

Du réel au numérique :

Comprendre et utiliser les filtres, sans prise de tête !

Commençons par quelques pensées à caractère philosophiques :

- *La chose la plus importante en communication est d'entendre ce qui n'est pas dit.* (Peter Drucker)
- *Il semble que la perfection soit atteinte non pas quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à enlever.* (Antoine de Saint-Exupéry)

<i>0 : Préambule</i>	P 2
<i>1 : Un peu d'histoire</i>	P 3
<i>2 Etude des filtres analogiques de base.</i>	P 4
2.1 Qu'est-ce qu'un filtre	P 4
2.1.1 Les filtres de la vie courante	P 4
2.1.2 Les filtres en électricité	P 5
2.2 Etude des filtres de base	P 6
2.2.1 Processus de discrétisation des équations	P 6
2.2.2 Etude des dimensions des facteurs	P 8
2.2.3 Interprétation des coefficients	P 8
<i>3 Etude comportementale comparée des filtres analogiques et numériques</i>	P 8
3.1 Réponse à un échelon	P 8
3.1.1 Etude d'un filtre du 1 ^{er} ordre passe bas (réponse à un échelon)	P 8
3.1.2 Etude d'un filtre du 2 ^{ème} ordre passe bas (réponse à un échelon)	P 9
3.2 Réponse en fréquence	P 11
3.2.1 Réponse en fréquence d'un filtre du premier ordre	P 11
3.2.2 Réponse en fréquence d'un filtre du deuxième ordre	P 13
<i>4 Et les autres filtres ?</i>	P 14
4.1 Les filtres passe Haut	P 14
4.2 Les filtres passe Bande	P 15
4.3 Les filtres coupe Bande	P 15
4.4 Récapitulatif en utilisant nos filtres du §3	P 16
4.5 Création de filtres numériques à partir des équations des fonctions de transfert	P 17
<i>5 Et si nous parlions d'ordres supérieurs à 2</i>	P 20
<i>6 La bibliothèque Arduino « FiltreNum »</i>	P 21
6.1 Principe de base (Problèmes du signal bruité)	P 21
6.2 La solution du filtrage numérique	P 22
6.3 Un exemple commenté	P 23
6.4 Les filtres de la bibliothèque et leurs méthodes associées	P 24
6.4.1 La hiérarchie des classes (objets)	P 24
6.4.2 Les variables globales, procédures et fonctions utiles :	P 26
6.5 Les différents temps d'exécution	P 26
6.6 Un exemple pour jouer avec la bibliothèque 'treat_All_Filters'	P 27
<i>Annexe 1 : Equations aux différences finies.</i>	P 29
<i>Annexe 2 : Théorème de Nyquist-Shanon</i>	P 33
<i>Annexe 3 : Pourquoi effectuer une caractérisation grâce à la fonction échelon</i>	P 35
<i>Annexe 4 : Tracés simplifiés de Bode</i>	P 38

Nota :

Vous trouverez cet article ainsi que tous les fichiers associés sur le site <http://www.altituduino.com>

O : Préambule

Si vous n'avez jamais entendu parler des transformées de Laplace ou des transformées en Z, ou que ces notions vous semblent bien lointaines, ou encore si ce sont des outils mathématiques que vous appliquez sans vraiment les comprendre, alors le reste de cet article est fait pour vous.

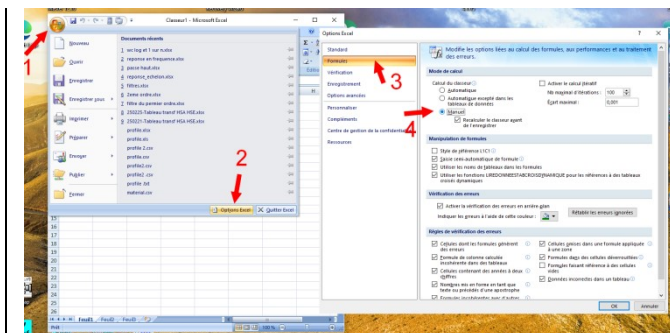
Lorsque l'on commence à s'intéresser aux filtres numériques, on se plonge dans des recherches littéraires pour appréhender les fondamentaux. On se retrouve alors devant une montagne de documents qui semblent affirmer qu'il est impossible que le commun des mortels ne maîtrise pas les sacro-saintes théories mentionnées précédemment. Et, si à la fin de cette lecture, vous restez sur votre faim (oui il s'agit bien de l'homonyme et non une faute !) c'est que j'ai gagné mon pari de vous faire franchir une marche et que vous êtes prêts à ingérer la littérature spécialisée sur le sujet.

Face à la difficulté d'intégrer cette littérature complexe, je me propose de vous offrir ici une rampe à faible pente pour accéder aux bases de la conception des filtres numériques. À cette fin, dans cet article, je vais m'efforcer d'utiliser uniquement le strict minimum de notions mathématiques, reprenant ainsi le problème depuis ses débuts, afin d'amener le lecteur à comprendre le pourquoi et le comment d'un filtre numérique du 1^{er} ordre et du 2nd ordre.

Certains pourront dire qu'il suffit d'avoir Matlab (ou Silab ou Octave...) pour modéliser efficacement ces filtres et comprendre (ou appliquer) les résultats. Cependant, ces logiciels lourds ne se trouvent pas si facilement et nécessitent une certaine habitude pour être manœuvrés. Dans cet article, nous nous appuierons sur un logiciel désormais grand public, Excel (ou tout autre tableur libre de droit). Les courbes et exemples numériques seront générés et analysés à l'aide de cet outil, accessible à un plus large public que Matlab et ses équivalents.

Nota :

1 : Les fichiers Excel joints à cet article contiennent plus de 10^6 cellules à calculer. Sur un PC peu performant, une mise à jour peut prendre plus de 30 secondes à chaque modification de cellule. Il est donc conseillé de désactiver l'option de calcul automatique dans les options d'Excel. Une fois les modifications effectuées, vous pouvez forcer le calcul en utilisant la touche F9 ou l'icône de calcul, si celle-ci a été ajoutée aux préférences du menu.

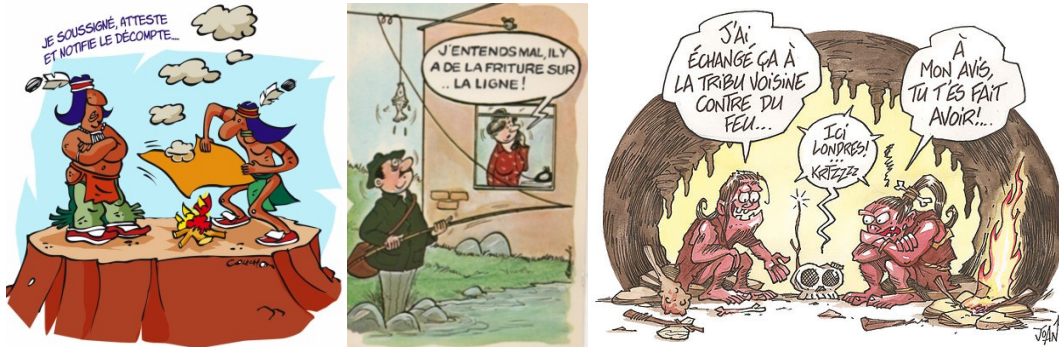


2 : Les fichiers ne sont pas verrouillés. Les seuls cas que vous pouvez modifier sans destructions massives sont celles en **jaune**.

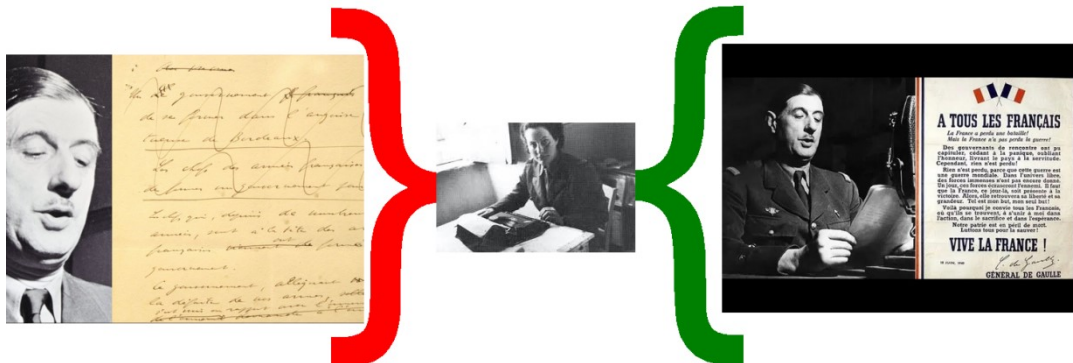
Je pourrais peut-être aller plus loin et à partir des bases exposées ci-après et 'mathématiser' mon discours pour rejoindre les multiples articles sur le sujet. Non ! je ne le ferais pas ! Du moins ici. Je pense que mes pairs ont sur le sujet, ont une plus grande longueur d'avance. Je me cantonnerais donc au soutien de la queue de peloton en les encourageant activement à terminer leur course avec un sentiment de réussite personnel qui leur permettra de prendre un départ à armes égales sur un prochain 'run'.

1 : Un peu d'histoire

Dès que l'humanité a su transmettre des informations sous forme de signaux codés, les notions de richesse et de parasitage du signal se sont introduites, rendant ainsi le message initial souvent indéchiffrable après transmission.



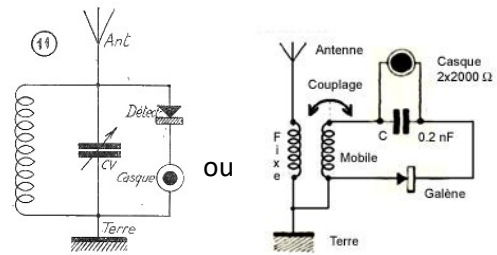
À l'aube des temps modernes, la fée électricité et les ondes sont devenues les vecteurs essentiels de transmission des messages. La détection et le filtrage des informations se sont donc réalisés à l'aide de composants électriques passifs, tels que les résistances, les selfs et les condensateurs. Un agencement judicieux de ces éléments permet de créer des circuits appelés filtres analogiques. Ces circuits isolent la partie intéressante du message tout en tentant d'éliminer les bruits ajoutés lors de la transmission. Ces dispositifs sont ainsi devenus des éléments indispensables de la chaîne de traitement de l'information.



Les équations de ces circuits sont décrites par des équations différentielles (oops), par des équations linéaire (oops), composé de la somme de termes faisant intervenir les dérivées première et seconde (oops), la vitesse de variation et la vitesse de variation de la vitesse de variation du signal.

Suivant le cas, on a alors un filtre du premier ordre (faisant intervenir la vitesse) ou du second ordre (faisant intervenir l'accélération – la vitesse de la vitesse – et éventuellement la vitesse elle-même).

Le poste radio à galène de nos aïeux (Amplitude Modulée) n'est un fait qu'un filtre coupe-bande de détection de la porteuse, suivi d'un filtre passe-bas (à l'origine constitué par le casque lui-même) pour isoler le message audio.



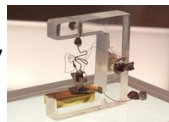
Par la suite, ce message sera amplifié par des lampes (TM1 du général Ferrié 1917



; 1950-1960



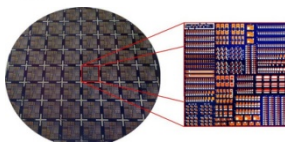
) puis des transistors (1947



; 1960



; 1990



...) mais c'est une autre histoire (quoi que...).

2 Etude des filtres analogiques de base.

2.1 Qu'est-ce qu'un filtre

2.1.1 Les filtres de la vie courante

Ne vous détrompez pas, dans votre vie courante, il n'existe pas deux choses ou actions identiques, le fait d'agir en fonction de cette différence est une opération délibérée de filtrage.



Mais revenons au concept plus restreint du tamis.



Le tamis est un filtre est un outil qui trie les objets en deux catégories (ceux que l'on veut voir en dessous du tamis et les autres) correspondant aux « trous » du tamis :

- Laisser passer une partie des objets à travers (les petits morceaux).
- Retenir l'autre partie en amont du tamis (les gros morceaux).

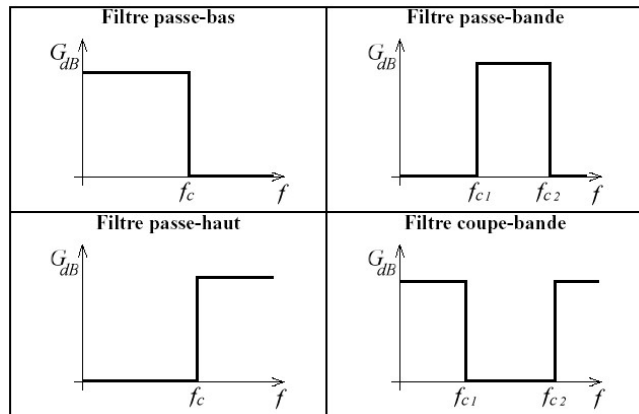
Avec un tamis, soit on récupère ce qui passe à traves du tamis, soit ce qui reste dans la bâté (passe bas et passe haut).

En jouant avec deux tamis de taille de trous différents, on peut isoler une tranche de taille de granulat soit pour la récupérer en produit final (passe bande) soit pour l'enlever du tout venant (coupe bande).

2.1.2 Les filtres en électricité

Tous les mélomanes (et les autres) ont déjà joués sur leur chaîne HI-FI avec les boutons 'Treeble', 'Bass' et pour certaines chaînes 'Medium'. Ils ont ainsi modifié le son émis par les hauts parleurs sans pour autant modifier la source du son. Ce faisant, ils ont mis en jeu une série de filtres électriques sur le spectre de fréquence émis par les larynx de 'Brassens' ou 'Claude François' par exemple.

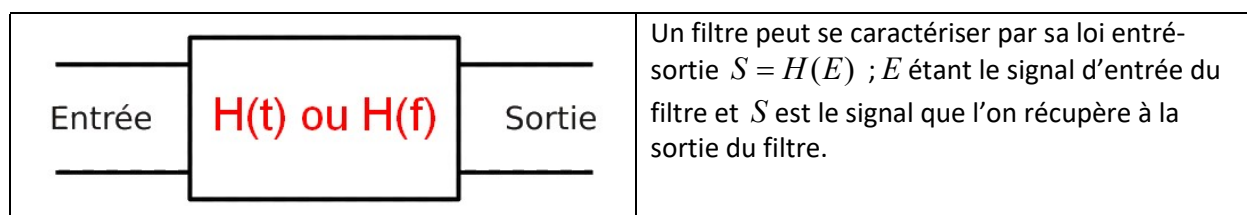
En électricité, les objets tamisés (on dira filtré désormais) sont des oscillations électriques (Ce sont des objets invisibles par nos sens mais dont on peut se faire une représentation sous forme de courbes, ou bien les visualiser à l'aide d'appareils de mesure qui traduisent en visuel ces variations de grandeur électriques que sont les oscilloscopes). On distingue classiquement quatre types de filtres :



- Les filtres passe-bas : on ne laisse passer que les composantes fréquentielles en dessous d'une certaine fréquence appelée fréquence de coupure (à l'instar de la taille de la maille de notre tamis de farine).
- Les filtres passe-haut : on ne laisse passer que les composantes fréquentielles au dessus d'une certaine fréquence (aussi la fréquence de coupure). Ces filtres sont le complémentaire dans le domaine fréquentiel des filtres passe-bas.
- Les filtres passe bande qui cumulent les deux précédents et laisse passer une partie du signal entre deux fréquences distincts.
- Les filtres coupe bande qui est le complémentaire d'un filtre passe bande. Dans cette catégorie, on a le célèbre filtre bouchon qui contrairement à sa dénomination laisse tout passer sauf une bande de fréquence réduite dont la taille de la fenêtre dépend du facteur de « qualité » du filtre. Historiquement on retrouve ce filtre sur les postes radio du début de la TSF.

La réalisation de ces filtres peut se faire à l'aide de composants passifs (qui ne nécessitent pas d'énergie annexe sous forme d'alimentation électrique) tels des résistances (les puristes parlent de résistors), condensateurs (ou capacités pour les non puristes) et selfs (inductances pour les puristes).

L'énergie véhiculée par les fréquences du signal éliminées, est absorbée par les éléments passifs du filtre (résistances, capacités, inductances) qui dissipent cette énergie sous forme de chaleur.



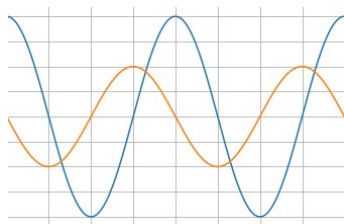
La fonction H s'appelle dans le jargon de la profession la fonction de transfert (Elle transfère en les modifiant les informations venant de l'entrée pour les afficher en sortie.).

Suivant les cas, S et E sont définis comme des fonctions du temps (réponse à un échelon par exemple) ou comme des fonctions composées d'ondes sinusoïdales. Mais ces deux définitions peuvent être unifiées par la théorie de Fourier ([C.F annexe 3](#)) puis par la transformée de Laplace (Stop je vais trop loin ! restons simple).

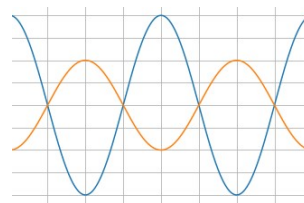
On s'intéressera donc au gain de la fonction H : $G = \frac{\|S\|}{\|E\|}$ que l'on exprimera en fonction de la variable temps ou fréquence suivant le point de vue. Le filtre a aussi un travers qui est de changer la phase (on décale la sortie dans le temps par rapport à l'entrée) du signal. On étudie la variation de cette phase (surtout dans l'étude de régime sinusoïdale) $\Delta\varphi = \text{phase}(S) - \text{phase}(E)$. Très souvent ce décalage est nul pour une fréquence faible puis varie de manière continue, jusqu'à avoir S et E oppositions de phase.

Exemple :

- En bleu le signal d'entrée
- En brun le signal de sortie



La courbe brune est en retard de 90° $\pi / 2$ (ou d'un quart de période) par rapport à la courbe bleue. Le gain de ce filtre est de 0.5 (4 carreaux/8carreaux)



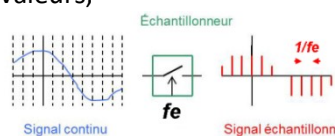
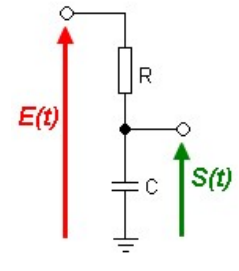
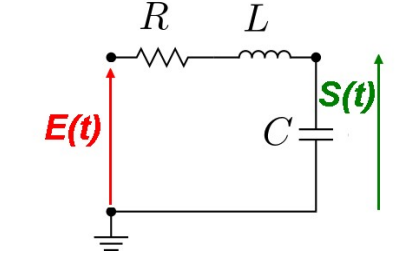
Ici on a toujours un gain de 0.5 mais les signaux sont en opposition de phase (déphasage de π rd, 180° ou d'une demi-période)

Nous allons dans la suite de notre exposé nous intéresser aux filtres passe-bas passifs et leurs équivalents numériques.

2.2 Etude des filtres de base

2.2.1 Processus de discrétisation des équations

Discrétisation des nombres dérivés d'une fonction analogique. C.F. annexe 1	Filtre du premier ordre (RC)	Filtre du second ordre (RLC Série)
---	------------------------------	------------------------------------

<p>Lorsque l'on s'intéresse à un signal représenté par une succession régulière (échantillonnage) de valeurs,</p>  <p>Signal continu Echantillonneur Signal échantillonné</p>		
<p>en se souvenant que la dérivée d'une fonction représente la pente de sa courbe, on montre assez facilement que, en prenant une fréquence d'échantillonnage $f_e = 1/\Delta t$ suffisamment grande, on a l'approximation</p> $\frac{dS}{dt} \approx \frac{S(t) - S(t - \Delta t)}{\Delta t}$ $\approx \frac{S_1 - S_0}{\Delta t}$ $\frac{d^2 S}{dt^2} \approx \frac{S(t + \Delta t) - 2 \cdot S(t) + S(t - \Delta t)}{\Delta t^2}$ $\approx \frac{S_1 - 2S_0 + S_{-1}}{\Delta t^2}$	$E = R \cdot i + \frac{1}{C} \int i \cdot dt$ $S = \frac{1}{C} \int i \cdot dt \Rightarrow i = C \frac{dS}{dt}$ $E = RC \frac{dS}{dt} + S$ <p>Les équations fournissent alors après discrétisation :</p> $S_1 = S_0 \left(1 - \frac{\Delta t}{RC} \right) + \frac{E \cdot \Delta t}{RC}$ <p>En posant $\tau = RC$ la constante de temps classique du circuit RC et</p> $n_0 = \frac{\tau}{\Delta t} \text{ un nombre adimensionnel, nous obtenons}$ $S_1 = \left(S_0 (n_0 - 1) + E \right) \frac{1}{n_0}$ <p>Avec les classiques $\omega_c = \frac{1}{\tau} = \frac{1}{n_0 \Delta t}$</p> <p>et $f_c = \frac{\omega_c}{2\pi} = \frac{1}{2\pi \tau} = \frac{1}{2\pi n_0 \Delta t}$ les pulsations de coupure et fréquences de coupure.</p> <p>On notera aussi la représentation suivante :</p> $S_1 = a_0 E + b_0 S_0$ <p>avec $a_0 = 1/n_0$ et $b_0 = \frac{n_0 - 1}{n_0}$</p>	$E = R \cdot i + L \frac{di}{dt} + \frac{1}{C} \int i \cdot dt$ $S = \frac{1}{C} \int i \cdot dt \Rightarrow i = C \frac{dS}{dt}$ $E = RC \frac{dS}{dt} + LC \frac{d^2 S}{dt^2} + S$ <p>En posant $\tau = RC$ la constante de temps classique du circuit RC et</p> $n_0 = \frac{\tau}{\Delta t} \text{ un nombre adimensionnel,}$ $\omega_0 = \frac{1}{\sqrt{LC}} \text{ et } m_0 = \sqrt{\frac{LC}{\Delta t^2}} = \frac{1}{\omega_0 \Delta t} \text{ un autre nombre adimensionnel nous trouvons après discrétisation}$ $S_1 = \frac{S_0 (n_0 + 2m_0^2 - 1) - S_{-1} m_0^2 + E}{n_0 + m_0^2} \text{ Le}$ <p>facteur de qualité du filtre Q défini par : $Q = \frac{E_{\max i}}{E_{\text{dissipée sur un cycle}}} = \frac{m_0}{n_0}$ qui représente aussi le dépassement ou la valeur de sortie pour $\omega = \omega_0$.</p> <p>On aura aussi $f_0 = \frac{\omega_0}{2\pi} = \frac{1}{2\pi m_0 \Delta t}$</p> <p>On notera aussi la représentation suivante :</p> $S_1 = a_0 E + b_0 S_0 + b_1 S_{-1}$ <p>avec $a_0 = 1/(n_0 + m_0^2)$</p> <p>et $b_0 = \frac{n_0 + 2m_0^2 - 1}{n_0 + m_0^2}; b_1 = \frac{m_0^2}{n_0 + m_0^2}$</p>

On prendra compte de l'artifice de représentation des équations de calcul du pas suivant, en fonction des valeurs actuelles et celles des pas précédents, sous forme de coefficients adimensionnels (qui n'ont pas d'unité spécifiques comme temps, volts...). Cette « ruse » nous permet de générer des prototypes de filtres numériques s'adaptant à la majorité des cas (on parlera dans la littérature mathématique et associée de forme canonique).

2.2.2 Etude des dimensions des facteurs

- $n_0 = \frac{\tau}{\Delta t} \equiv \frac{[T]}{[T]} = 1$ donc adimensionnel.
- $m_0 = \frac{1}{\omega_0 \Delta t} = \sqrt{\frac{LC}{\Delta t^2}} \equiv \sqrt{\frac{[U][T^2]}{[Q]} \cdot \frac{[Q]}{[U]}} \cdot \frac{1}{[T^2]} = 1$ donc adimensionnel.
- De même a_0, b_0, b_1 sont construits à partir de termes adimensionnels et sont adimensionnels.

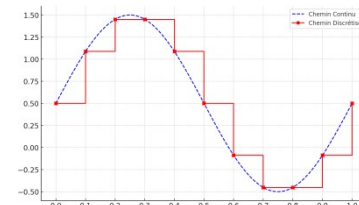
2.2.3 Interprétation des coefficients

- On notera que l'équation d'un filtre du deuxième ordre avec une valeur de self tendant vers 0 revient à un filtre du premier ordre.
- $n_0 = \frac{\tau}{\Delta t}$ représente le nombre de points d'une moyenne glissante équivalente.
- $m_0 = \frac{1}{\omega_0 \Delta t} = \frac{1}{2\pi \cdot \text{freq}_0 \cdot \Delta t} = \frac{T_0}{2\pi \cdot \Delta t}$ représente à 2π près, le nombre de points d'échantillonnage pour une période correspondant à la fréquence propre du circuit.

3 Etude comportementale comparée des filtres analogiques et numériques

On lit dans la littérature qu'un filtre se caractérise par sa réponse à un échelon (une augmentation brusque puis maintien de la valeur en entrée) ([C.F Annexe 3](#)). On a vu dans le paragraphe précédent que l'alter ego numérique du filtre analogique n'intégrait pas de coefficients (pour le filtre du premier ordre) incluant des fréquences. Et pourtant les filtres permettent de couper des plages de fréquences (les mélomanes et constructeurs d'enceintes acoustiques ne me contrediront pas !).

En fait, en discrétisant la fonction d'entrée, nous considérons que durant l'intervalle de temps entre deux prises de valeurs, la valeur est constante et, elle représente ce que la littérature savante appelle un échelon (la valeur bouge brusquement puis reste constante). La grandeur d'entrée est donc transformée en une succession d'échelons. La connaissance de la réponse du filtre à ce type de signal permet donc de caractériser sa réponse à n'importe quel signal.



Mais à quelle fréquence est-il nécessaire de discrétiser la grandeur d'entrée ?

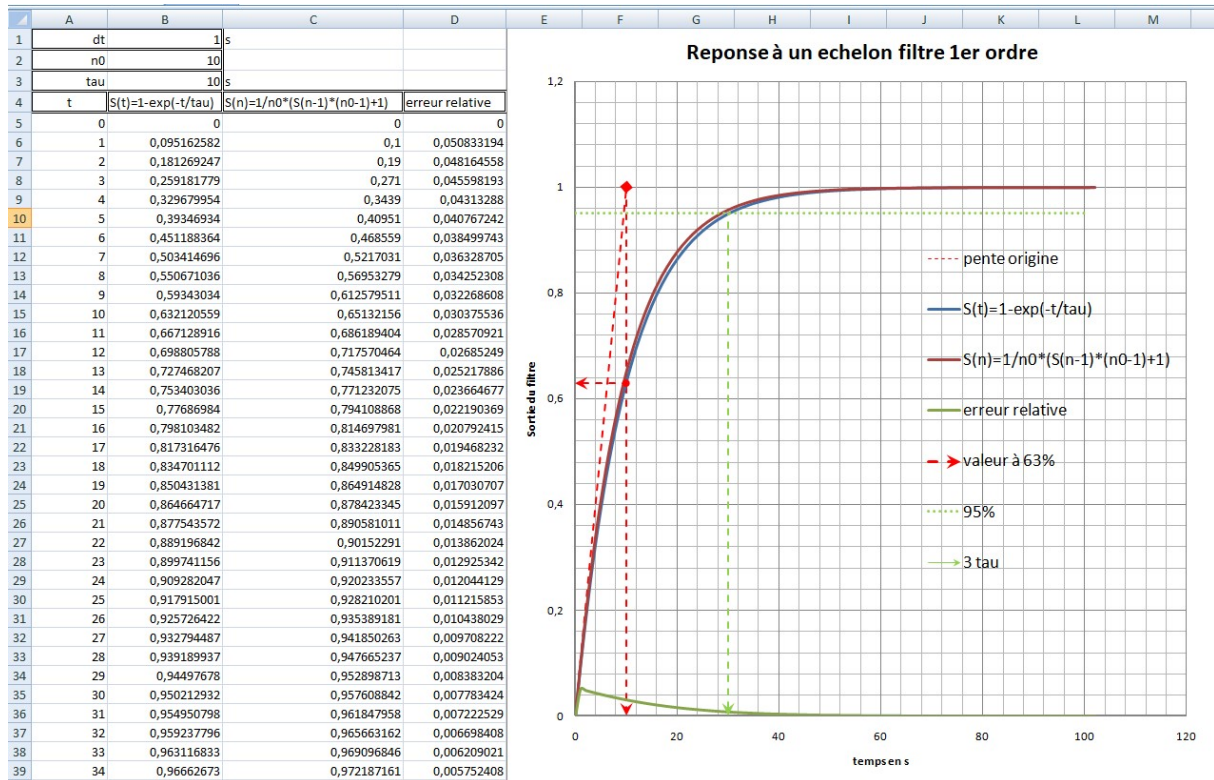
Si nous voulons représenter un signal variable dont la fréquence est de f_{\max} , il nous faut au moins deux points discrétisés par morceau complet de sinusoïde pour reconstituer le signal et donc la fréquence d'échantillonnage ne peut être inférieure à $2 \cdot f_{\max}$ (C.F. annexe 2).

3.1 Réponse à un échelon

3.1.1 Etude d'un filtre du 1^{er} ordre passe bas (réponse à un échelon)

L'équation aux dérivées se résout en $S(t) = 1 \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$

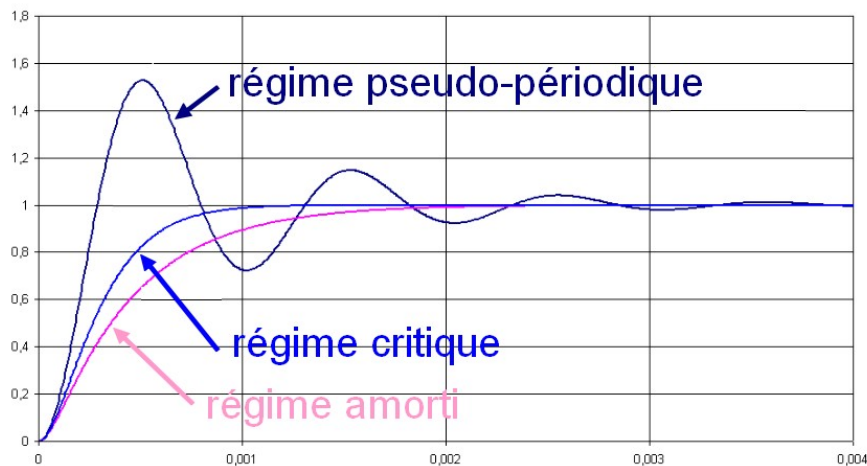
Dans le cas de la résolution numérique obtenu par discrétisation et en utilisant la méthode des différences finies, une étude graphique simple menée avec Excel (reponse_echelon.xlsx onglet « 1^{er} ordre »), montre que, hors mis sur les premiers points, la courbe obtenue par discrétisation est très proche de la solution analytique. On retrouve les valeurs caractéristiques connues de la précision à 5%, de la valeur de la tangente à l'origine et du point caractéristique ($\tau, 0.63$).



L'erreur relative due à la discrétisation s'amenuise très rapidement de 5% à 0%. Cela peut-être sembler beaucoup, mais, Il est bon de rappeler que dans le cas d'un filtre construit à partir de composants électroniques analogiques, les caractéristiques des composants sont rarement garanties à moins de 10%...

3.1.2 Etude d'un filtre du 2^{ème} ordre passe bas (réponse à un échelon)

Pour ce genre de filtre l'équation aux dérivées ne résout pas simplement en une fonction. Suivant les coefficients R, L et C, nous avons trois équations distincts caractérisées par



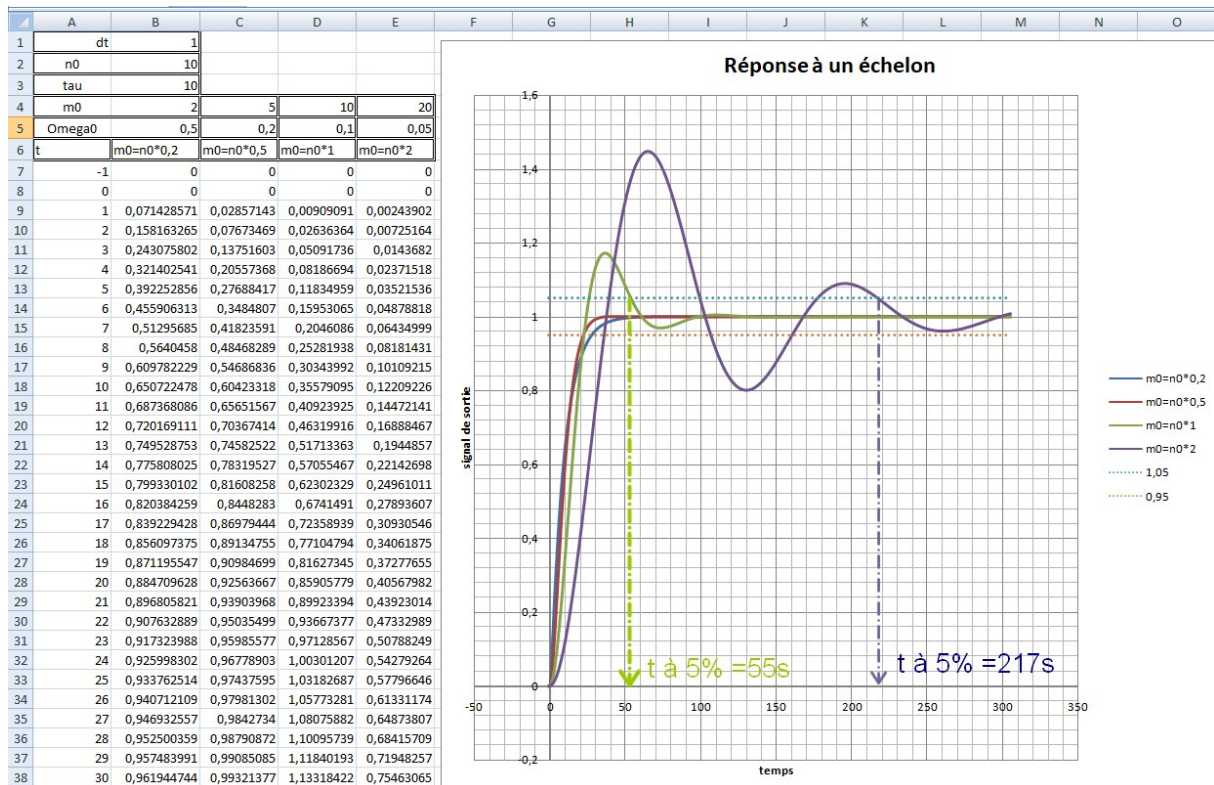
La littérature et le monde du net fourmillent de ces démonstrations que je ne reprendrais pas ici (<http://remy.duperray.free.fr/page24/page6/files/COURS-Echelon-tension.pdf> par exemple).

Régime amorti	Régime critique	Régime pseudopériodique
$R > 2\sqrt{\frac{L}{C}}$	$R = 2\sqrt{\frac{L}{C}}$	$R < 2\sqrt{\frac{L}{C}}$

En reprenant les coefficients n_0 et m_0 définis auparavant, on montre aisément que l'on a les trois régimes en fonction n_0 et m_0 :

Régime amorti	Régime critique	Régime pseudopériodique
$n_0 > 2m_0$	$n_0 = 2m_0$	$n_0 < 2m_0$

Dans le cas de la résolution numérique obtenu par discrétisation et en utilisant la méthode des différences finies, nous n'avons pas à nous occuper de ces trois cas distincts. Suivant les valeurs des coefficients n_0 et m_0 nous évoluons sans heurts entre les trois régimes sans se soucier de la forme exacte de la solution mathématique. La simulation ci-dessous menée avec Excel (reponse_echelon.xlsx onglet « 2eme ordre ») donne le résultat ci-dessous :



Pour ces trois régimes, la pente à $t=0$ est nulle. La vitesse de montée initiale du signal en sortie de filtre dépend du facteur τ et donc de $n_0 \Delta t$

3.2 Réponse en fréquence

Dans cette partie, on ne s'intéressera qu'au clone numérique généré par Excel. Comme je ne présume en rien la puissance de calcul de votre PC (ou Mac), pour vous éviter des plantages et désagréments, basculez vos fichiers Excel en mode calcul manuel sur demander comme indiqué dans le chapitre 0.

Je vous laisse le soin de brancher votre oscilloscope GBF et de fouiller dans vos composants électroniques pour comparer les résultats de la simulation numérique à la réalité vrai de l'électronique analogique.

La grosse différence entre utiliser un filtre analogique ou un filtre numérique numérique dans le traitement de signaux réside dans la plage de fréquence de validité de ce filtre. Dans le cas du filtre analogique elle est limité par la qualité des composants électriques et la manière de les agencer pour éviter les interférences (notamment à hautes fréquences). Dans le cas du filtre numérique, nous sommes limité en fréquence par le sacrosaint théorème de Niquist ([C.F. annexe 2](#)) et par la rapidité de calcul du processeur et interface d'acquisition employé.

Dans les fichiers Excel qui sous-tendent la suite de cet exposé, nous avons limités les fréquences représentables conformément au théorème de Niquist-Shanon

3.2.1 Réponse en fréquence d'un filtre du premier ordre

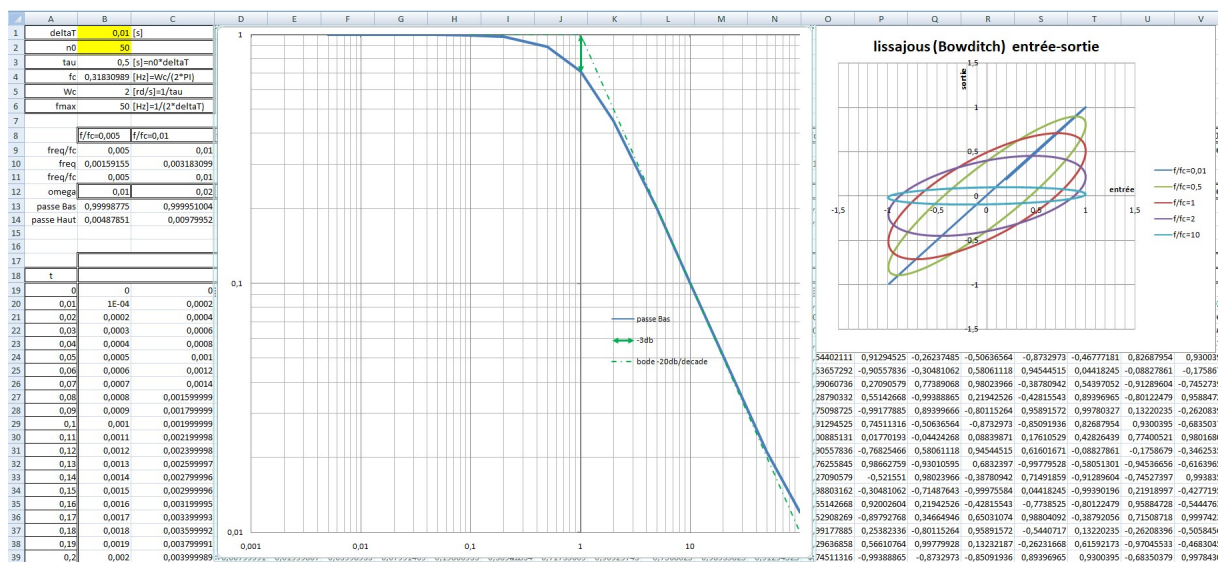
Il est important de visualiser l'atténuation en amplitude en fonction de la fréquence et tout aussi la variation de phase entre le signal d'entrée et le signal de sortie.

Pour cette dernière visualisation, nous utiliserons un fac-similé de courbes de lissajous générées à partir des deux séries de valeurs entrée sortie. On rappelle que dans les « oscilo » de grand papa (donc le mien !), l'ellipse de lissajous produite par le mode x-y avec l'entrée en x et la sortie en y se déforme au gré de la phase.

les axes de l'ellipse s'inclinent et miracle les axes deviennent confondus avec ox et oy pour les phase

- 0 : la trace est un segment horizontal (ou vertical suivant les branchements).
- $\pi/2$: la trace est une ellipse avec ses axes confondus avec ceux de l'écran.
- π : la trace est un segment horizontal (ou vertical comme pour la phase nulle).

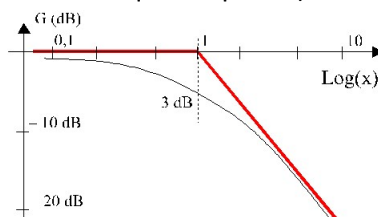
Allez on se lance avec Excel :



On repère :

- le passage à la fréquence critique à -3db et l'approximation courante des filtres du premier ordre : Un segment à gain 0 suivi d'une demi droite avec une pente en puissance de -20db par décade en puissance. En effet une décroissance en tension d'un facteur 10 produit une décroissance en puissance d'un facteur 100 soit 20 db. $P = \frac{U^2}{R}$ et, à R constant

- Droite horizontale suivie d'une droite de pente -20db par décade (division de la grandeur par 10 pour une multiplication de la fréquence par 10).



Ce diagramme s'appelle le diagramme asymptotique de Bode.

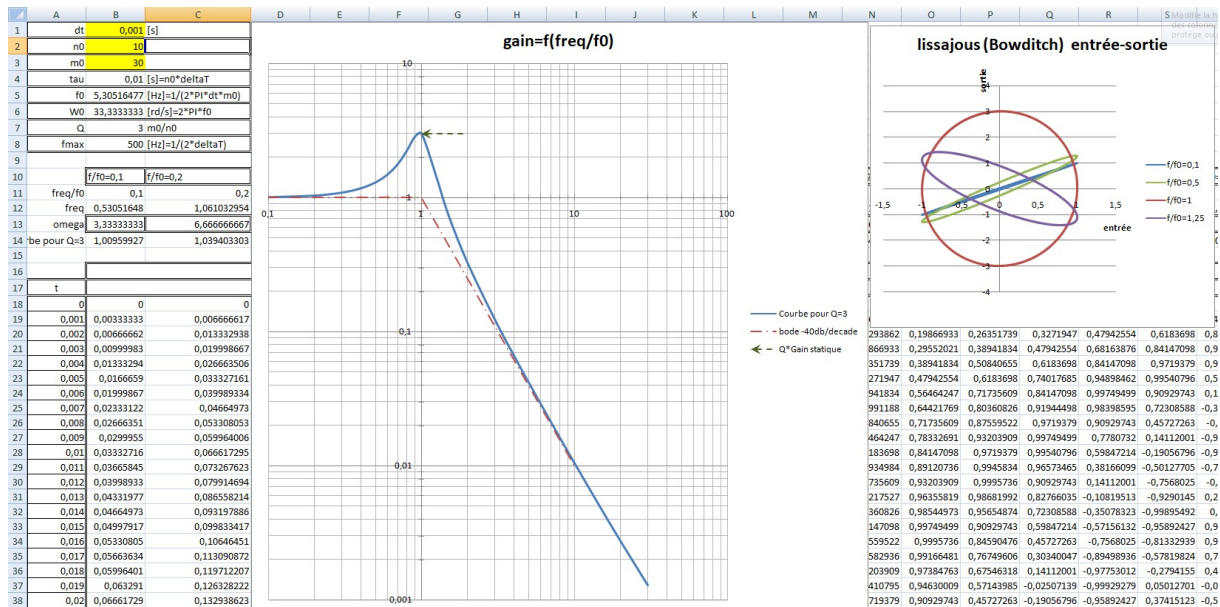
Dans le diagramme asymptotique, on réduit la courbe caractéristique de l'amplitude de la fonction de transfert à ses asymptotes et points caractéristiques. (C.F. Annexe 4)

$$10 \log \left(\frac{P_1}{P_2} \right) = 20 \log \left(\frac{U_1}{U_2} \right) = \Delta db$$

3.2.2 Réponse en fréquence d'un filtre du deuxième ordre

Les généralités sont ici les mêmes que pour le filtre du premier ordre.

Commençons tout de suite avec une vue du résultat Excel



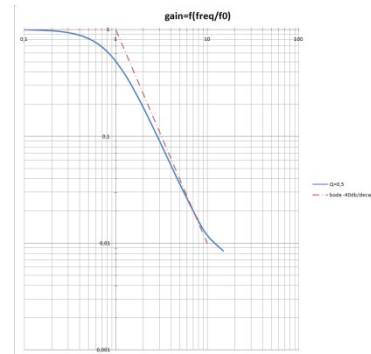
On repère :

- le passage à la fréquence critique avec un dépassement qui dépend de la qualité du filtre (Dépassement important si $Q > 1$ et amortissement si $Q < 0.8$).

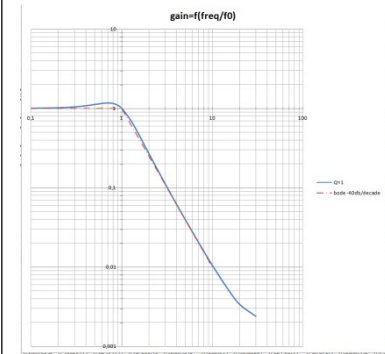
La valeur du dépassement peut aussi se voir sur la courbe 'lissajous' pour $f/f_0 = 1$ en comparant les valeurs sur les axes (ici un facteur 3) soit $20 \log(3) = 9db$.

- L'approximation de Bode des filtres du second ordre : Un segment à gain 0 suivi d'une demi droite avec une pente en puissance de -40db par décade.

Q=0.5



Q=1

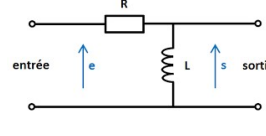
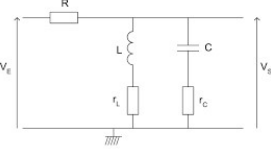
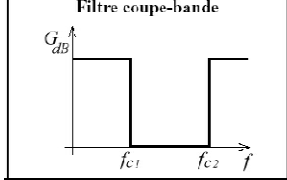
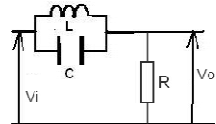
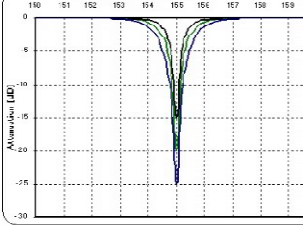


Ces filtres sont plus efficaces de par leur pente asymptotique plus importante. Ils necessite néanmoins plus de calcul numériques (plus de multiplications et divisions) et donc seront à éviter dans le cas de traitements à fréquence élevée d'échantillonnage.

4 Et les autres filtres ?

Dans le chapitre 2.1 nous avons annoncé fièrement l'existence de quatre types de filtres et nous sommes désormais sur notre faim pour trois d'entre eux : le filtre passe haut, le filtre passe bande et le filtre coupe bande.

La réalisation de ces trois filtres peut se faire en utilisant des composants analogiques :

Filtre LC pour un passe haut	Filtre LC suivi d'un filtre RC pour un passe bande	Filtre Bouchon pour un coupe bande
		<div> <p>Filtre coupe-bande</p>  <p>Exemple de solution pour un filtre réjecteur</p>  $\frac{V_0}{V_i} = \frac{1 - LC\omega^2}{1 + (L\omega/R) - LC\omega^2}$ </div> 

Tous ces filtres fonctionnent en éliminant sous forme de chaleur l'énergie des fréquences qu'ils coupent. Une étude électrique suivie d'une discréditation est possible pour chacun de ces filtres. Comme nous avons déjà assez transpiré, nous allons nous appuyer sur ce que nous connaissons déjà et faire appel à un peu de réflexion.

Dans le domaine du numérique, le passage ou la coupure de fréquences ne provoque pas plus de déperdition d'énergie !

En y regardant de près, un filtre passe haut est un filtre qui récupère tout sauf (on dirait une règle de grammaire française !) ce que laisse passer le filtre passe bas correspondant.

4.1 Les filtres passe Haut

Un filtre passe haut (*pHaut*) est « l'opposé » d'un filtre passe bas (*pBas*).

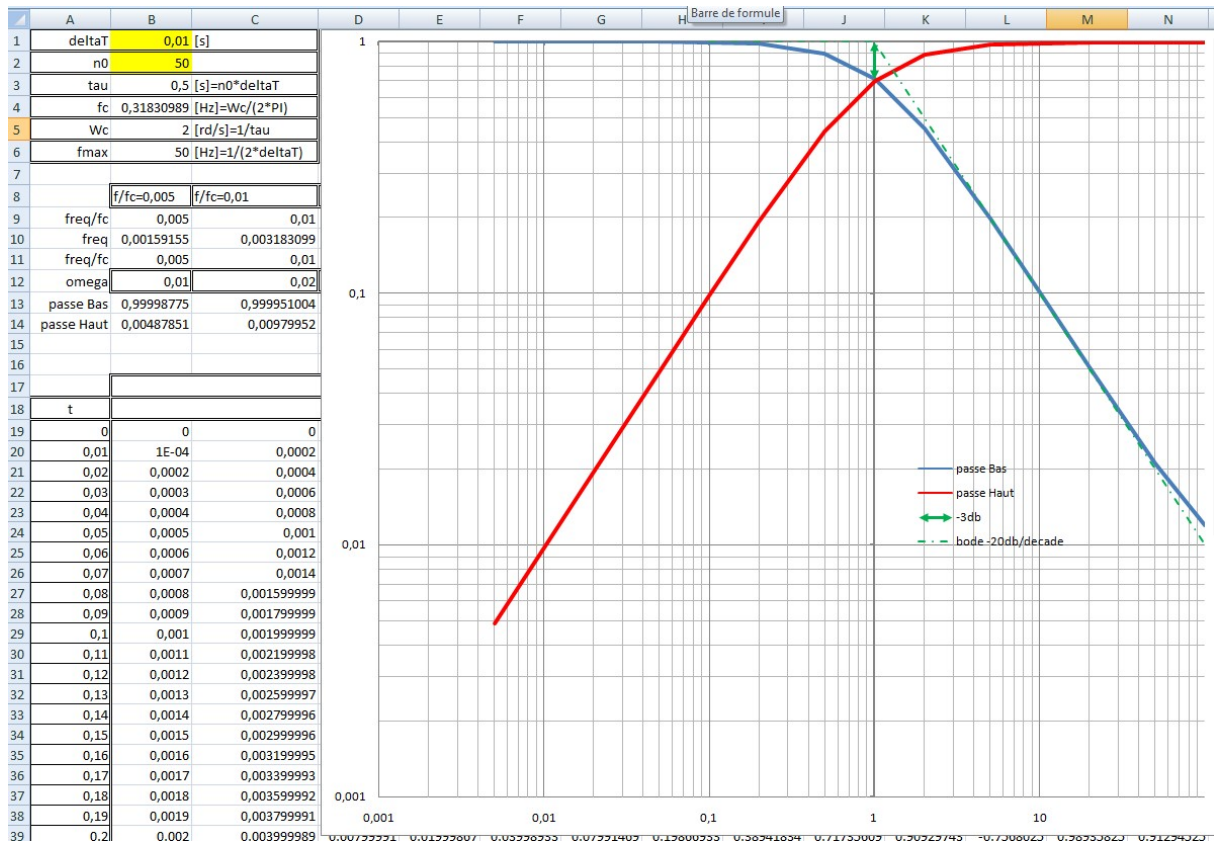
Nous pouvons donc affirmer que $pHaut(signal, fc) = G_{s_{0,bas}} * signal - pBas(signal, fc)$ avec

$G_{s_{0,bas}} = \left[\frac{sortie}{entree} \right]_{freq=0}$ le gain statique du filtre passe bas, et, fc la fréquence de coupure du filtre.

On notera que si $G_{s_{\infty,haut}} = \left[\frac{sortie}{entree} \right]_{freq=\infty}$, le gain statique du filtre passe haut ainsi calculé vaut

$$G_{s_{\infty,haut}} = G_{s_{0,bas}} \cdot$$

Une vue du fichier Excel qui retrace cette astuce



4.2 Les filtres passe Bande

Si nous voulons garder que la bande de fréquence $[fc1, fc2]$ avec $fc1 < fc2$

Il est facile de voir que

$$pBande(signal, fc1, fc2) = G_{s_{0,bas}} * signal - pBas(signal, fc1) - pHaut(signal, fc2) \text{ soit}$$

$$pBande(signal, fc1, fc2) = G_{s_{0,bas}} * signal - pBas(signal, fc1) - (G_{s_{0,bas}} * signal - pBas(signal, fc2))$$

soit : $pBande(signal, fc1, fc2) = pBas(signal, fc2) - pBas(signal, fc1)$

avec $fc1$ et $fc2$ suffisamment éloignés, on a $G_{s_{[fc1, fc2], bande}} = \left[\frac{\text{sortie}}{\text{entree}} \right]_{[fc1, fc2]} = G_{s_{0,bas}}$

4.3 Les filtres coupe Bande

De même que pour les filtres passe haut nous pouvons dire que

$$coupeBande(signal, fc1, fc2) = passeBas(signal, fc1) + passeHaut(signal, fc2) \text{ soit}$$

$$coupeBande(signal, fc1, fc2) = passeBas(signal, fc1) + (G_{s_{0,bas}} * signal - passeBas(signal, fc2))$$

de même que précédemment, avec $fc1$ et $fc2$ suffisamment éloignés, on a

$$G_{s_{[fc1, fc2], coupebande}} = \left[\frac{\text{sortie}}{\text{entree}} \right]_{[0, fc1] \cup [fc2, \infty[} = G_{s_{0,bas}}$$

Cela ressemble de plus en plus à une arithmétique simple !

4.4 Récapitulatif en utilisant nos filtres du §3

Par chance (allez, je l'avoue, c'était prémédité !) les gains statiques de nos filtres du §3 sont de 1 !

Avec $fc1$ et $fc2$ suffisamment distant, nous avons $G_{s_{0,bas}} = \left[\frac{\text{sortie}}{\text{entree}} \right]_{\text{freq}=0} = 1$

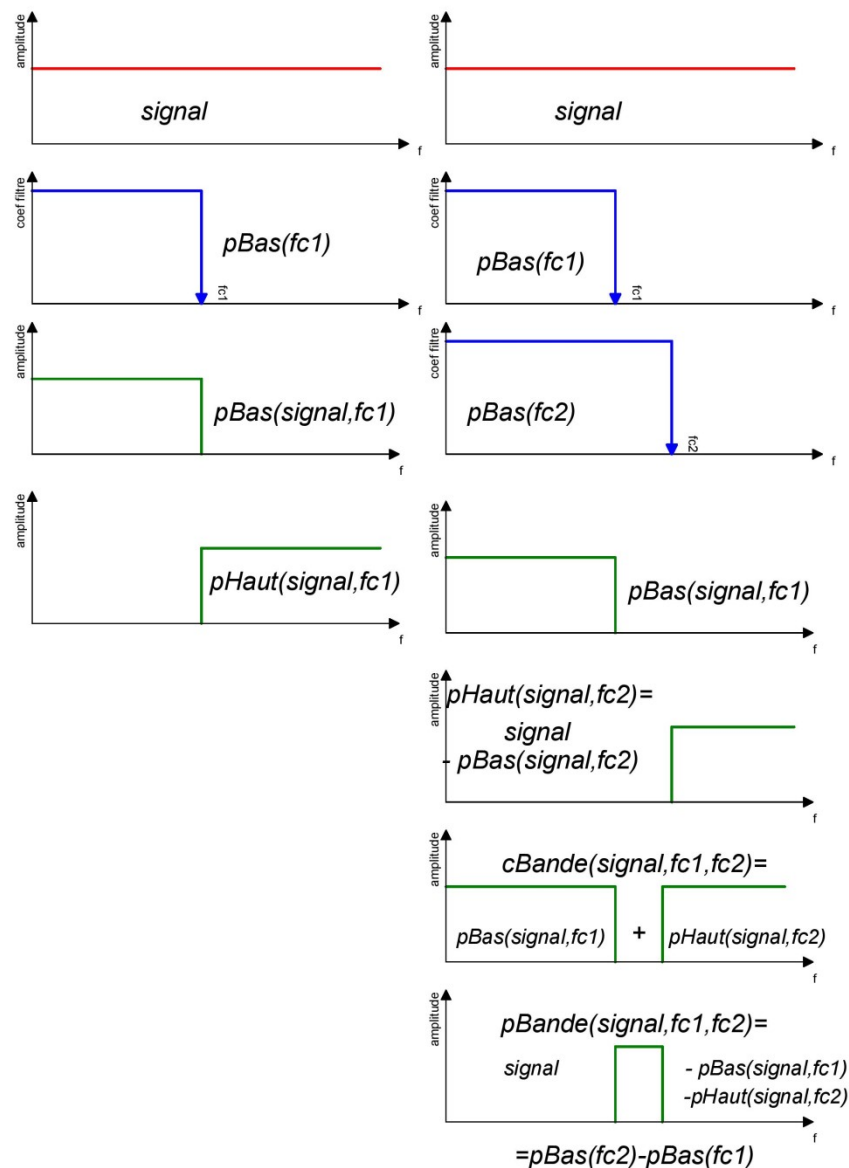
En simplifiant nos équations précédentes :

$$pHaut(\text{signal}, fc) = \text{signal} - pBas(\text{signal}, fc)$$

$$pBande(\text{signal}, fc1, fc2) = pBas(\text{signal}, fc2) - pBas(\text{signal}, fc1)$$

$$coupeBande(\text{signal}, fc1, fc2) = pBas(\text{signal}, fc1) + \text{signal} - pBas(\text{signal}, fc2)$$

Un petit résumé en images :



En rajoutant quelques points virgules et quelques petits mots doux du style void, float ... on arrive à une bibliothèque c++ (mais laissons cela au [chapitre 6](#) suivant.)

4.5 Création de filtres numériques à partir des équations des fonctions de transfert

Bon Résumons notre démarche actuelle :

- 1 : Le schéma électrique du filtre
- 2 : Mise en équation et obtention de l'équation différentielle du filtre
- 3 : Discrétisation
- 4 : Création du modèle numérique équivalent.

Mais dans la littérature, la plupart des filtres sont définis par leur fonction de transfert exprimées en fonction complexes (avec des coefficients intégrant des termes possédant des parties complexes $j\omega, \omega^2 \dots$) ou en exprimés en fonctions de Laplace (avec de $p, p^2 \dots$). Ces objets mathématiques ont bien été issus de circuits électriques réels.

Si la démarche de passer de l'équation différentielle à ces formulations plus abstraites (c'est un point de vue ...), a pu être faites, il est possible de penser que l'opération peut être menée dans l'autre sens (rétro Engineering ou espionnage industriel ...). Dans des conditions restrictives que ne manquerons pas de m'opposer les mathématiciens pur de pur, mais qui dans la vie courante du technicien ou de l'ingénieur sont quasi toujours réalisées (un système est au repos à l'instant initial), il faut se baser sur le petit tableau suivant qui permet de passer de l'une à l'autre des modélisations mathématiques :

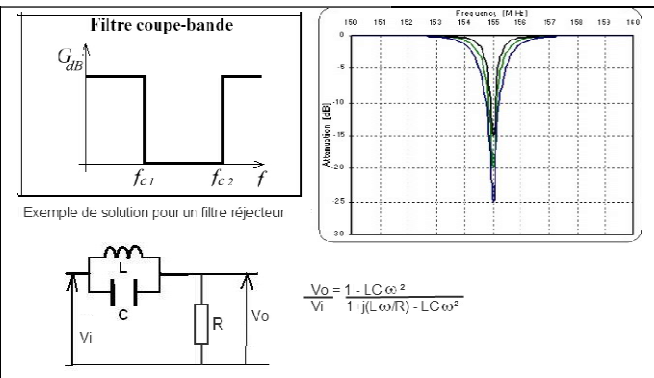
Domaine temporel réponse en fréquence	Domaine fréquentiel complexe	Domaine de Laplace
$k(\sin(\omega t))$	$\underline{k}(e^{j\omega t})$	$K(p) = L(k(\omega t))$
$k(\sin(\omega t)) = a \cdot f(\sin(\omega t)) + b \cdot g(\sin(\omega t))$	$\underline{k}(e^{j\omega t}) = a \cdot \underline{f}(e^{j\omega t}) + b \cdot \underline{g}(e^{j\omega t})$	$K(p) = a \cdot F(p) + b \cdot G(p)$
$\frac{df(\sin(\omega t))}{dt}$	$j\omega \underline{f}(e^{j\omega t})$	$p \cdot K(p) = L(k(\omega t))$
$\frac{d^2 f(\sin(\omega t))}{dt^2} = \frac{d}{dt} \left(\frac{df(\sin(\omega t))}{dt} \right)$	$(j\omega)^2 \underline{f}(e^{j\omega t}) = -j\omega^2 \underline{f}(e^{j\omega t})$	$p^2 K(p) = pL\left(\frac{dk(\omega t)}{dt}\right)$
$\int f(\sin(\omega t))dt$	$\frac{1}{j\omega} \underline{f}(e^{j\omega t}) = -\frac{1}{\omega} \underline{f}(e^{j\omega t})$ <i>nota : $j^2 = -1$</i>	$\frac{1}{p} K(p)$

Tout d'abord, il est nécessaire que la fonction de transfert soit exprimée de manière canonique (ce qui souvent la norme les 'catalogues' de fonctions de transfert) soit :

- dans le domaine complexe sous la forme :
$$\underline{H} = \frac{\underline{S}(e^{j\omega t + \varphi})}{\underline{E}(e^{j\omega t})} = \frac{K}{(j\omega)^\alpha} \frac{\sum_{i=0}^{i=n} a_i (j\omega)^i}{\sum_{i=0}^{i=m} b_i (j\omega)^i}$$
- dans l'espace de Laplace sous la forme :
$$H(p) = \frac{S(p)}{E(p)} = \frac{K}{p^\alpha} \frac{\sum_{i=0}^{i=n} a_i p^i}{\sum_{i=0}^{i=m} b_i p^i}$$

On notera que dans le domaine complexe les termes $a_i (j\omega)^i$ donne lieu à des termes imaginaires pur si i est impair et à des termes réels si i est pair (idem pour les termes $b_i (j\omega)^i$).

Nous allons effectuer l'opération sur un exemple déjà évoqué précédemment au §4 sur le filtre coupe bande.



Sa fonction de transfert dans le domaine fréquentiel (en mode complexe) et donnée par

$$\frac{\underline{out}(e^{j\omega t + \varphi})}{\underline{in}(e^{j\omega t})} = \frac{1 - LC\omega^2}{1 + j\frac{L\omega}{R} - LC\omega^2} \text{ avec } \omega \text{ la pulsation du signal d'entrée (en rd/s) et } \varphi \text{ le déphasage}$$

du signal de sortie par rapport au signal d'entrée. Soit en regroupant les termes constants, $j\omega$, $j^2\omega^2$, ..., $j^n\omega^n$ en se souvenant que $j^2 = -1$, et en séparant E et S :

$$\underline{out}(e^{j\omega t + \varphi}) \left(1 + j\frac{L\omega}{R} - LC\omega^2 \right) = \underline{in}(e^{j\omega t}) (1 - LC\omega^2). \text{ On sépare les termes :}$$

$\underline{out} + \frac{L}{R} j\omega \cdot \underline{out} + LCj^2\omega^2 \underline{out} = \underline{in} + LCj^2\omega^2 \underline{in}$ En regardant le tableau ci-dessus, on identifie les termes, on repasse dans le domaine temporel et on retrouve l'équation différentielle de ce circuit :

$$out(t) + \frac{L}{R} \frac{d out(t)}{dt} + LC \frac{d^2 out(t)}{dt^2} = in(t) + LC \frac{d^2 in(t)}{dt^2} \text{ où avec la notation des physiciens :}$$

$$out(t) + \frac{L}{R} \dot{out}(t) + LC \ddot{out}(t) = in(t) + LC \ddot{in}(t)$$

Reprenons notre cuisine avec l'alter-égo de la transformée de Laplace.

La fonction de transfert dans le domaine de Laplace est elle donnée par l'équation :

$$\frac{OUT(p)}{IN(p)} = \frac{1 + p^2 LC}{1 + p \frac{L}{R} + p^2 LC} . \text{ En effectuant des opérations de calcul similaires, on obtient}$$

$OUT(p) \left(1 + p \frac{L}{R} + p^2 LC \right) = IN(p) (1 + p^2 LC)$ et, on retrouve par identification, la même équation différentielle du domaine temporel (heureusement !).

Et c'est parti pour la discrétisation..

En notant S_n et E_n les valeurs numériques prises par la sortie et l'entrée et en appliquant la méthode déjà utilisée pour les filtres du premier et deuxième ordre nous avons :

$$S_0 + \frac{L}{R} \left(\frac{S_0 - S_{-1}}{\Delta t} \right) + LC \left(\frac{S_0 - 2S_{-1} + S_{-2}}{\Delta t^2} \right) = E_0 + LC \left(\frac{E_0 - 2E_{-1} + E_{-2}}{\Delta t^2} \right)$$

Nous savons que la pulsation de coupure de bande de ce filtre correspond à $\omega_0 = \frac{1}{\sqrt{LC}}$; $LC = \frac{1}{\omega_0^2}$

avec $\tau = RC$ il vient en réorganisant les termes

$$\left(1 + \frac{L}{R\Delta t} + \frac{LC}{\Delta t^2} \right) S_0 + \left(-\frac{L}{R\Delta t} - \frac{2LC}{\Delta t^2} \right) S_{-1} + \left(\frac{LC}{\Delta t^2} \right) S_{-2} = \left(1 + \frac{LC}{\Delta t^2} \right) E_0 + \left(-\frac{2LC}{\Delta t^2} \right) E_{-1} + \frac{LC}{\Delta t^2} E_{-2}$$

en prenant des coefficients n_0 et m_0 tels que $\tau = n_0 \Delta t = RC$ et $\omega_0 = \frac{1}{m_0 \Delta t} = \frac{1}{\sqrt{LC}}$ nous obtenons :

$$S_0 = \frac{1}{\left(1 + \frac{m_0^2}{n_0} + m_0^2 \right)} \left(\left(\frac{m_0^2}{n_0} + 2m_0^2 \right) S_{-1} - m_0^2 S_{-2} + (1 + m_0^2) E_0 - 2m_0^2 E_{-1} + m_0^2 E_{-2} \right), \text{ ou bien, avec un}$$

$$\text{minimum d'opérations de multiplications } S_0 = \frac{\left(\frac{1}{n_0} + 2 \right) S_{-1} - S_{-2} + \left(\frac{1}{m_0^2} + 1 \right) E_0 - 2E_{-1} + E_{-2}}{\frac{1}{m_0^2} + \frac{1}{n_0} + 1}$$

Remarques :

- Nous avons choisi des coefficients adimensionnels n_0 et m_0 analogue à ceux choisis dans le filtre du deuxième ordre.
- Ici la sortie ne dépend pas que de la valeur instantanée de l'entrée mais, joue sur un effet « mémoire » de l'entrée E_n . Ceci arrive très souvent dans les filtres un peu plus complexes que le RLC série ou RC ou RL.

- Si $m_0=0$ ce qui correspond à une fréquence coupée infinie (donc non existante) le filtre se résume à $S_0 = E_0$ (reprendre la formulation non simplifiée !) soit aucun filtre ! Pour les érudits, ceci ne vous rappelle pas un certain « cousinage » avec le théorème de la valeur finale dans le domaine de Laplace ? Pour les autres, constatons uniquement !

- Le facteur de qualité de ce filtre analogique $Q = \frac{Z_{\text{résonnance}}}{X_{\text{réactive}}} = \frac{1}{R} \sqrt{\frac{L}{C}} = \frac{\sqrt{LC}}{RC} = \frac{1}{\omega_0 \tau} = \frac{m_0}{n_0}$.

- La bande passante de ce filtre $B_p = \frac{f_0}{Q} = f_{ch} - f_{cb}$ (plage où l'atténuation est <3db) pourra

aisément se voir sur la simulation numérique. On notera enfin que cette bande passante semble 'équilibrée' en échelle log (sur les fréquences) mais qu'elle ne l'est pas en échelle linéaire. Un solveur d'équation (*dcode*) nous a fourni la solution exacte pour trouver les

fréquences de coupure :

$$\begin{cases} f_{cb} = 0.5 f_0 \sqrt{\frac{4(\tau^2 \omega_0^2 + 0.5)}{\tau^2 \omega_0^2} - 2} \sqrt{\frac{4(\tau^2 \omega_0^2 + 0.5)^2}{\tau^4 \omega_0^4} - 4} \\ f_{ch} = 0.5 f_0 \sqrt{\frac{4(\tau^2 \omega_0^2 + 0.5)}{\tau^2 \omega_0^2} + 2} \sqrt{\frac{4(\tau^2 \omega_0^2 + 0.5)^2}{\tau^4 \omega_0^4} - 4} \end{cases}$$

La remarque graphique ci-dessus $\log\left(\frac{f_0}{f_{cb}}\right) = \log\left(\frac{f_{ch}}{f_0}\right)$ et $Q = \frac{f_0}{f_{ch} - f_{cb}}$ nous fourni les équations

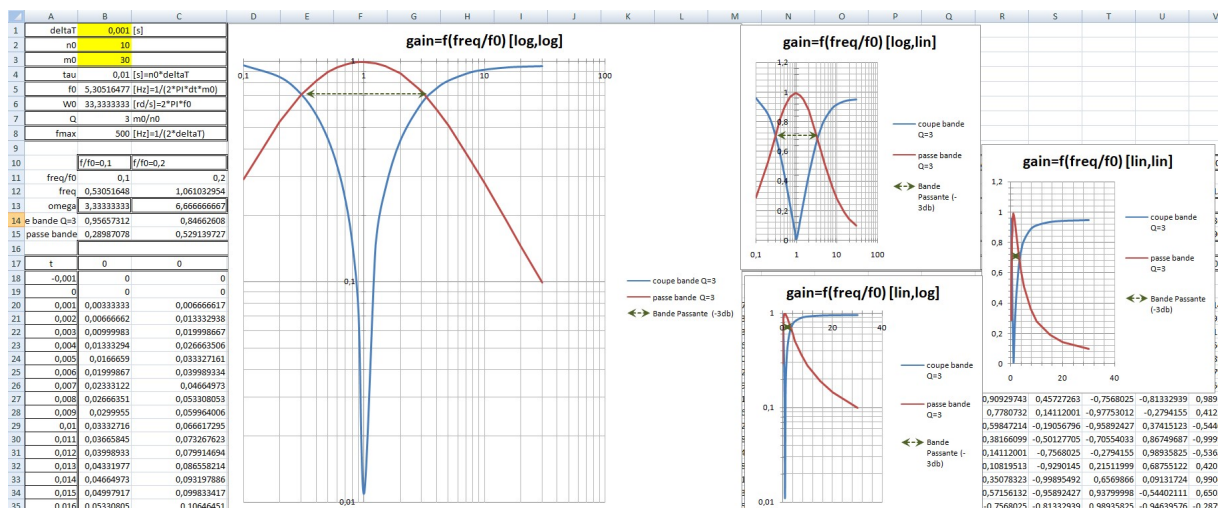
nettement plus exploitable :

$$\begin{cases} f_{ch} = f_0 \frac{\sqrt{4Q^2 + 1} + 1}{2Q} \text{ et } f_{cb} = f_0 \frac{\sqrt{4Q^2 + 1} - 1}{2Q} \\ f_0 = \sqrt{f_{ch} f_{cb}} \text{ et } Q = \frac{\sqrt{f_{ch} f_{cb}}}{f_{ch} - f_{cb}} \end{cases}$$

que nous utiliserons dans la

bibliothèque.

On est parti avec la simulation Excel.



Sur la même simulation vous trouvez le filtre coupe bande (en bleu) et le filtre complémentaire passe bande (en rouge) réalisé de manière numérique par la désormais célèbre formule

$$\text{passeBande}(\text{signal}, f_{c1}) = \text{signal} * G_{\text{coupe bande}} - \text{coupeBande}(\text{signal}, f_{c1})$$

5 Et si nous parlions d'ordres supérieurs à 2

La conception de filtres numériques jusqu'alors se base sur une modélisation des filtres analogiques classiques (il en existe autant que votre imagination peut mettre en série et parallèle des selfs, capacités et résistors auquel les amplis opérationnels on dopé l'imagination ...) Ces mises en équation peuvent faire apparaître des dérivées troisièmes, quatrièmes ... On parle alors de filtres du 3^{ème} ordre ou du 4^{ème} ordre. La mise en équation par discréditation est possible mais elle nécessite un grand nombre de calculs gourmand en temps de la part du microprocesseur.

Il est néanmoins possible à l'aide des filtres de base que nous avons analysés de réaliser des filtres d'ordre supérieurs en les cascadeant à la manière des poupées russes.

monFiltre(signal,fc)=filtrePremierOrdre(filtrePremierOrdre(signal,fc),fc) réalise un filtre du second ordre avec un facteur de qualité voisin de 0.5

Numériquement, on réalise comme cela l'équivalent de la mise en série de filtres élémentaires.

En ce qui concerne la mise en parallèle (avec un ampli op en mode sommateur), nous l'avons déjà réalisé en créant les filtres coupeBande du premier ordre).

Votre imagination est donc décuplée par des opérations arithmétiques simples entre le résultat des filtres numériques (+,-,*,/) et encore plus si vous intégrez des fonctions du style max(a,b) ou min(a,b).

Ceci pour dire la richesse de filtrage possible en mode numérique (à condition d'avoir une puissance de calcul suffisante) est sans commune mesure face aux solutions analogiques. L'avènement des circuits de traitement de signaux (DSP) profite de cette richesse.

6 La bibliothèque Arduino Filtre

6.1 Principe de base (Problèmes du signal bruité)

Habituellement, lorsque l'on programme un processus qui prend en compte la valeur d'entrées numériques ou analogiques, on a un programme du style :

```
/*
  petit programme simpliste d'utilisation des
  entrées
*/
#define EntreeBool (2) // on a une entrée tout
ou rien (TOR) sur la broche 2 du micro-
contrôleur
#define EntreeAna (A0) // et une entrée
analogique en 10 bits (0->1023) sur l'entrée
analogique A0 du micro-contrôleur
#define seuil (300) // le seuil de
déclenchement de notre entrée analogique

void setup() {
  pinMode(EntreeBool,INPUT_PULLUP);
  // put your setup code here, to run once:
}
void actionTOR()
{
  // Ce qu'il faut faire si EntreeBool est vrai
}
void actionAna()
{
  // Ce qu'il faut faire sir EntreeAna renvoi
une valeur > seuil
}
```

A l'utilisation, avec notre bidouille électronique (un nano, une alim et quelques fils ...), on constate que notre bricoleur de copain (copine) lorsqu'il met en route sa perceuse, le programme ou du moins notre système s'affole et génère une quantité d'actions style actionTOR et actionAna. As-on par mégarde inventé l'intelligence Artificielle ? Notre programme aurait-il pris une conscience ? As-t-il une aversion pour l'étagère qui est en phase de construction ? Non ! Isaac Asimov (C.F. les lois de la robotique) peut dormir tranquille, le diagnostic est bien plus terre à terre que cela. Notre système n'est pas assez blindé du point de vue électrique et reçoit de la part du moteur des impulsions électromagnétiques provoquées par les étincelles des charbons entre autre. Ces impulsions invisibles sont converties en

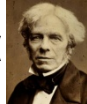
```

}
void loop() {
  if (digitalRead(EntreeBool)) actionTOR();
  if (analogRead(EntreeAna)>seuil) actionAna();
}

```

impulsions de tension par les fils qui relient les éléments de notre bricolage (et évidemment aussi les circuits imprimés de notre nano). Ces tensions parasites sont prises comme étant des signaux valables par notre système.

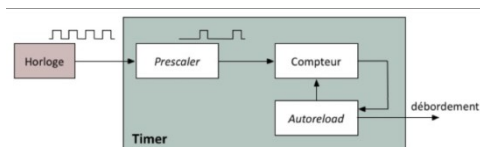
Plusieurs solutions s'offrent à nous :

- 1 Arrêter de jouer les apprentis sorciers et laisser aux vrais sorciers le soin de nous faire payer au prix fort leurs prestations (si vous lisez cet article, c'est que cette option est déjà proscrite !).
- 2 Se prémunir contre la 4^{ème} guerre mondiale et blinder (électriquement) tout notre circuit, le mettre dans une cage de Faraday ( 1791-1867)
- 3 Installer des filtres analogiques à partir de composants discrets sur les entrées (filtres RC ou RL) (il va falloir sortir le fer à souder et identifier les valeurs des composants du tiroir de grand-papa...).
- 4 Revoir notre manière d'acquérir les entrées et les filtrer de manière numérique pour éviter de prendre des artéfacts électriques.
- 5 Mixer les solutions 1 à 4 pour résoudre les problèmes.

6.2 La solution du filtrage numérique

L'idée général n'est pas d'éliminer les parasites électriques et leurs actions sur notre circuit électronique, mais essayer de minimiser leur prise en compte (les noyer dans la masse).

Comme nous l'avons vu précédemment, cette méthode nécessite d'effectuer (en anglais a synchronous sampling) un échantillonnage synchrone (tout les dt temps).



Les microcontrôleurs possèdent des « timers » qui sont des compteurs (décompteurs) câblés greffés sur l'horloge maitresse du circuit via des diviseurs.

Ces « timers » sont très utilisés pour les opérations d'entrée-sortie à fréquence fixe (on pensera aux sorties séries, génération de signaux PWM ...).

Lorsqu'ils arrivent en fin de comptage, ils peuvent si nous le désirons, arrêter le processus principal et lancer un processus en sommeil (processus d'interruption) via une fonction d'interruption.

IL est clair que la fonction d'interruption ne doit pas avoir de boucles infinies (voir pas de boucles indéfinies du tout) pour pouvoir à la fin de cette fonction rendre la main au programme principal et le redémarrer là ou elle l'avait interrompu.

Tout les dt secondes, la système d'interruption lance une fonction qui effectue

- L'acquisition des entrées

- Le traitement (filtrage numérique) de ces entrées.

Le processus principal lui ne verra alors les entrées que par les lunettes roses des filtres numériques qui ont fait leur travail de masquage, dépoussiérage..

6.3 Un exemple commenté

Avant de faire une énumération à la Prévert des fonctions de filtrage de cette bibliothèque, regardons le petit programme introductif modifié par ce concept de filtrage.

```
// *****
// Le même programme qui filtre les entrées
// *****

#include <TimerOne.h> // Intégration de la librairie TimerOne qui gère le timer N°1
#include <FiltreNum.h> // Intégration de la bibliothèque

#define EntreeBool (2) // on a une entrée tout ou rien (TOR) sur la broche 2 du micro-contrôleur
#define EntreeAna (A0) // et une entrée analogique en 10 bits (0->1023) sur l'entrée analogique A0 du
micro-contrôleur

#define seuil (300) // le seuil de déclenchement de notre entrée analogique

#define dt (1000) // dt en microsecondes soit 1ms 0.001s correspond à une fréquence d'acquisition de
1000Hz
// La plage de fréquence d'analyse des filtres est donc au maximum de 1000/2=500Hz (Théorème de
Nyquist)

// Les filtres d'entrées et leur pin d'attachement ou leurs variables de chainages
// déclaration de la broche via une variable
InputType in1 = { in1.pin = EntreeBool };
FilterNumLowPassFirstOrder f1(in1, INPUT_PULLUP, 10); //un filtre numérique avec une fréquence de
coupure de 1/(10*dt*2*PI)=15.9 Hz
FilterTrigger f2(InputType{ .bool_ptr = f1.getValBoolPtr() }, INPUT_PTR_BOOL,400,600,0,1023); // un
filtre trigger de schmidt chaîné au filtre f1 avec ses deux seuils de déclenchement à 400 et 600 et ses
deux valeurs de saturation 0 et 1023
// ce dernier filtre permet d'éviter les clignotant intempestifs de basculement à un seuil fixe.

// autre manière de déclarer la broche
#define fc (20.) // Hz
#define n0 int((1000000 / (dt * 2 * M_PI * fc)) + 0.5)
FilterNumLowPassFirstOrder f3(InputType{ .pin = EntreeAna }, INPUT_ANALOG, n0);
FilterNumLowPassFirstOrder f4(InputType{ .i_ptr = f3.getValIntPtr() }, INPUT_PTR_INT, n0);
// le filtre f4 est chaîné au filtre f3
FilterNumLowPassFirstOrder f5
    (InputType{ .i_ptr = f4.getValIntPtr() }, INPUT_PTR_INT, dt / 1000000., 20.);
// le filtre f5 est chaîné au filtre f4
// il est défini par rapport à la fréquence d'acquisition et la fréquence de coupure.
// le résultat en sortie de f4 est un filtre du deuxième ordre avec un facteur de qualité de 0.5 et une
fréquence de coupure fc.
// en sortie de f5 nous avons un filtre du 3ème ordre (-60 db par décade dans la phase de coupure

void setup() { // Creation de la pile chaînées de filtres à traiter
    addFilter(f1);
    addFilter(f2);
    addFilter(f3);
    // initialisation de ces filtres
    initFilters();
    // mise en place du timer avec sa valeur de selection
    Timer1.initialize(dt);
    // mise en place de la procédure d'interruption (effectuée tout les dt
    // cette fonction s'appelle treatmentFilters dans la bibliothèque Filtre
    Timer1.attachInterrupt(treatmentFilters, 1000);
    // ajouter votre code ici
    Serial.begin(9600); // par exemple
}
```

```

void actionTOR1() {
    // Ce qu'il faut faire si EntreeBool est vrai filtré par f1
}
void actionTOR2() {
    // Ce qu'il faut faire si EntreeBool est vrai filtré par f2(f1)
}
void actionAna1() {
    // Ce qu'il faut faire si EntreeAna renvoi une valeur > seuil filtré par f3
}
void actionAna2() {
    // Ce qu'il faut faire si EntreeAna renvoi une valeur > seuil filtré par f4(f3)
}

void loop() {
    if (f1.getValBool()) actionTOR1();
    if (f2.getValBool()) actionTOR2();
    if (f2.getValInt() > seuil) actionAna1();
    if (f3.getValInt() > seuil) actionAna2();
}

```

Ouille ouille ouille ça se complique ! Regardez de plus près et vous verrez que tout est quasi géré par la bibliothèque. Le code source de cette bibliothèque est néanmoins intéressant à regarder (mais plus tard si il vous reste du temps et de la curiosité.)

6.4 Les filtres de la bibliothèque et leurs méthodes associées

Dans ce paragraphe, nous ne détaillerons que le strict nécessaire. Le lecteur aura fort de se documenter via le fichier source de la bibliothèque « filtreNum.h ». Pour une utilisation normale de la bibliothèque, seul les méthodes et procédures surlignées en jaune sont normalement utilisées.

6.4.1 La hiérarchie des classes (objets)

$$\begin{array}{l}
 \text{FilterObject} \rightarrow \text{FilterNumObject} \rightarrow \left\{ \begin{array}{l}
 \text{FilterNumLowPassFirstOrder} \rightarrow \text{FilterNumHighPassFirstOrder} \\
 \text{FilterNumLowPassSecondOrder} \rightarrow \text{FilterNumHighPassSecondOrder} \\
 \text{FilterNumCutBandSecondOrder} \rightarrow \text{FilterNumPassBandSecondOrder} \\
 \text{FilterNumAmplif} \\
 \text{FilterNumTrigger}
 \end{array} \right.
 \end{array}$$

Classe *FilterObject* :

Les attributs :

int actualValue = 0	La valeur de sortie calculée initialisée à 0
int mediumPlus	Constante calculée pour les transformations int boolean
int mediumMoins	idem
Boolean out	La sortie calculée en mode boolean suivant


Les méthodes :

FilterNumObject()	Créateur de base
boolean getOutBool()	Récupère la valeur en tant que booléen
int getOutVal()	Récupère la valeur de sortie en tant qu'entier
int *getValIntPtr()	Récupère l'adresse de la valeur de sortie entière (pour chainage des filtres)
boolean *getValBoolPtr()	Récupère l'adresse de la valeur de sortie booléenne (pour chainage des filtres)
compute()	Calcul la valeur de sortie
void virtual init()	initialisation du filtre
int *getAdrOut()	utilisé pour chaîner les filtres via une valeur de type int
int getInput()	actualisation des entrées

Classe *FilterNumObject*

Les attributs :

InputType input une variable composite une broche ou une variable
byte mode = INPUT défini le type d'entrée : le mode pin une broche le type de variable si variable
int valIn = 0; la valeur de l'entrée actualisée par getInput() initialisée à 0

(conditions de Heaviside  1850-1925)

float fReelDt = 0; la fréquence de référence si besoin
float QReel = 1 la qualité du filtre si besoin

Les Méthodes :

FilterNumObject() Le constructeur de base

FilterNumObject(InputType in, byte inMode) Le constructeur

void init() Initialisation du filtre

int getInput() actualisation des entrées gère les types d'entrées

float getFreqRef() La fréquence de référence (coupure ...

float getFreqf1() Si plusieurs fréquences

float getFreqf2()

float getQuality() Le facteur de qualité

Class FilterNumLowPassFirstOrder :

Les méthodes :

FilterNumLowPassFirstOrder() Le constructeur de base

FilterNumLowPassFirstOrder(InputType in, byte inMode, int n0) Le constructeur défini avec n0

FilterNumLowPassFirstOrder(InputType in, byte inMode, float dt, float fc) Le constructeur défini par la fréquence
compute() calcul de la valeur de sortie du filtre

Class FilterNumHighPassFirstOrder:

Les méthodes :

FilterNumHighPassFirstOrder(InputType in, byte inMode, int n0) constructeur n0 ; $fc=1/(2*\pi*n0*dt)$

FilterNumHighPassFirstOrder(InputType in, byte inMode, float dt, float fc) définition en fréquence de dt $fc=1/(2*\pi*n0*dt)$

compute() calcul de la valeur de sortie du filtre

init() initialisation du filtre

Class FilterNumLowPassSecondOrder:

Les méthodes :

FilterNumLowPassSecondOrder() Constructeur de base

FilterNumLowPassSecondOrder(InputType in, byte inMode, int n0, int m0) constructeur avec la définition en n0, m0

FilterNumLowPassSecondOrder(InputType in, byte inMode, float dt, float fc, float Q) définition en fréquence de dt $fc=1/(2*\pi*n0*dt)$ et qualité

compute() calcul de la valeur de sortie du filtre

getQreel() renvoi la valeur calculée de Q

Class FilterNumHighPassSecondOrder:

Les méthodes

FilterNumHighPassSecondOrder(InputType in, byte inMode, int n0, int m0) constructeur avec la définition en n0, m0
 $Q=n0/m0$ $f0=1/(2*\pi*m0*dt)$

FilterNumHighPassSecondOrder(InputType in, byte inMode, float dt, float f0, float Q) définition en fréquence de dt
 $fc=1/(2*\pi*n0*dt)$ et qualité

compute() calcul de la valeur de sortie du filtre

Class FilterNumCutBandSecondOrder:

Les méthodes

FilterNumCutBandSecondOrder() Le constructeur de base

FilterNumCutBandSecondOrder(InputType in, byte inMode, int n0, int m0) constructeur avec la définition en n0, m0
 $Q=n0/m0$ $f0=1/(2*\pi*m0*dt)$

`FilterNumCutBandSecondOrder(InputType in , byte inMode, float dt, float fc, float Q)` définition en fréquence de dt
 $fc=1/(2*\pi*n0*dt)$ et qualité

`FilterNumCutBandSecondOrder(InputType in , byte inMode, float dt, float *f1, float *f2)` définition en bande de fréquence
 $f1 < f2$

`compute()` calcul de la valeur de sortie du filtre
`getFreqf1()` retourne f1 calculé
`getFreqf2()` retourne f2 calculé
`getQreel()` renvoi la valeur calculée de Q

Class FilterNumPassBandSecondOrder:

Les méthodes :

`FilterNumPassBandSecondOrder(InputType in , byte inMode, int n0, int m0)` constructeur avec la définition en n0, m0
 $Q=n0/m0$ $f0=1/(2*\pi*m0*dt)$

`FilterNumPassBandSecondOrder(InputType in , byte inMode, float dt, float f0, float Q0)` définition en fréquence et dt
 $fc=1/(2*\pi*n0*dt)$ et qualité

`FilterNumPassBandSecondOrder(InputType in , byte inMode, double dt, float *f1, float *f2)` définition en bande de
fréquence $f1 < f2$

`compute()` calcul de la valeur de sortie du filtre

Class FilterNumAmplif:

Les méthodes :

`FilterNumAmplif(InputType in , byte inMode, float v)` constructeur avec facteur amplification
`compute()` calcul de la valeur de sortie du filtre

Class FilterNumTrigger:

Les méthodes :

`FilterNumTrigger(InputType in , byte inMode, int sb, int sh, int satb, int sath)` constructeur avec la valeur de seuil bas,
seuil haut, saturation basse, saturation haute

`compute()` calcul de la valeur de sortie du filtre
`int getOutVal()` récupère la valeur de sortie en tant qu'entier
`bool getOutBool()` récupère la valeur booléenne de sortie

****Nota**** Pour ce dernier objet, une déclaration du type `FilterNumTrigger(xx,yy,512,512,1023,0)` se comportera comme une porte logique Non.

6.4.2 Les variables globales, procedures et fonctions utiles :

`filter * stackFilter` La variable qui sert à actionner tous les filtres. **Ne pas toucher !!**

`addFilter(FilterObject * f)` ajoute un filtre à la pile des filtres

`void takelnputs()` La procédure de prise des entrées

`void treatmentFilters()` La procédure de traitement des filtres

`void initFilters()` initialisation de la pile de filtres

La procédure d'interruption à greffer au timer

typiquement avec la bibliothèque timerone

mise en place du timer avec sa valeur de selection

Timer1.initialize(dt); // mise en place de la procédure d'interruption (effectuée tout les dt

Timer1.attachInterrupt(treatmentFilters, 1000); // attache la procedure treatmentFilters au timer

`void interTimer()` Le maitre à danser que l'on ne vois pas lors de l'exécution du ballet !

6.5 Les différents temps d'exécution

Vous aviez déjà compris qu'il fallait jongler entre le paramètre dt (période de scrutation des entrées géré par le timer) et les différents nombre $n0$, $m0$ pour avoir le filtre qui convient à notre problème. Un paramètre supplémentaire se greffe à la fête : le temps d'exécution (de calcul) d'un filtre. En effet, l'exécution du code de calcul de ces filtres est indépendante de l'exécution du programme principal, mais néanmoins la somme des temps d'exécution du code de calcul de ces filtres ne peut être supérieur à dt la période de scrutation des entrées sous réserve de bloquer complètement le programme principal et de provoquer un plantage « royal » de notre système multitâche (eh oui !) implémenté sur un processeur mono tâche et surtout monoprocesseur. Pour histoire c'est le

problème qu'a rencontré l'astronaute neil Amstrong (1930-2012) lorsqu'il s'est posé sur la

lune avec le module lunaire d'Apollo 11 (1969) (l'AGC) et sa célèbre erreur de erreur 1202 !). Il convient donc de connaître les temps calculs des différents filtres

Voici les valeurs relevées sur un Arduino Uno pour le fonctionnement interne du filtre hors processus d'acquisition.

Premier Ordre (n multiple de 2)	10 micro secondes
Premier Ordre (n quelconque)	15 micro secondes
Deuxième Ordre	57 micro secondes
Passe bande et coupe bande	70 micro secondes
Trigger	3 micro secondes
Amplificateur	2 micro secondes

Il convient de préciser qu'une acquisition analogique prend à peu près 120 micro secondes qu'il faudra rajouter aux temps ci-dessus.

Remarque :

- Il vaut mieux réaliser un filtre du deuxième ordre avec un facteur de qualité de 0.5 par empilement de deux filtres du premier ordre $57 > 2 \times 15$
- Réaliser un filtre passe bande ou coupe bande à l'aide d'empilement de filtres du deuxième ordre est pénalisant (et plus compliqué) au vue des performances du filtre passe bande $70 < 2 \times 57$

- Pour une entrée analogique à filtrer, il faut donc considérer à la louche 150 microsecondes. En mettant une période d'acquisition de $1ms = 1000$ microsecondes, il nous reste $(1000 - 150) / 1000 = 0.85$ soit 85% de puissance de calcul de notre champion pour traiter le reste de nos problèmes. La fréquence maxi d'analyse de notre filtre sera donc de 50 Hz ($200ms$ de période $= 2 \times dt$) et en prenant un filtre du premier ordre nous aurons

$$f_c = \frac{1}{2\pi} \cdot \frac{1}{0.001} \cdot \frac{1}{n} = \frac{159.15}{n} \text{ Hz}$$
 La condition de Nyquist nous imposera donc d'avoir au moins $n > 3$. En prenant une fréquence de coupure de 10hz, il faut avoir $n = 15.9$ nous prendrons 16 qui est une puissance de 2 et qui permet d'effectuer des calculs plus rapides.

f_c (hz)	40	20	10	5
n	4	8	16	32

Ouf vous l'avez compris, chaque cas est un cas d'espèce générateur de cheveux blancs.

6.6 Un exemple pour jouer avec la bibliothèque 'treat_All_Filters'

Je me permets de vous livrer ici le programme qui m'a permis de développer et déboguer cette bibliothèque. Vous le trouverez dans les exemples.

Ce programme permet de traiter en visualisant les résultats une série de signaux préfabriqués qui permettent de voir le rendu graphique de tel ou tel filtre sur la visu 'traceur' de l'IDE Arduino.

Connectez le PC à un arduino (de préférence prendre un mega qui vous permettra de par ses capacités mémoire accrues de vous livrer à des modifications non létales --plantage-- pour cause de collision entre pile et tas C.F. [Gestion De La Mémoire Dynamique Des Micocontrôleurs Par Les Compilateurs](#). Du même auteur).

On aura au préalable chargé la bibliothèque filtreNum et timerOne.

Le programme permet de choisir les courbes que l'on veut afficher en renvoyant via la voie série les commandes suivantes :

+/- : ajoute ou enlève les courbe désignées après

1..9,A..F : le numéro des courbes

* : toutes les courbes

Ex :

-* : provoquera la disparition de toutes les courbes

+* : provoquera l'apparition de toutes les courbes

-*+5D : élimine toutes les courbes et affiche la N°5 et N°D

+1 : rajoute la courbe N°1

....

A vous de jouer en regardant danser les lignes et en observant le code.

Annexe 1 :

Equations aux différences finies.

La résolution numérique des équations aux dérivées partielles est un domaine qui a depuis longtemps intéressé les ingénieurs. Elle a pris son vrai essor à l'apparition des systèmes de calcul par

itération dopés par la création des calculateurs industriels (ENIAC 1945



30 tonnes ;

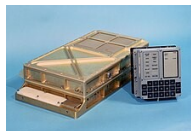
IBM 701 1952



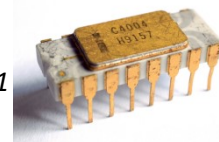
3.2 tonnes ; ordinateur du LEM



1969 l'AGC



32 kg) et par la suite des micro-processeurs (N4004 1971



2 300

transistors 754kHz 4bits; MC6800 1975



4200 transistors 1.5 Mhz 8bits ;... ; i9 2025



4.2 milliards de transistors 3.5Ghz 64bits).

Parmi les multiples contributeurs à ces techniques nous retiendrons ici les mathématiciens :

Léonard Euler (1707-1783)
mathématicien et physicien
suisse qui passa la plus grande
partie de sa vie dans l'Empire
russe et en Prusse.



Il est considéré comme un
éminent mathématicien du
XVIII^{ème} siècle

Il s'est notamment intéressé au
calcul infinitésimal et donc à la
méthode des différences finies.

Carl Ruge (1856-1927)
mathématicien et physicien
allemand :



Améliorent la méthode d'Euler et créent en 1901 la méthode
connue sous le nom de méthode de Runge-Kutta

Martin Wilhem Kutta (1867-
1944) mathématicien
allemand :



Ces méthodes permettent de convertir une équation différentielle (équations faisant intervenir les dérivées successives d'une grandeur) en une équation (souvent du premier degré) permettant de résoudre numériquement de proche en proche les valeurs de la fonction cherchée.

Ces solutions sont très employées dans le monde industriel car, je vous rappelle qu'il n'existe que très peu de solutions exactes exprimables par une fonction connue (dites explicites par les mathématiciens) desdites équations.

Pourquoi différences finies ? : Vous allez voir dans la suite, qu'une fonction dérivée est remplacée par une *différence* de deux valeurs distincts (divisé par un facteur mais ...). Il s'ensuit une transformation de l'équation différentielle en une suite de *différences* (et somme) d'un nombre *fini* de valeurs de la fonction initiale.

Bon fini les palabres place à l'action !

La dérivée d'une fonction continument dérivable s'exprime en un point par les formules

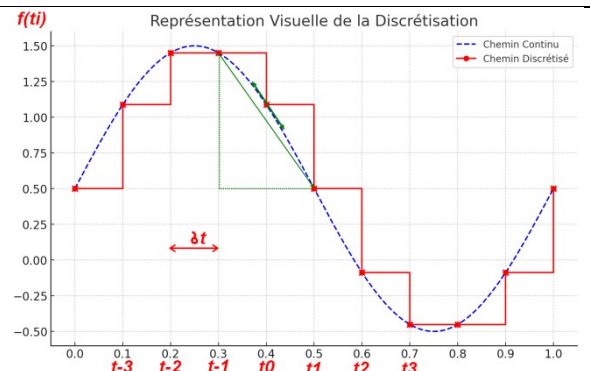
$$\frac{df(t)}{dt} = \lim_{dt \rightarrow 0} \left(\frac{f(t+dt) - f(t-dt)}{2dt} \right) = \lim_{dt \rightarrow 0} \left(\frac{f(t) - f(t-dt)}{dt} \right) = \lim_{dt \rightarrow 0} \left(\frac{f(t+dt) - f(t)}{dt} \right).$$

Ces deux dernières formules sont les limites à gauche et limite à droite correspondant aux nombres dérivés à droite et à gauche. Dans le cas d'une fonction continue continument dérivable (de classe C1 pour les puristes), ces nombres sont égaux et représentent le nombre dérivé au point d'abscisse t .

Nous pouvons effectuer une discrétisation de la fonction par des valeurs prises de manières régulières, tous les δt (avec δt suffisamment petit au regard de la courbure de la courbe initiale).

En notant

$$\begin{cases} \dots \\ f_1 = f(t + \delta t) = f(t_1) \\ f_0 = f(t) = f(t_0) \\ f_{-1} = f(t - \delta t) = f(t_{-1}) \\ \dots \end{cases}$$



Nous pouvons approximer la pente de la tangente à la courbe au voisinage du point d'abscisse t_0

par :

$$\left[\frac{df(t)}{dt} \right]_{t=t_0} \approx \frac{f_1 - f_{-1}}{2\delta t}$$

En prenant l'approximation d'Euler (moins juste mais néanmoins souvent suffisante de la dérivée à

droite

$$\left[\frac{df(t)}{dt} \right]_{t=t_0} \approx \frac{f_1 - f_0}{\delta t}$$

En ce qui concerne la dérivée seconde, $\frac{d^2 f(t)}{dt^2} = \frac{d}{dt} \left(\frac{df(t)}{dt} \right)$ en considérant les points

échantillonnés f_1, f_0, f_{-1} , en calculant les nombres dérivés aux points fictifs $t_0 + \frac{\delta t}{2}$ et $t_0 - \frac{\delta t}{2}$ soit

$$\left[\frac{df(t)}{dt} \right]_{t=t_0+\delta t/2} \approx \frac{f_1 - f_0}{\delta t} \text{ et } \left[\frac{df(t)}{dt} \right]_{t=t_0-\delta t/2} \approx \frac{f_0 - f_{-1}}{\delta t} \text{ nous trouvons}$$

$$\frac{d^2 f(t)}{dt^2} \approx \frac{\frac{f_1 - f_0}{\delta t} - \frac{f_0 - f_{-1}}{\delta t}}{\delta t} = \frac{f_1 - 2f_0 + f_{-1}}{(\delta t)^2}$$

En continuant la gymnastique intellectuelle, nous trouvons la dérivée 3^{ème} qui n'est autre que la dérivée première de la dérivée seconde soit :

$$\frac{d^3 f(t)}{dt^3} \approx \frac{\frac{d^2 f(t+\delta t)}{dt^2} - \frac{d^2 f(t-\delta t)}{dt^2}}{2\delta t}$$

$$\left[\frac{d^3 f(t)}{dt^3} \right]_{t=t_0} \approx \frac{\frac{f_2 - 2f_1 + f_0}{(\delta t)^2} - \frac{f_0 - 2f_{-1} + f_{-2}}{(\delta t)^2}}{2\delta t}$$

$$\left[\frac{d^3 f(t)}{dt^3} \right]_{t=t_0} \approx \frac{f_2 - 2f_1 + 2f_{-1} - f_{-2}}{2(\delta t)^3}$$

Et un cran plus loin ...

$$\frac{d^4 f(t)}{dt^4} \approx \frac{\frac{d^3 f(t+\delta t)}{dt^3} - \frac{d^3 f(t-\delta t)}{dt^3}}{2\delta t}$$

$$\left[\frac{d^4 f(t)}{dt^4} \right]_{t=t_0} \approx \frac{\frac{f_3 - 2f_2 + 2f_0 - f_{-1}}{2(\delta t)^3} - \frac{f_1 - 2f_0 + 2f_{-2} - f_{-3}}{2(\delta t)^3}}{2\delta t}$$

$$\left[\frac{d^4 f(t)}{dt^4} \right]_{t=t_0} \approx \frac{f_3 - 2f_2 - f_1 + 4f_0 - f_{-1} - 2f_{-2} + f_{-3}}{4(\delta t)^4}$$

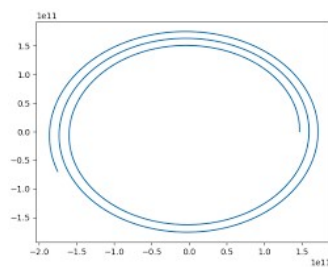
Nota : Dans la vie courante (celle d'un technicien ingénieur ou mathématicien s'intéressant à la modélisation de phénomènes physiques...), il est extrêmement rare de regarder plus loin que la dérivée troisième d'un phénomène... En effet la dérivée première correspond à la vitesse, la dérivée seconde à l'accélération (vitesse d'augmentation de la vitesse) et la troisième à une variation plus ou moins brusque de l'accélération correspondant souvent à une phase

transitoire dans laquelle, les actions influant sur le phénomène subissent des variations sous forme d'échelon (souvent des chocs ou assimilés).

La méthode d'Euler ne s'intéresse qu'à la dérivée première d'un phénomène. La résolution d'un problème différentiel qui met en œuvre (comme la majorité des problèmes) l'accélération (ou la dérivée seconde) du phénomène, diverge assez rapidement.

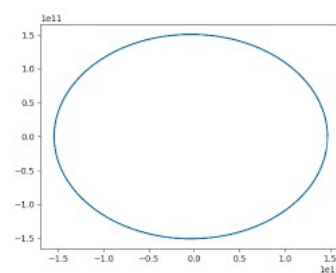
La méthode de Runge-Kutta elle permet d'intégrer avec plus de précision les dérivées d'ordre supérieur (notamment la dérivée seconde) et donc produit un résultat numérique plus conforme à la réalité physique de beaucoup de phénomènes.

Un exemple très classique constitue la résolution de la trajectoire d'un satellite autour de la terre.



Méthode d'euler :

On constate une trajectoire en forme de spirale due à la non prise en compte de la dérivée seconde.

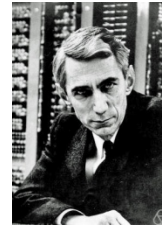


Méthode Runge-Kutta :

La trajectoire calculée peut diverger légèrement due aux erreurs de calculs cumulées (problème de représentation des nombres Réels en informatique) mais reste globalement stable.

Annexe 2 :

Théoreme de Nyquist-Shanon



Harry Nyquist ingénieur américain
(1889-1976)

Claude Shanon ingénieur mathématicien américain
(1916-2001)

L'acquisition d'un signal analogique commence généralement par le discrétiser en prenant une valeur tout les Δt , c'est ce que l'on appelle le processus d'échantillonnage. Il appartient de choisir judicieusement son Δt afin de garantir les deux conditions suivantes :

- 6 Δt doit être suffisamment grand pour que l'électronique (ou le processus numérique) associée soit en dessous de sa fréquence de coupure $\left(f_{ech} = \frac{1}{\Delta t}\right)$.
- 7 Δt doit être suffisamment petit pour permettre la reconstruction du signal initial sans trop le déformer (ou du moins en en conservant sa partie utile).

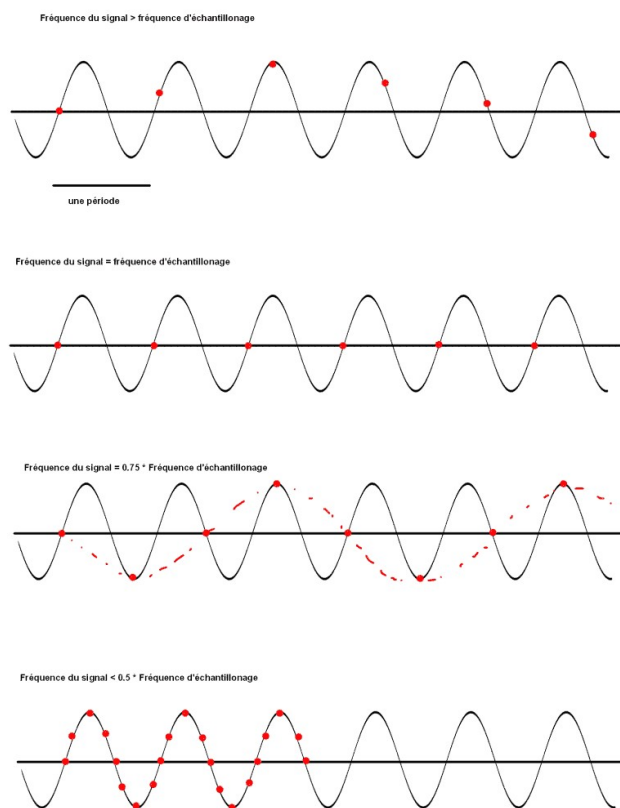
Petite démonstration pseudo-graphique :

Dans le schéma ci-contre, on a visualisé la discrétisation du signal initial (en noir) par des valeurs régulières (en rouge). La reconstruction du signal (relier les points rouges) donne une allure du signal perçu lors de la phase de discrétisation.

On remarque que :

- 8 Si la fréquence de discrétisation est basse, on a un rendu d'un ronflement basse fréquence qui ne correspond à rien au signal initial.
- 9 Si la fréquence de discrétisation est grande, on peut reconstruire le signal par quelque chose qui ressemble au signal d'origine.

Une démonstration très mathématiques permet de montrer ce que l'œil et le bon sens devine : Il faut au moins 2 points d'échantillon par morceau complet de sinusoïde pour pouvoir reconstituer le signal.



A moins on peut créer une pseudo ondulation basse fréquence (c'est le phénomène que l'on utilise en analyse stroboscopique des phénomènes vibratoires en prenant une fréquence d'échantillonnage très proche de la fréquence du phénomène à étudier).

La règle de Nyquits-Shanon est donc simple $f_{ech} = \frac{1}{\Delta t} > 2 \cdot f_{signal}$.

Ce qui nous intéresse le plus souvent est son corolaire : Nous ne pouvons traiter efficacement un signal analogique de fréquence supérieur à 0.5 fois la fréquence d'échantillonnage (qui nous est souvent imposée par la technologie d'acquisition et de traitement informatique utilisée).

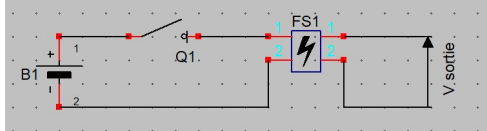
Annexe 3 :

Pourquoi effectuer une caractérisation grâce à la fonction échelon

A : Pourquoi un échelon ?

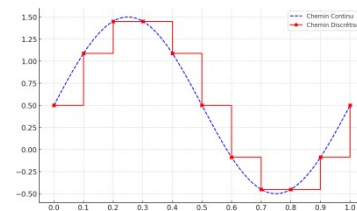
Plusieurs raisons peuvent être invoquées pour lesquelles la notion de caractérisation des filtres est intimement liée à la réponse à un échelon :

1 : Historiquement, l'échelon est la fonction électrique la plus facile à générer :



On ferme brusquement l'interrupteur et l'on regarde le résultat sur un enregistreur au niveau de la sortie. On déduit les valeurs caractéristiques du filtre en analysant la courbe obtenue (pente à l'origine, dépassement)

2 : En fait, en discrétisant la fonction d'entrée, nous considérons que durant l'intervalle de temps entre deux prises de valeurs, la valeur est constante et, elle représente ce que la littérature savante appelle un échelon (la valeur bouge brusquement puis reste constante). La grandeur d'entrée est donc transformée en une succession d'échelons. La connaissance de la réponse du filtre à ce type de signal permet donc de caractériser sa réponse à n'importe quel signal.



3 : Un échelon est en fait une somme infini de fréquences (ce que nous allons essayer de vous montrer dans le B suivant.

B : Analyse fine d'un échelon

B-1 : Petit rappel sur la transformée de fourier



Jean Baptiste Joseph Fourier est un mathématicien et physicien français né le 21 mars 1768 à Auxerre et mort le 17 mai 1830 à Paris.

Il est connu pour avoir déterminé par le calcul la diffusion de la chaleur, en utilisant la décomposition d'une fonction périodique en une série trigonométrique, qui sous certaines conditions, converge vers la fonction. Son aversion pour les températures froides rencontrées lorsqu'il était préfet de l'Isère à certainement contribué à ses études sur la diffusion de la chaleur (la théorie de la thermodynamique ne sera ébauchée par Sadi Carnot --1796-1832-- qu'après la mort de Fourier). Il est aussi l'un des premiers à avoir évoqué la notion d'effet de serre pour l'atmosphère terrestre.

Joseph Fourier a donné son nom aux séries de Fourier, un outil mathématique fondamental pour l'étude des fonctions périodiques et utilisé aussi pour la résolution de certaines équations aux dérivées partielles, et aux intégrales de Fourier, leur extension aux fonctions non périodiques.

Etablir la transformée de Fourier d'une fonction échelon n'est pas aussi simple qu'il le paraît (elle n'est pas vraiment périodique !!). Nous allons donc (que les mathématiciens puristes me pardonnent) essayer de voir cette fonction échelon comme un passage à la limite d'une fonction créneau (fonction créneau avec une fréquence très faible).

Toute fonction périodique (continue ou non !) admet une décomposition en somme de fonctions trigonométriques simples du style

$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \sin(n \cdot \omega_0 \cdot t) + \sum_{n=1}^{\infty} b_n \cos(n \cdot \omega_0 \cdot t)$ <p>ou</p> $f(t) = S_0 + \sum_{n=1}^{\infty} S_n \sin(n \cdot \omega_0 \cdot t + \varphi_n)$ <p>En s'arrangeant bien dans le choix du top 0 de la fonction, on peut avoir une fonction paire ou impaire et donc éliminer les termes en cos. Soit mathématiquement</p> $\forall n > 0, b_n = 0$	<p>Avec $S_n = \sqrt{a_n^2 + b_n^2}$ pour $n \geq 1$ $S_0 = a_0$</p>
--	--

B-2 : Si nous regardons plus précisément la fonction créneau :

	<p>Sa décomposition en sommes de fourrier donne :</p> $f(t) = 0.5 + \frac{2}{\pi} \left(\sum_{n=1}^{\infty} \frac{1}{n} \sin(n \cdot \omega_0 t) \right)$ <p>avec $\omega_0 = 2\pi f_0 = \frac{2\pi}{T_0}$</p> <p>ou plus simplement :</p>
--	--

$$f(t) = 0.5 + \frac{2}{\pi} \left(\sin(\omega t) + \frac{1}{3} \sin(3\omega t) + \frac{1}{5} \sin(5\omega t) + \frac{1}{7} \sin(7\omega t) + \dots \right)$$

B-3 : Revenons à notre fonction échelon

Pour $\begin{cases} t < 0, f(t) = 0 \\ t > 0, f(t) = 1 \end{cases}$

Or, en effectuant un zoom au voisinage de l'origine de cette fonction, on semble retrouver une partie de notre fonction créneau.

Une autre manière de voir les choses, une fonction créneau avec une fréquence de base très faible se rapproche de la fonction échelon. C'est cette vision que nous allons pousser à la limite.

Si les fonctions sont « voisines », il semble évident que leurs représentations de fourrier convergent l'une vers l'autre. Nous pouvons donc dire que la fonction échelon peut être représentée par une

décomposition de fourrier (à la limite pardon M. Fourier) $f(t) = \lim_{\omega_0 \rightarrow 0} \left[0.5 + \frac{2}{\pi} \left(\sum_{n=1}^{\infty} \frac{1}{n} \sin(n \cdot \omega_0 t) \right) \right]$

soit une somme infinie de fonctions sinusoïdales très peu espacées en fréquences.

On peut aussi se dire qu'il est possible d'affiner le « cousinage » entre la fonction échelon et la fonction créneau en prenant un ω_0 de plus en plus petit ce qui correspond à un $T_0 = \frac{2\pi}{\omega_0}$ de plus en plus grand (une fonction créneau n'ayant à la limite plus qu'un plateau à 1) .

B-4 : Conclusion

La caractérisation d'un filtre à la réponse à un échelon, en prenant en compte les propos ci-dessus permet en effet de caractériser le filtre à une gamme de fréquence allant de 0 à l'infini.

Annexe 4 :

Tracés simplifiés de Bode

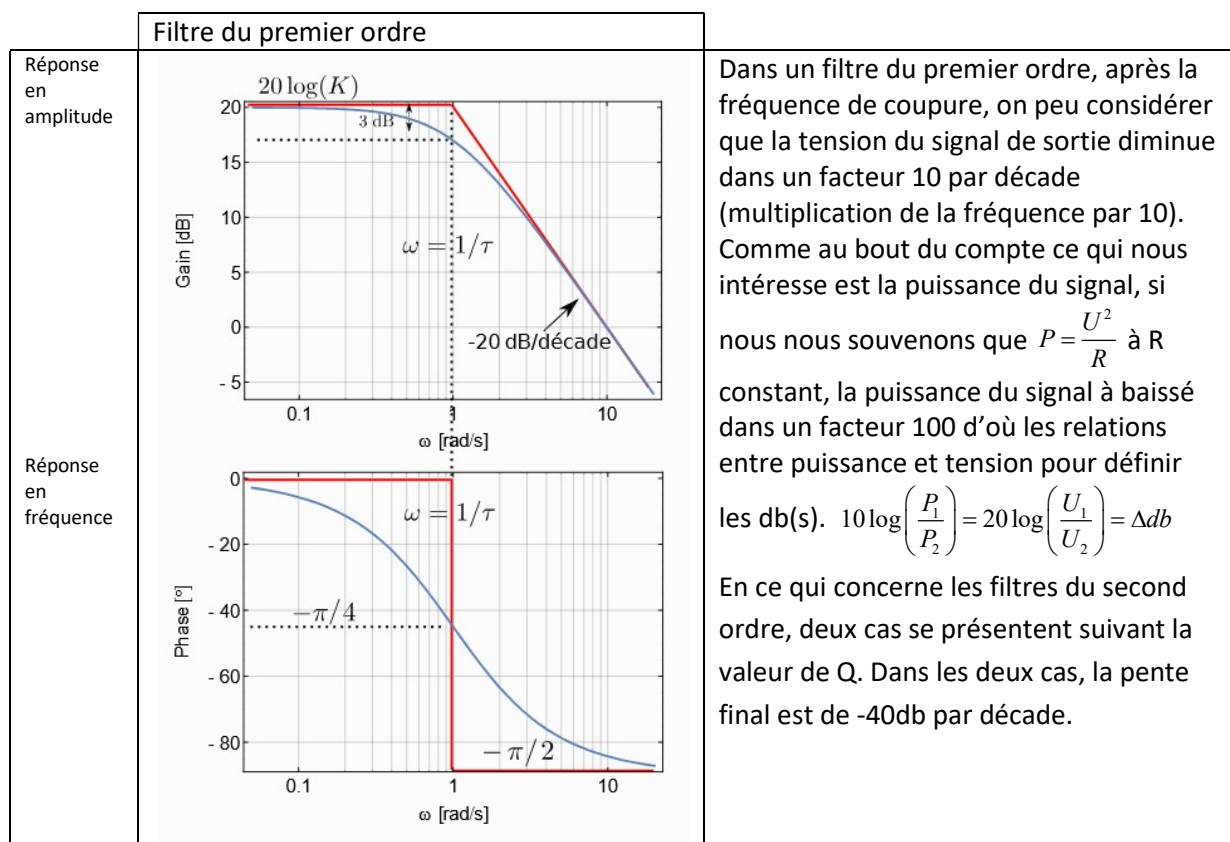


Hendrik Wade Bode (24 décembre 1905 - 22 juin 1982) est un ingénieur et inventeur américain d'origine néerlandaise. Pionnier de la régulation moderne et des télécommunications.

Il est connu entre autre pour avoir créé une méthode de tracé des actions fréquentielles d'un filtre (et d'une association de filtres) en ne s'intéressant qu'aux points caractéristiques de ces filtres tant dans le domaine du gain et de la phase de la sortie par rapport à l'entrée :

- 10 Pour une fréquence nulle (ou très basse C.F. les ****nota**** qui suivent)
- 11 A la fréquence de coupure (ou caractéristique)
- 12 Pour une fréquence tendant vers l'infini (du moins grande devant la plage d'utilisation)

Dans le cas des filtres du premier ordre et du second ordre avec K le gain statique nous avons les représentations simplifiées en db (décibel) suivantes (courbes rouges):



**** nota ****

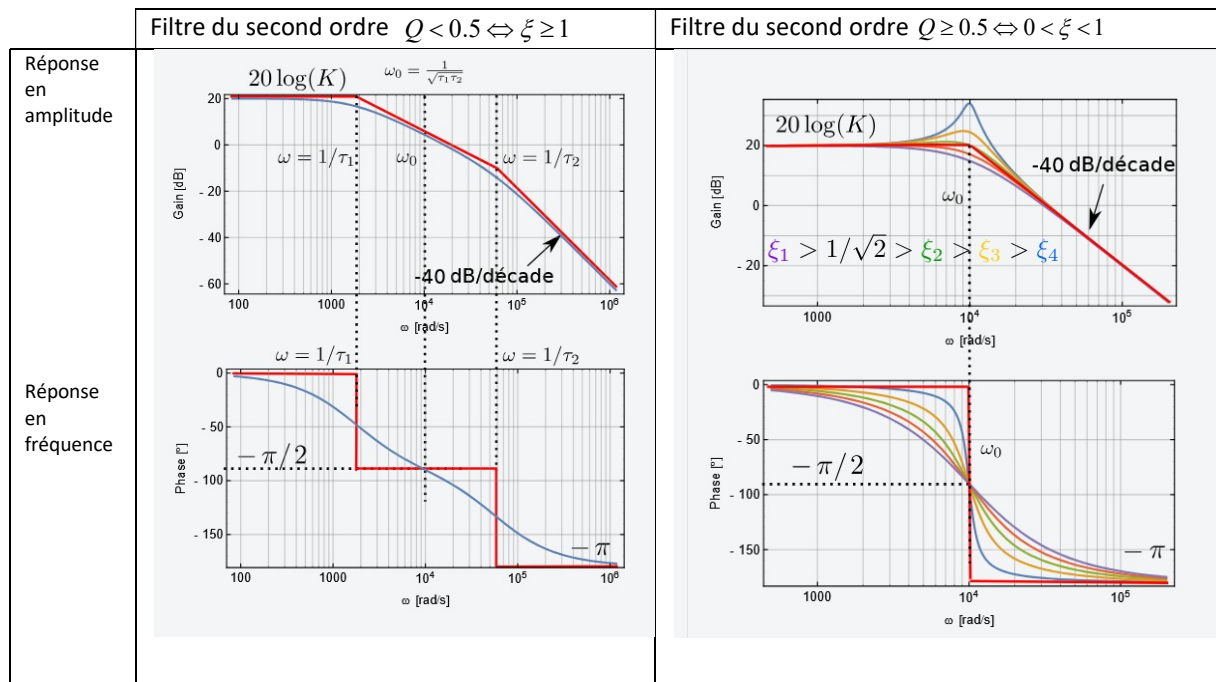
- 13 Dans la littérature spécialisée, on rencontre généralement deux concepts : le facteur d'amortissement ξ et le facteur de qualité, noté Q avec $\xi = \frac{1}{2Q}$. Certains préfèrent utiliser des termes proches de 1, plutôt que des valeurs de Q qui peuvent dépasser 20, voire 100. Ce choix est avant tout une question de perspective. Cependant, il peut introduire une certaine confusion pour les néophytes, un public auquel je m'adresse actuellement et qui n'a pas besoin de complexité supplémentaire.
- 14 L'axe des abscisses peut être gradué en fréquence (Hz) ou en pulsation (rd/s) $\omega = 2\pi f$ ce qui ne change rien dans les tracés futurs (si ce n'est de conserver une homogénéité dans toute l'étude).
- 15 Si l'axe des ordonnées est gradué (sur une échelle logarithmique) en tension (ce qui est le cas dans nos fichiers Excel), les pentes des droites tracées sur le graphique (qui n'apparaissent droites uniquement parce que nous sommes dans une échelle log-log), relient 2 points $(x_1, y_1) \leftrightarrow (x_2, y_2)$ tel que $\frac{x_2}{x_1} = 10; \frac{y_1}{y_2} = \begin{cases} 10 & \text{pour une pente à } -20\text{db} \\ 100 & \text{pour une pente à } -40\text{db} \end{cases}$. Autrement dit en résumant : 20db correspond à une décade en tension.
- 16 Dans ce type de graphique (log-log) il est impossible de visualiser le point (0,0). Toute bonne calculatrice fera une crise de nerf en calculant $\log(0)$ et elle aura raison (du moins ses concepteurs)! Un mathématicien sans calculatrice (si-si cela existe du moins existait avant le



1^{er} Février 1972 date de lancement de la HP-35

) vous dira que

$\lim_{x \rightarrow 0^+} \log(x) \rightarrow -\infty$ et, que le domaine de définition du \log $Def(\log) =]0, +\infty[$ exclut toutes les valeurs inférieures ou égales à 0.



Les filtres du second ordre avec un $Q < 0.5$ sont rarement employés (ils sont en fait équivalents à deux filtres du premier ordre en cascade). De plus on recherche plus souvent la sélectivité que le flou autour d'un point précis. Cette sélectivité est donnée par un facteur $Q > 0.5$ voir plus.

Pour le tracé d'un filtre complexe créé par succession de filtres simples, on trace de gauche à droite en repérant les points de cassure du squelette (courbe rouges). On peut dire en règle générale que l'on ajoute les pentes (dérivée du gain) et que l'on ajoute les corrections en phase.