TP6 : Etude du filtrage numérique

Térence Marchi, Anycia Raulet

1 Introduction :

Le but de ce TP est de réaliser plusieurs filtrages différents d'un même signal pour les comparer et mettre en valeurs leurs avantages et leurs inconvéniants.

2 Filtrage analogique

Le filtrage analogique est une manière de filtrer avec des composants réel. Nous avons réalisé un filtre passe bas du premier ordre, (RC) avec $R = 22\ k\Omega$ et $C = 100\ nF$. Nous avons pris un signal d'entrée en créneau que nous avons mis en entrée d'un circuit RC. Le signal filtré est donc la tension aux bornes de la capacité.

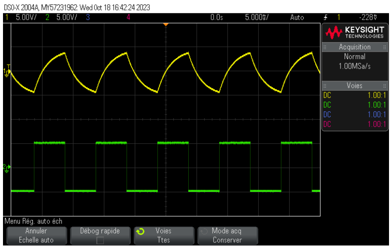


Figure 1: Signal créneau en vert et signal filtré en jaune

On remarque que le filtrage est effectif et qu'il possède la forme habituelle pour un ordre 1.

3 Filtrage numérique avec Latis-Pro

Le filtrage numérique est une fonction réalisé par un calculateur sur un signal préalablement numérisé par une CAN. Le calculateur peut être un ensemble de circuits intégrés (puces), des processeurs programmables, ou un logiciel dans un ordinateur. C'est ce dernier cas que nous envisageons mainteant.
Nous donnons à la carte sysam le signal créneau de notre GBF pour qu'elle pluisse le numériser et l'envoyer sur Latis-Pro. Ainsi, nous pouvons programer le filtrage directement avec Latis-Pro

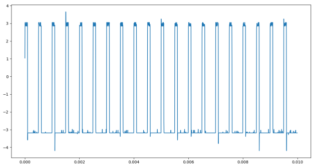
Figure 2: Signal filtré analogiquement en vert et signal filtré avec Latis-Pro en jaune

On remarque que le filtrage est effectif et qu'il est très similaire au filtrage analogique.

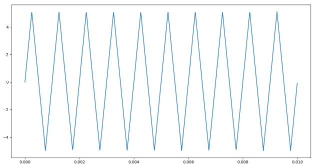
4 Filtrage numérique avec Python

Nous avons ensuite réalisé le filtrage en utilisant Python et la méthode d'Euler explicite. Pour cela nous avons récupérer les donées CSV d'un signal créeau, triangle et une sinusoide sur l'avant de l'oscilloscope. Nous avons importé les données CSV en les séparant en deux listes. Et on a appliqué la méthode d'Euler explicite.

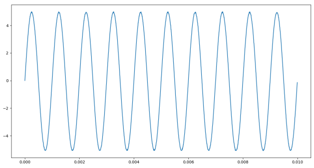
$$S_{n+1} = (g_{e_n} - S_n)\frac{T_e}{\tau} + S_n$$



(a) Signal créneau

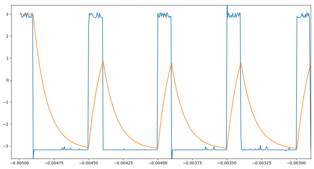


(b) Signal triangulaire

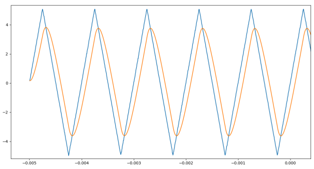


(c) Signal sinusoidal

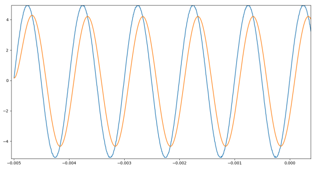
Figure 3: Siganux de fréquence 1kHz non filtré



(a) Signal créneau



(b) Signal triangulaire

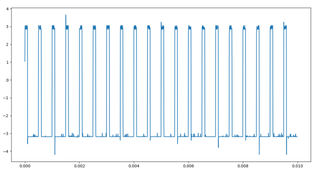


(c) Signal sinusoidal

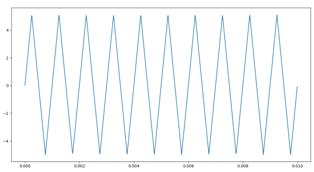
Figure 4: Siganux de fréquence 1kHz filtré

On remarque que le filtrage du créneau donne la courbe d'un ordre un attendue, le filtrage du signal triangulaire lui le rends presque sinusoidal pur avec un déphasage et pour finir le filtrage de la sinusoide ne change pas son allure mais la déphase. Nous avons ensuite réalisé la même expérience mais cette fois en utilisant pour le filtrage la méthode d'Euler implicite.

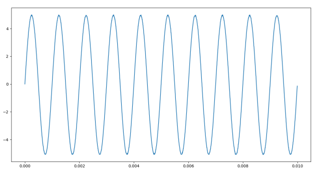
$$S_n = (g_{e_n} - S_{n-1})\frac{T_e}{\tau} + S_{n-1}$$



(a) Signal créneau

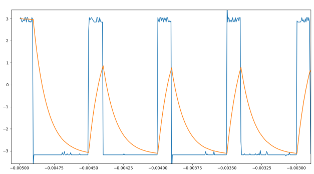


(b) Signal triangulaire

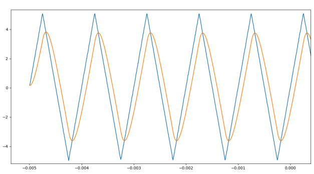


(c) Signal sinusoidal

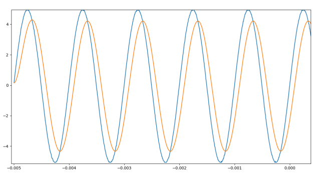
Figure 5: Siganux de fréquence 1kHz non filtré



(a) Signal créneau



(b) Signal triangulaire



(c) Signal sinusoidal

Figure 6: Siganux de fréquence 1kHz filtré

Nous remarquons que les graphiques de filtrages sont identique et que le filtrage avec Euler explicite marche aussi bien que le filtrage avec Euler implicite dans ce contexte précis. En réalité, dans la majorité des cas, la méthode d'Euler implicite est plus stable bien que plus lente.

5 Conclusion :

Finalemant nous remarquons que les différents types de filtrages donnent des résultats similaires, cependant le filtrage numérique est tout de même plus pratique que le filtrage analogique, le simple fait de vouloir changer la fréquence de coupure sur un montage réel est bien plus factidieux que sur un prgramme. De plus l'outil Latis-Pro a tendance à prendre beaucoup de temps à faire les calculs lorsque l'on demande beaucoup de période ce qui est moins visible avec Python.