alpha_{min}} \leftarrow 1$; $\alpha_{\alpha_{min}:Q} \leftarrow \frac{1}{l^{0.8}}$ avec $l = 1 : (Q - \alpha_{min})$
$M \leftarrow (M_1, ..., M_Q)$ ▷ Initialisation du nombre d'itérations du MCMC
$M_{1:M_{max}} \leftarrow 5$; $M_{M_{max}+1:Q} \leftarrow 1$
for $q \in \{1, ..., Q\}$ **do**
▷ Etape E
$\xi^q \leftarrow$ MCMC (cf Algorithme 1) avec $M = M_q$ itérations et θ^{q-1}, pour récupérer une trajectoire de ξ
▷ Etape SA
$S_1 \leftarrow \frac{1}{n}\sum_{i=1}^n (Y_i - f(x_i(\xi^q), \varphi))^2$ ▷ Calcul des statistiques exhaustives
$S_2 \leftarrow \sum_{i=1}^n \xi_{i-1}^q \xi_i^q$
$S_3 \leftarrow \sum_{i=1}^n (\xi_{i-1}^q)^2$
$S_4 \leftarrow \sum_{i=1}^n (\xi_i^q)^2$
▷ Mise à jour des approximations stochastiques
$s_1^q \leftarrow s_1^{q-1} + \alpha_q(S_1(\xi^q) - s_1^{q-1})$
$s_2^q \leftarrow s_2^{q-1} + \alpha_q(S_2(\xi^q) - s_2^{q-1})$
$s_3^q \leftarrow s_3^{q-1} + \alpha_q(S_3(\xi^q) - s_3^{q-1})$
$s_4^q \leftarrow s_4^{q-1} + \alpha_q(S_4(\xi^q) - s_4^{q-1})$
▷ Etape M
$\widehat{\varphi}^q \leftarrow \arg\min_\varphi \sum_{i=1}^n \left(Y_i - f(x_i(\xi_i^q), \varphi)\right)^2$ ▷ Actualisation de θ^q
$\widehat{\psi}^q \leftarrow \frac{s_2^q}{s_3^q}$
$\widehat{\omega^2}^q \leftarrow s_1^q$
$\widehat{\gamma^2}^q \leftarrow \frac{1}{n}(\widehat{\psi^2}^q s_3^q - 2\widehat{\psi}^q s_2^q + s_4^q)$
end for

3 Simulation & résultats

[Phrase d'introduction a la partie]

3.1 MCMC indépendant

Nous avons, dans un premier temps, testé l'efficacité de l'algorithme MCMC programmé indépendamment de l'algorithme SAEM. Pour ce faire, nous avons fixé les valeurs des paramètres : $A = 0.5$, $B = -0.25$, $b = 1$,

$a = 0.1$, $\beta = 0.05$, $\sigma = 0.1$, $\omega = 0.01$ et $\delta = 1$. Ces valeurs ont été utilisées pour simuler une trajectoire de ξ_x cible avec une distribution Y associée selon le modèle sinusoïdal. L'algorithme MCMC a été réalisé avec $M = 150$ itérations et obtient, à partir de ces mêmes valeurs de paramètres et des observations Y, les résultats présentés sur la Figure 6.

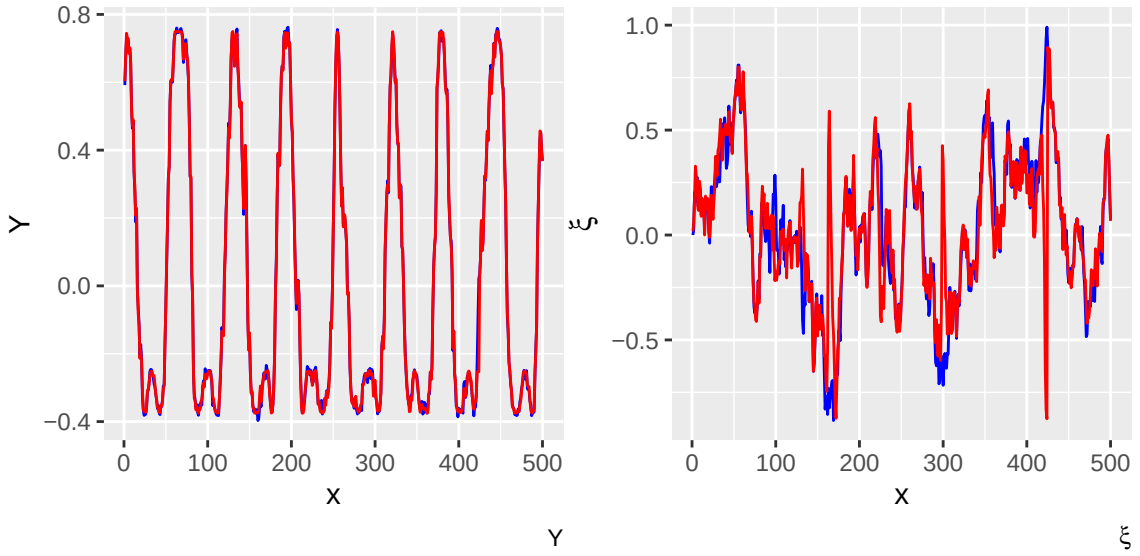

Figure 6: Superposition de la distribution Y cible et de la trajectoire xi réellement utilisée (en bleu) ainsi que de la trajectoire de xi obtenue par l'algorithme MCMC et la distribution Y qui l'utilise (en rouge).

La trajectoire de ξ_x obtenue par le MCMC (en rouge) est très proche de celle à l'origine des observations Y. Nous remarquons cependant que quelques points de ξ sont très mal approchés par l'algorithme. Effectivement pour plusieurs positions x_i, la distance entre la valeur originelle de ξ_i et la valeur estimée par l'algorithme semble grande. Cependant, ces différences n'influencent que très peu l'allure du signal Y, qui reste très satisfaisant. Afin de comprendre la raison de ces erreurs, nous nous sommes intéressés au comportement du δ_i au fil des itérations, relativement à celui du taux d'acceptation acc_rate_i pour un des points concernés. Nous avons comparé leur évolution pour la position x_i où l'erreur est maximale et celle où elle est minimale.

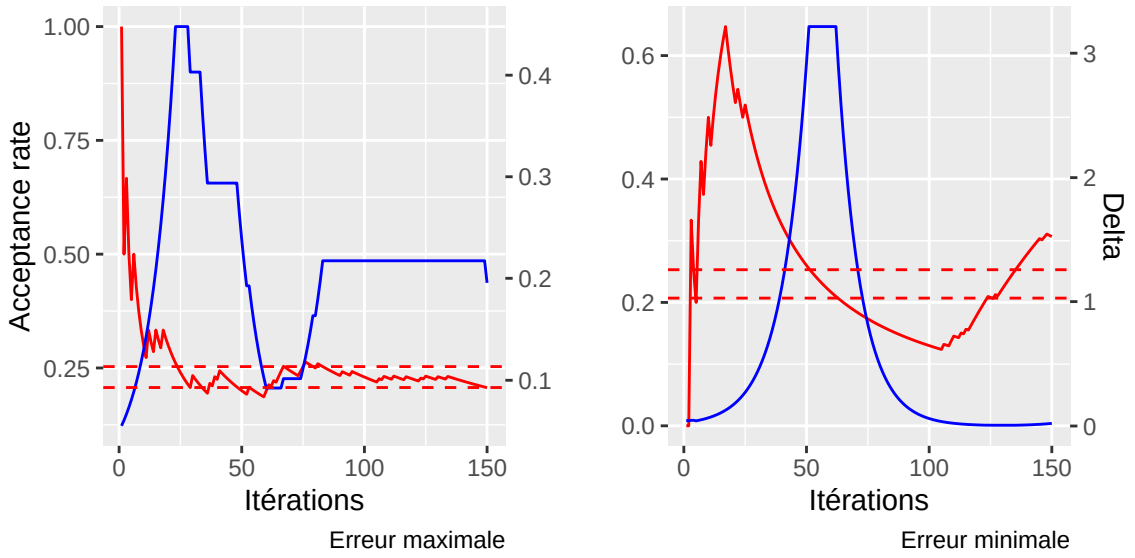

Figure 7: Distributions du taux d'acceptation et du delta adaptatif au fil des itérations pour deux positions différentes : celle pour laquelle l'erreur d'estimation est maximale et celle où elle est minimale.

Nous pouvons voir sur la Figure 7, que dans les deux cas, le delta adaptif évolue correctement. Effectivement le taux d'acceptation est premièrement supérieur à la borne supérieure de notre seuil ce qui entraine l'augmentation de δ. Cette augmentation permet alors au taux de diminuer et donc de rentrer dans nos deux bornes de seuil, et δ stagne durant cette période. Le taux d'acceptation devient ensuite trop bas, et alors la valeur de δ diminue afin de le faire augmenter à nouveau. Nous remarquons seulement que, dans le cas concernant l'erreur maximale, l'évolution du taux d'acceptation est moins rapide que dans le cas de l'erreur minimale, par exemple il atteint l'intervalle de seuil environ vers la $40^{\text{ième}}$ itération contre environ la $20^{\text{ième}}$ pour le point où l'erreur est minimale. Cela entraine une valeur maximale de δ bien plus élevée, d'environ 1.6 contre 0.3. Malgré cette observation, nous n'identifions pas la source de ces points aberrants. Cependant, étant donné que ceux-ci n'impactent pas l'estimation de la distribution des observations Y, l'algorithme MCMC programmé reste selon nous satisfaisant.

3.2 Filtre particulaire

3.3 SAEM

En gardant l'objectif d'estimer optimalement les paramètres utilisés pour obtenir une réalisation des observations Y cibles, nous avons effectué $Q = 500$ itérations de l'algorithme SAEM présenté précédemment. Les paramètres α_{min} et M_{max} ont été fixés après plusieurs tests et analyses graphiques aux valeurs suivantes : $\alpha_{min} = 90$ et $M_{max} = 20$, ce qui correspond à des valeurs retrouvées régulièrement dans la littérature. La trajectoire de ξ_x obtenue après les itérations ainsi que les observations Y l'utilisant sont présentés sur la Figure 8. Nous observons sur cette figure un léger décalage pour les premières positions de la trajectoire de ξ_x estimée. Malgré ce décalage, les deux courbes sont assez proches et nous retrouvons ce résultat pour la distribution de Y.

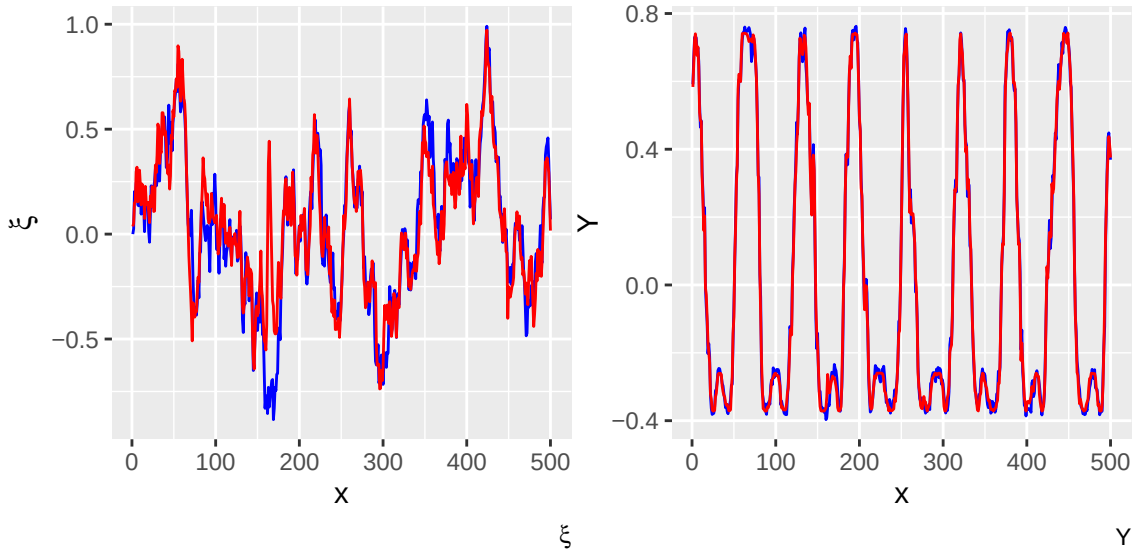

Figure 8: Superposition de la distribution Y cible et de la trajectoire xi réellement utilisée (en bleu) ainsi que de la trajectoire de xi obtenue par l'algorithme SAEM et la distribution Y qui l'utilise (en rouge).

Concernant l'estimation des autres paramètres du modèle, les chaines de Markov obtenues au fil des itérations pour les coefficients ω, ψ et γ sont présentées sur la Figure 9. Nous avons fait le choix d'initialiser ces 3 valeurs à 0.5 pour la première itération. Comme nous pouvons le voir sur cette figure, les chaines des paramètres ω et ψ convergent correctement vers les vraies valeurs des paramètres. Cependant, celle du paramètre γ semble converger plus doucement ce qui entraine une légère surestimation de ce paramètre.

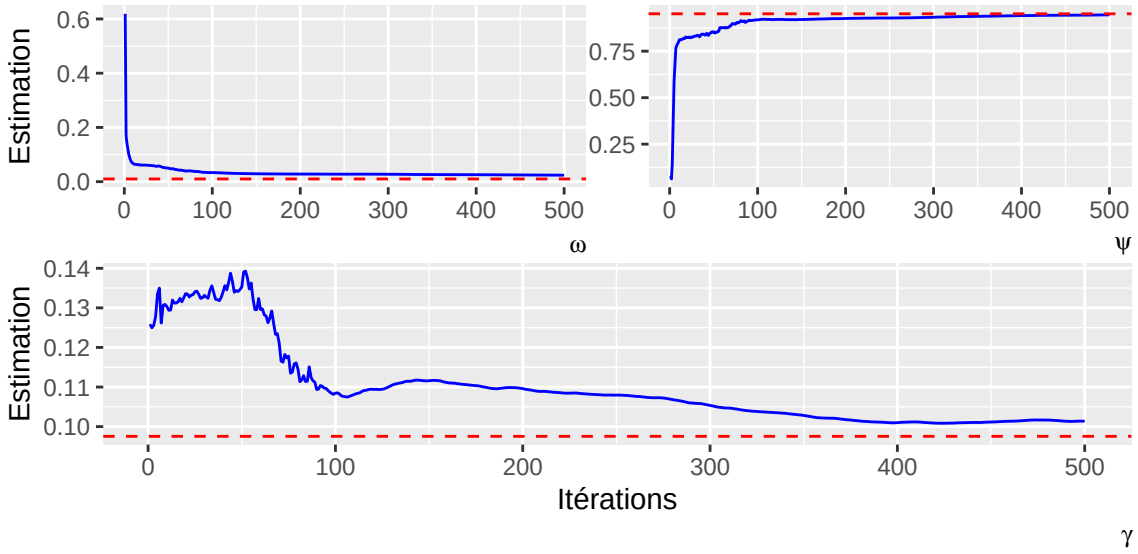

Figure 9: Présentation de l'estimation des coefficients ω, ψ et γ par l'algorithme SAEM, ainsi que des droites représentant la valeur cible de ces paramètres (en rouge).

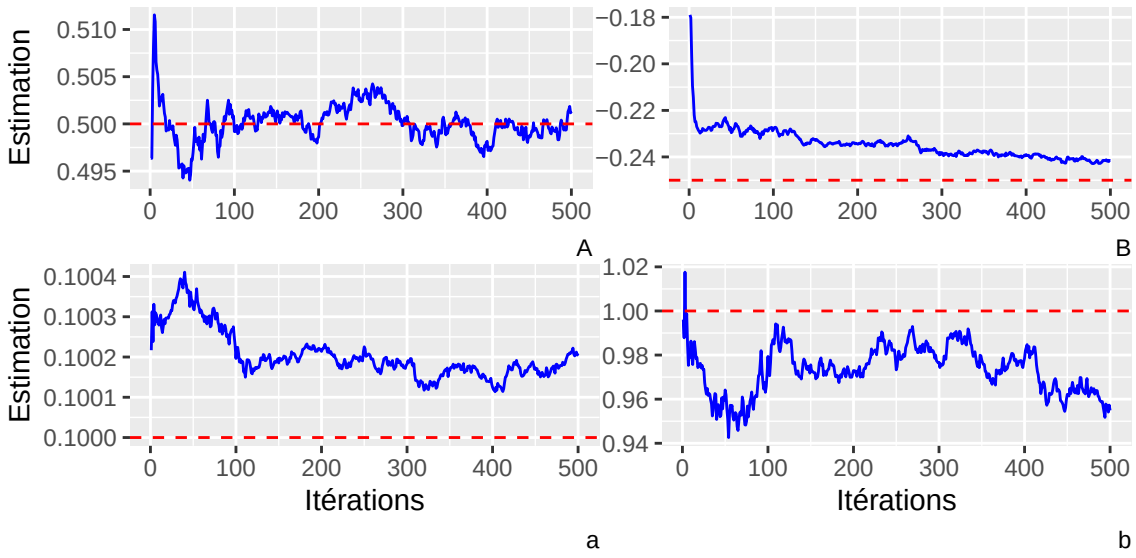

Figure 10: Présentation de l'estimation des coefficients A, B, a et b par l'algorithme SAEM, ainsi que des droites représentant la valeur cible de ces paramètres (en rouge).

Afin de favoriser la convergence de l'algorithme, les paramètres A et B ont été initialisés selon une réalisation de loi uniforme définie entre $-max(|Y|) - \omega$ et $max(|Y|) + \omega$. Cette initialisation prend du sens quant à la définition du modèle sinusoïdal. Dans le même objectif, le paramètre a a été initialisé comme [initialisation de a à préciser] et le paramètre b, en fonction de l'initialisation de a comme [initialisation de b à préciser]. Comme nous pouvons le voir sur la Figure 10, les estimations des coefficients A et B convergent normalement vers la valeur attendue des paramètres. Nous observons, cependant, que bien que très proche de la "vraie" valeur, l'estimation du paramètre a ne converge pas à proprement parler. Pour finir, l'estimation de b semble plus difficile à accomplir : nous ne remarquons pas de convergence vers la valeur attendue et l'estimation finale apparaît éloignée de la valeur fixée initialement.

Les estimations faites par l'algorithme sont donc satisfaisantes pour les paramètres ω, ψ, γ, A, B et a. Il n'arrive cependant pas à estimer convenablement le paramètre b.

3.4 Plan d'expérience

Afin de se rendre compte de la variation de nos estimateurs vis-à-vis de la valeur initiale, nous avons éffectué un plan d'expérience. En effet, notre objectif est d'estimer sur un grand nombre d'itération, nos paramètres, afin d'analyser leurs variations autour de leur valeur fixée. Les paramètres à estimer sont A, B, a, b, ainsi que γ, ψ etω. Nous avons choisi d'effectuer 1000 estimations de chacun de ses paramètres, et calculer la "root-mean-square error" (RMSE) sur chacune d'entre elles, avec pour formule :

$$RMSE = \frac{\sqrt{(\hat{\theta} - \theta)^2}}{\theta}$$

Avec $\hat{\theta}$ notre paramètre éstimé et θ notre paramètre initial fixé.

En vu que l'estimation de chacun de nos paramètres doit se rapprocher et varier autour de notre valeur initiale, nous devrions avoir des estimations $\hat{\theta}$ variant autour de nos $\hat{\theta}$, soit une $RMSE$ variant autour de 0. A travers ces 1000 itérations, nous avons effectué des boxplots de nos valeurs pour la RMSE pour chacun de nos sept paramètres.

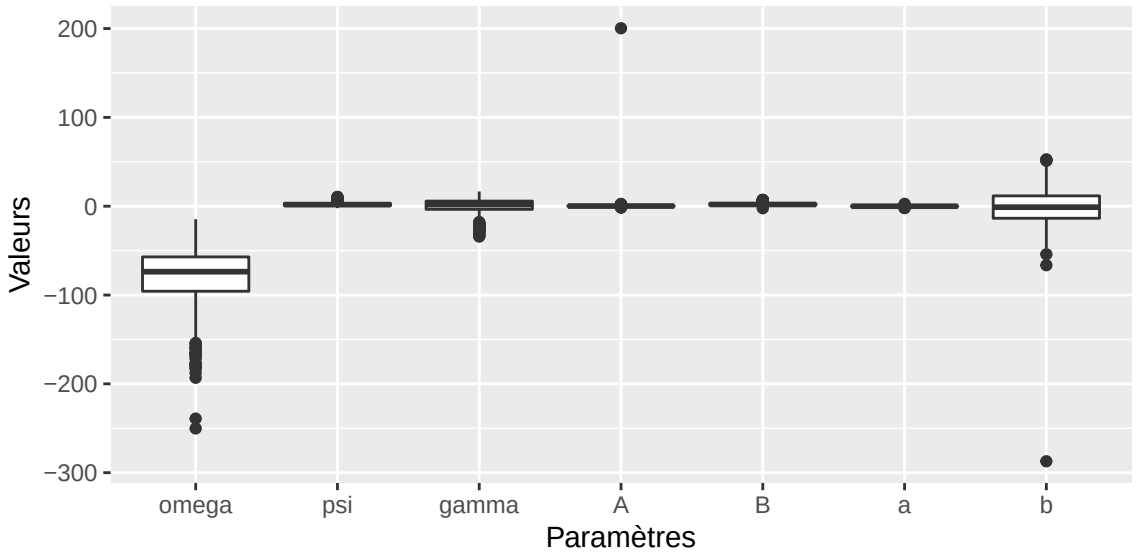

Figure 11: Boxplot des estimations de la RMSE pour chacun de nos paramètres.

Comme attendu, nous avons la RSME proche et variant autour de 0 pour tout nos paramètres hormis ω. En effet, ω semble être sous-estimé et avoir une variation plus grande. Cependant, nous pouvons penser que cette variation est due au fait que la valeur de ω initiale de 0.01 est faible, il est donc plus probable que les variations soient plus importantes en terme de pourcentage.

Pour conclure sur le plan d'expérience, mise à part pour ω,l'estimation de nos paramètres semble varier autour de nos paramètres initiaux fixés. Nous pouvons penser que nos estimations semble être assez proche de la réalité. Cependant, nous devons comprendre comment mieux estimer ω.

Références

[1] Wikipedia contributors. *Narval*. 2022. URL: https : / / fr . wikipedia . org / wiki / Narval (visited on 02/02/2023).

[2] Jean-Pierre Sylvestre. *Dans l'Arctique canadien avec la licorne de mer*. 2018. URL: https://www.peuple-animal.com/article,lecture,1160_dans-l-arctique-canadien-avec-la-licorne-de-mer.html (visited on 02/02/2023).

[3] Peter Bondo. *The narwhal's tusk reveals its past living conditions*. 2021. URL: https://phys.org/news/2021-03-narwhal-tusk-reveals-conditions.html (visited on 02/02/2023).